Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

LatticeCORE

CORDIC IP Core User's Guide

August 2012
IPUG81_01.3

Table of Contents

Introduction

This user's guide provides a description of Lattice's Coordinate Rotation Digital Computer (CORDIC) IP core. The CORDIC IP core is configurable and supports several functions, including rotation, translation, sin and cos, and arctan. Two architecture configurations are supported for the arithmetic unit: parallel, in which the output data is calculated in a single clock cycle, and word-serial, in which the output data is calculated over multiple clock cycles. The input and output data widths and computation iterative numbers are configurable over a wide range of values. The IP core uses full precision arithmetic internally while supporting variable output precision and several choices of rounding algorithms.

Quick Facts

Table 1-1 through Table 1-7 give quick facts about the CORDIC IP core for LatticeECP™, LattceECP2™, LatticeECP2M™, LatticeECP3™, LatticeSC/M™, LatticeXP™, and LatticeXP2™ devices, respectively.

Table 1-1. CORDIC IP Core for LatticeECP Devices Quick Facts

		CORDIC IP Configuration			
		Rotate Parallel	**Translate Parallel**	**Rotate Serial**	**Translate Serial**
Core Requirements	FPGA Families Supported	LatticeECP			
	Minimal Device Needed	LFECP6E-3T114	LFECP6E-3T114	LFECP6E-3T114	LFECP6E-3T114
Resource Utilization	Targeted Device	LFECP20E-5F484C	LFECP20E-5F484C	LFECP20E-5F484C	LFECP20E-5F484C
	Data Path Width	16	16	16	16
	LUTs	1300	1200	600	700
	sysMEM EBRs	0	0	0	0
	Registers	1300	1200	300	400
Design Tool Support	Lattice Implementation	Lattice Diamond™ 1.0 or ispLEVER® 8.1			
	Synthesis	Synopsys® Synplify™ Pro for Lattice D-2009.12L-1			
	Simulation	Aldec® Active-HDL™ 8.2 Lattice Edition			
		Mentor Graphics® ModelSim™ SE 6.3F			

Table 1-2. CORDIC IP Core for LatticeECP2 Devices Quick Facts

		CORDIC IP Configuration			
		Rotate Parallel	**Translate Parallel**	**Rotate Serial**	**Translate Serial**
Core Requirements	FPGA Families Supported	LatticeECP2			
	Minimal Device Needed	LFE2-6E-5T144C	LFE2-6E-5T144C	LFE2-6E-5T144C	LFE2-6E-5T144C
Resource Utilization	Targeted Device	LFE2-20E-7F484C	LFE2-20E-7F484C	LFE2-20E-7F484C	LFE2-20E-7F484C
	Data Path Width	16	16	16	16
	LUTs	1300	1300	600	600
	sysMEM EBRs	0	0	0	0
	Registers	1200	1200	300	400
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1			
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1			
	Simulation	Aldec Active-HDL 8.2 Lattice Edition			
		Mentor Graphics ModelSim SE 6.3F			

Table 1-3. CORDIC IP Core for LatticeECP2M Devices Quick Facts

		CORDIC IP Configuration			
		Rotate Parallel	**Translate Parallel**	**Rotate Serial**	**Translate Serial**
Core Requirements	FPGA Families Supported	LatticeECP2M			
	Minimal Device Needed	LFE2M20E-5F256C	LFE2M20E-5F256C	LFE2M20E-5F256C	LFE2M20E-5F256C
Resource Utilization	Targeted Device	LFE2M20E-7F484C	LFE2M20E-7F484C	LFE2M20E-7F484C	LFE2M20E-7F484C
	Data Path Width	16	16	16	16
	LUTs	1300	1300	600	600
	sysMEM EBRs	0	0	0	0
	Registers	1200	1200	300	400
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1			
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1			
	Simulation	Aldec Active-HDL 8.2 Lattice Edition			
		Mentor Graphics ModelSim SE 6.3F			

Table 1-4. CORDIC IP Core for LatticeECP3 Devices Quick Facts

		CORDIC IP Configuration			
		Rotate Parallel	**Translate Parallel**	**Rotate Serial**	**Translate Serial**
Core Requirements	FPGA Families Supported	LatticeECP3			
	Minimal Device Needed	LFE3-17EA-6FTN256CES	LFE3-17EA-6FTN256CES	LFE3-17EA-6FTN256CES	LFE3-17EA-6FTN256CES
Resource Utilization	Targeted Device	LFE3-70E-8FN484CES	LFE3-70E-8FN484CES	LFE3-70E-8FN484CES	LFE3-70E-8FN484CES
	Data Path Width	16	16	16	16
	LUTs	1300	1300	600	700
	sysMEM EBRs	0	0	0	0
	Registers	1300	1200	300	400
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1			
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1			
	Simulation	Aldec Active-HDL 8.2 Lattice Edition			
		Mentor Graphics ModelSim SE 6.3F			

Table 1-5. CORDIC IP Core for LatticeSC/M Devices Quick Facts

		CORDIC IP Configuration			
		Rotate Parallel	**Translate Parallel**	**Rotate Serial**	**Translate Serial**
Core Requirements	FPGA Families Supported	LatticeSC/M			
	Minimal Device Needed	LFSC3GA15E-5F256C	LFSC3GA15E-5F256C	LFSC3GA15E-5F256C	LFSC3GA15E-5F256C
Resource Utilization	Targeted Device	LFSC3GA25E-7F900C	LFSC3GA25E-7F900C	LFSC3GA25E-7F900C	LFSC3GA25E-7F900C
	Data Path Width	16	16	16	16
	LUTs	1700	1700	900	1000
	sysMEM EBRs	0	0	0	0
	Registers	1300	1300	400	400
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1			
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1			
	Simulation	Aldec Active-HDL 8.2 Lattice Edition			
		Mentor Graphics ModelSim SE 6.3F			

Table 1-6. CORDIC IP Core for LatticeXP Devices Quick Facts

		CORDIC IP Configuration			
		Rotate Parallel	Translate Parallel	Rotate Serial	Translate Serial
Core Requirements	FPGA Families Supported	LatticeXP			
	Minimal Device Needed	LFXP3C-3Q208C	LFXP3C-3Q208C	LFXP3C-3Q208C	LFXP3C-3Q208C
Resource Utilization	Targeted Device	LFXP20E-5F484C	LFXP20E-5F484C	LFXP20E-5F484C	LFXP20E-5F484C
	Data Path Width	16	16	16	16
	LUTs	1300	1200	600	700
	sysMEM EBRs	0	0	0	0
	Registers	1300	1200	300	400
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1			
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1			
	Simulation	Aldec Active-HDL 8.2 Lattice Edition			
		Mentor Graphics ModelSim SE 6.3F			

.

Table 1-7. CORDIC IP Core for LatticeXP2 Devices Quick Facts

		CORDIC IP Configuration			
		Rotate Parallel	Translate Parallel	Rotate Serial	Translate Serial
Core Requirements	FPGA Families Supported	LatticeXP2			
	Minimal Device Needed	LFXP2-5E-5M132C			
Resource Utilization	Targeted Device	LFXP2-30E-7F484C			
	Data Path Width	16	16	16	16
	LUTs	1300	1300	600	600
	sysMEM EBRs	0	0	0	0
	Registers	1200	1200	300	400
Design Tool Support	Lattice Implementation	Lattice Diamond 1.0 or ispLEVER 8.1			
	Synthesis	Synopsys Synplify Pro for Lattice D-2009.12L-1			
	Simulation	Aldec Active-HDL 8.2 Lattice Edition			
		Mentor Graphics ModelSim SE 6.3F			

Features

- Functions supported:
  - Vector rotation (polar to rectangular)
  - Vector translation (rectangular to polar)
  - Sin and cos
  - Arctan
- Input data widths from 8 to 32 bits
- Configurable number of iterations used to derive output from 4 to 32

- Optional pre-rotation module
- Optional amplitude compensation scaling module to compensate for the CORDIC algorithm's output amplitude scale factor
- Selectable rounding algorithm: truncation, rounding up, rounding away from zero, convergent rounding
- Selectable parallel architectural configuration for throughput optimization
- Selectable word-serial architectural configuration for area optimization
- Signed 2's complement data
- Optional clock enable (ce) and synchronous reset (sr) control signals
- Full precision internal arithmetic

Functional Description

This chapter provides a functional description of the CORDIC IP core.

General Description of the CORDIC Algorithm

The CORDIC algorithm is an iterative method that uses simple arithmetic operations such as addition, subtraction, bit shift and table look up to perform hyperbolic and trigonometric functions.The CORDIC algorithm was initially designed to perform a vector rotation, where the vector (x, y) is rotated through the angle θ yielding a new vector (x', y') . Using a matrix form, a planar rotation for a vector of (x, y) is defined as:

$$x' = x\cos\theta - y\sin\theta \qquad (1)$$

$$y' = y\cos\theta + x\sin\theta$$

Note that θ is the angle that is to be traversed. With the CORDIC algorithm, the traversal is accomplished in iterative steps in which each step completes a small part of the rotation.

A single step is defined by the following equation:

$$x_{i+1} = \cos\theta_i(x_i - y_i\tan\theta_i) \qquad (2)$$

$$y_{i+1} = \cos\theta_i(y_i + x_i\tan\theta_i)$$

The number of multipliers required is reduced by selecting the angle steps such that the tangent of a step is a power of 2. The angle for each step is given by:

$$\theta_i = \arctan(1/2^i) \qquad (3)$$

Multiplying or dividing by a power of 2 can be implemented using a simple shift operation.

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

$$\sum_{i=0}^{\infty} d_i\theta_i = \theta \text{ where } d_i = \{-1;+1\} \qquad (4)$$

This results in the following equation for $\tan\theta_i$:

$$\tan\theta_i = d_i 2^{-i} \qquad (5)$$

Combining equations 2 and 5 results in:

$$x_{i+1} = \cos\theta_i(x_i - y_i \cdot d_i \cdot 2^{-i}) \qquad (6)$$

$$y_{i+1} = \cos\theta_i(y_i + x_i \cdot d_i \cdot 2^{-i})$$

The iterative rotation can now be expressed as:

$$x_{i+1} = K_i(x_i - y_i \cdot d_i \cdot 2^{-i}) \qquad (7)$$

$$y_{i+1} = K_i(y_i + x_i \cdot d_i \cdot 2^{-i})$$

where:

$$K_i = \cos(\tan^{-1}2^{-i}) = 1/(\sqrt{1 + 2^{-2i}})$$

$$di = \pm 1$$

The CORDIC rotator is normally operated in one of two modes. The first, called rotation, rotates the input vector by a specified angle. The second mode, called vectoring, rotates the input vector to the x-axis while recording the angle required to make that rotation.

For rotation mode, the CORDIC equations are:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i} \quad (8)$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

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

where $d_i = -1$ if $z_i < 0$, +1 otherwise. Here z_i is the residual angle in the angle accumulator with the initial value z_0 as the angle to be rotated.

In vectoring mode, the CORDIC 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. For vectoring mode, the CORDIC equations are:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i} \quad (9)$$

$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i}$$

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

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

In sin/cos mode, the unit vector is rotated by the input phase angle θ generating the output vector $(\cos(\theta), \sin(\theta))$. The rotation mode CORDIC operation can simultaneously compute the sine and cosine of the input angle θ. Setting the x component to 1 and y component to zero reduces the rotation mode. This results the equations 11 from equations 1:

$$x' = \cos\theta \quad (10)$$

$$y' = \sin\theta$$

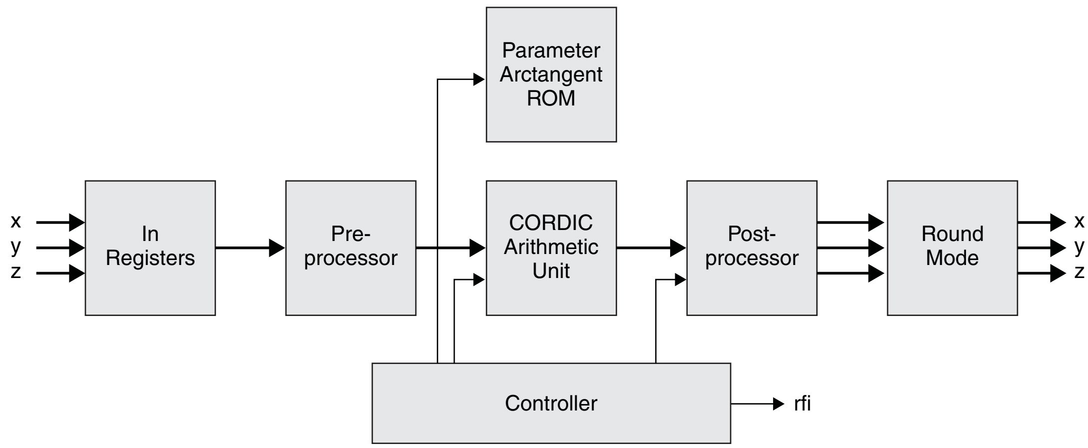
In arctangent mode, $\theta = \arctan(y_0 / x_0)$ is directly computed using the vectoring mode if the angle accumulator is initialized with zero.

$$z_n = z_0 + \arctan(y_0 / x_0) \quad (11)$$

Block Diagram

Figure 2-1 shows a block diagram of the CORDIC IP Core.

Figure 2-1. CORDIC IP Core Block Diagram



Data Path

Pre-processor

The CORDIC rotation and vectoring algorithms are limited to rotation angles between -π/2 and π/2 This limitation is due to the use of 2° for the tangent in the first iteration. For composite rotation angles larger than π/2, an additional rotation is required.

CORDIC Arithmetic Unit

The CORDIC arithmetic unit performs the actual CORDIC algorithm. Two architecture configurations are available for the arithmetic unit: parallel (with single-cycle data throughput) and word-serial (with multiple-cycle throughput). The parallel configuration has a pipeline-structured core and can perform a CORDIC transformation each clock cycle, producing a new output every cycle. In contrast with the parallel structure, word-serial architecture produces a new output every N cycles. Here N is the user input in the IPexpress™ GUI for the "Iteration Number" parameter.

Arctan ROM

The arc tangent ROM stores the $\tan^{-1}(2^{-i})$ values. Its data width is variable, address width is log2(number of iterations-1), address depth is 2 ^ log2(number of iterations-1).

Controller

The controller module control generates all signals necessary for carrying out the iterations, including ROM addressing, ready for input (rfi) and output valid (outvalid). I/O port definition details are explained in Table 2-5.

Post-processor

The CORDIC algorithm introduces a scale factor that causes a magnitude gain that must be compensated for at the end (see Equation 8). The post-processor module contains logic to correct the scale factor. In addition, it corrects the phase rotation introduced by the pre-processor module (if present).

Rounding

The rounding module provides four types of rounding, depending on the ROUNDING parameter:

- **None (truncation)** – Discards all bits to the right of the output least significant bit and leaves the output uncorrected.
- **Rounding up** – Rounds up if the fractional part is exactly one-half.
- **Rounding away from zero** – Rounds away from zero if the fractional part is exactly one-half.

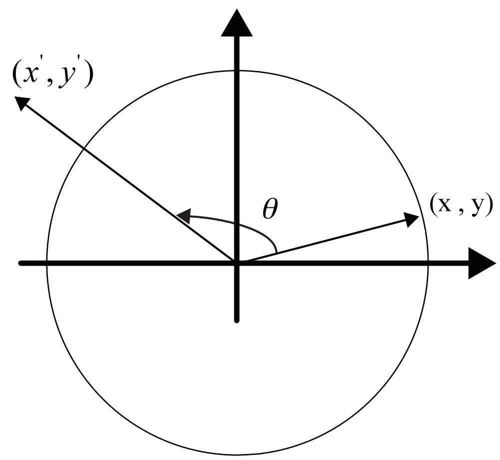
- **Convergent rounding** – Rounds to the nearest even value if the fractional part is exactly one-half.

CORDIC Functions

Vector Rotation

Polar to Rectangular Translation: In vector rotation mode, the input vector (x, y) is rotated by a specified angle, θ, giving the a new output vector, (x', y') . Because of the CORDIC algorithm scale factor, a magnitude gain will be introduced as shown in Figure 2-2. This magnitude gain is compensated for by the CORDIC IP post-processor module.

Figure 2-2. Vector Rotation



The inputs, xin, yin and phasein, are limited to the ranges given in Table 2-1. Inputs outside the ranges will produce unpredictable results.

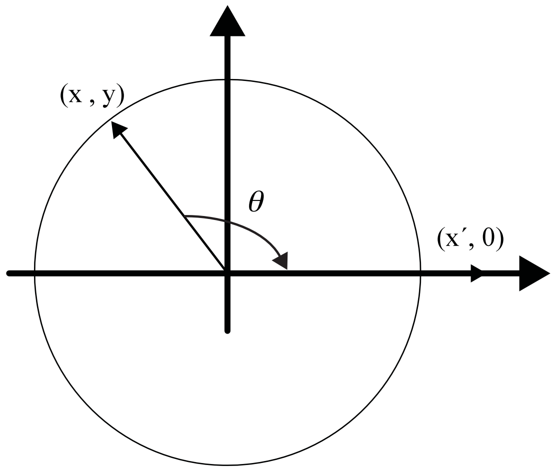
Table 2-1. Vector Rotation Input/Output

Signal	Description
xin	Input X Coordinate Range: $-1 \le$ xin ≤ 1
yin	Input Y Coordinate Range: $-1 \le$ yin ≤ 1
phasein	Input Rotation Angle Range: $-\pi \le$ Phasein $\le \pi$
xout	Output X Coordinate Range: $-\sqrt{2} \le$ xout $\le \sqrt{2}$
yout	Output Y Coordinate Range: $-\sqrt{2} \le$ yout $\le \sqrt{2}$

Vector Translation

Rectangular to Polar Translation: In vector translation mode, the input vector (x, y) is rotated through whatever angle is necessary to align the result vector with the x-axis, as shown in Figure 2-3. Output is the angle rotated and the magnitude on the x-axis after rotation.

Figure 2-3. Vector Translation



The inputs, xin and yin, are limited to the ranges given in Table 2-2. Inputs outside the ranges will produce unpredictable results.

Table 2-2. Vector Translation Input/Output

Signal	Description
xin	Input X Coordinate Range: $-1 \le xin \le 1$
yin	Input Y Coordinate Range: $-1 \le yin \le 1$
xout	Output Magnitude Range: $-\sqrt{2} \le xout \le \sqrt{2}$
phaseout	Output Phase Range: $-\pi \le Phaseout \le \pi$

Sin and Cos

In sin/cos mode, the unit vector is rotated by the input phase angle θ providing the output vector $(\cos(\theta), \sin(\theta))$.

The input angle, phasein, is limited to the range given in Table 2-3. Inputs outside this range will produce unpredictable results.

Table 2-3. Sin and Cos Input/Output

Signal	Description
phasein	Input Phase Range: $-\pi \le Phasein \le \pi$
xout	Output cos(θ) Range: $-1 \le xout \le 1$
yout	Output sin(θ) Range: $-1 \le yout \le 1$

Arctan

In arctan mode, the input vector, (x, y) is rotated until the y component is zero, yielding the output angle, $\arctan(y/x)$.

The inputs xin and yin are limited to the ranges given in Table 2-4. Inputs outside the ranges will produce unpredictable results.

Table 2-4. Arctan Input/Output

Signal	Description
xin	Input X Coordinate Range: $-1 \le xin \le 1$
yin	Input Y Coordinate Range: $-1 \le yin \le 1$
phaseout	Output Phase Range: $-\pi \le Phaseout \le \pi$

Interface Diagram

The top-level interface diagram for the CORDIC IP core is shown in Figure 2-4. The description of the Input/Output (I/O) ports for the CORDIC IP core is provided in Table 2-5.

Figure 2-4. Top-Level Interface for CORDIC IP Core

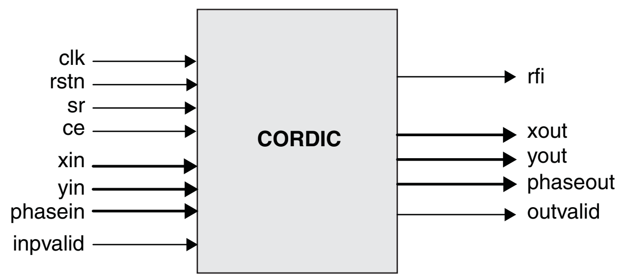


Table 2-5. Top-Level Port Definitions

Port	Bits	I/O	Description
General I/Os			
clk	1	I	System clock for data and control inputs and outputs.
rstn	1	I	System-wide asynchronous active-low reset signal.
xin	DINWIDTH	I	X component of input sample
yin	DINWIDTH	I	Y component of input sample
phasein	DINWIDTH	I	Phase component of input sample
inpvalid	1	I	Input valid signal. The input data is read in only when inpvalid is high.
xout	DOUTWIDTH	O	X component of output sample
yout	DOUTWIDTH	O	Y component of output sample
phaseout	DOUTWIDTH	O	Y component of output sample
outvalid	1	O	Output data qualifier. Output data is valid only when this signal is high.
rfi	1	O	Ready for input. This output, when high, indicates that the IP core is ready to receive the next input data. A valid data may be applied at xin, yin and phasein only if rfi was high during the previous clock cycle.
Optional I/Os			
ce	1	I	Clock Enable. Independent.
sr	1	I	Synchronous Reset. Independent.

Configuring the CORDIC IP Core

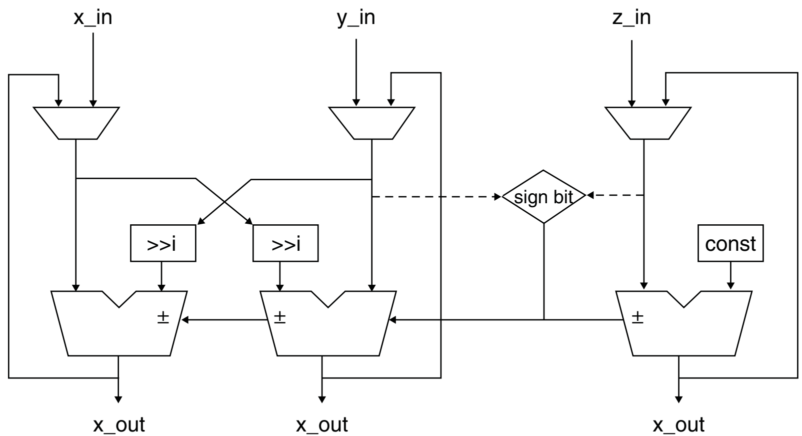
Basic Options

The options for mode, architecture, number of iterations and compensation are independent and specified in the "Basic Options" tab of the GUI. Refer to "Basic Options Tab" on page 20.

Architecture Specification

The CORDIC IP core provides two architecture configurations for the arithmetic unit: parallel (with single cycle data throughput) and word-serial (with multiple-cycle throughput). Because of the pipelined structure, the core can perform a CORDIC transformation each clock cycle, thus producing a new output every cycle. In contrast with parallel structures, word-serial architecture produces a new output every N cycles. Figure 2-5 shows a basic CORDIC arithmetic unit.

Figure 2-5. Basic CORDIC Arithmetic Unit



Iterations Specification

Parameter iteration specifies the number of internal add-sub iterations performed by the CORDIC processor in deriving the result. It determines the accuracy of the output: if the number is larger, the accuracy of the output is higher.

Pre-rotation Specification

When the pre-rotation module is selected, the CORDIC operational range extends to the full circle; otherwise the operational range is limited between -π/2 and π/2. Angle ranges outside the ranges will produce an unpredictable result if the pre-rotation module is not selected. The following describes an initial pre-rotation ±π/2:

$$x' = -d \bullet y \quad (12)$$

$$y' = d \bullet x$$

$$z' = z + d \bullet \pi/2$$

Compensation Specification

In the CORDIC algorithm, the magnitude outputs, xout and yout, are generated with a magnitude gain. The compensation module provides three configurations to compensate for the CORDIC magnitude scale factor.

- **None** – The outputs xout and yout will not be compensated. It is the user's obligation to compensate and scale for the magnitude outputs gain introduced by the CORDIC algorithm. Refer to page 9 of this document for details, especially the 'K' factor in equation 7.
- **LUT-based** – The outputs xout and yout are compensated using a LUT-based multiplier.
- **DSP-based** – The outputs xout and yout are compensated using a DSP-based multiplier.

Advanced Options

The controls in this tab are used to define the various data widths and rounding methods used in the data path. The widths of the input data and output data can be defined independently.

Round Method Specification

The CORDIC IP core provides four rounding modes. Examples of round method are given in Table 2-6.

- **Truncation** – The outputs, xout, yout and phaseout, are truncated. The LSBs are removed to match the specified output width.
- **Rounding up** – The outputs, xout, yout and phaseout, are rounded up (0.5 rounded up).
- **Rounding away from zero** – The outputs, xout, yout and phaseout, are rounded (0.5 rounded up, -0.5 rounded down).
- **Convergent rounding** – The outputs, xout, yout and phaseout, are rounded towards the nearest even number.

Table 2-6. Round Method

	Truncation	Rounding Up	Rounding Away from Zero	Convergent Rounding
1.50	1	2	2	2
-1.50	-2	-1	-2	-2
0.50	0	1	1	0
-0.50	-1	0	-1	0
0.25	0	0	0	0
-0.25	-1	0	0	0
0.65	0	1	1	0

Input/Output Width Specification

The input/output data widths can be configured in the range 8 to 32 bits.

Data Format Specification

The data signals are: xin, yin, xout and yout. The input data signals, xin and yin, must be in the range [-1,1]. Input data outside the range will produce unpredictable results.

- **Input Data Signals**
  Input data signals are represented in decimal format using bus format (as little endian). For N-bit input data signal, the (N-2) LSB represent the fractional component to the left of the decimal place and the MSB represents the sign bit.

  For example, when the DINWIDTH is 8, +1 and –1 are represented as:

  “01000000” => 01.000000 => +1.0
  “11000000” => 11.000000 => -1.0

  When the DINWIDTH is 12, +1 and –1 are represented as:

  “010000000000” => 01.0000000000 => +1.0
  “110000000000” => 11.0000000000 => -1.0

- **Output Data Signals**
  If compensation is LUT- based or DSP-based, the output data signal format is the same as the input data signal format. The range of the output data signal is $[-\sqrt{2}, \sqrt{2}]$.

  For N-bit output data signal, the (N-2) LSB represent the fractional component to the left of the decimal place and the MSB represents the sign bit.

For example, when the DOUTWIDTH is 8, in the data format, +1 and –1 are represented:

"01000000" => 01.000000 => +1.0
"11000000" => 11.000000 => -1.0

When the DOUTWIDTH is 12, in the data format, +1 and –1 are represented:

"010000000000" => 01.0000000000 => +1.0
"110000000000" => 11.0000000000 => -1.0

If compensation is None, the output data signals format is different from the input data signals. Due to the magnitude gain introduced by the CORDIC algorithm, without the compensation, the range of the output data signal can be larger than 2 or less than -2, so it will need 2 bits to represent the decimal number.

For the N-bit output data signal, the (N-3) LSB represent the fractional component to the left of the decimal place and the MSB represents the sign bit.

For example, when the DOUTWIDTH is 8, in the data format, +1 and –1 are represented:

"00100000" => 001.00000 => +1.0
"11000000" => 111.00000 => -1.0

When the DOUTWIDTH is 12, in the data format, +2 and –2 are represented:

"010000000000" => 010.000000000 => +2.0
"110000000000" => 110.000000000 => -2.0

+2.25 and –2.25 are represented:

"010010000000" => 010.010000000 => +2.25
"101110000000" => 101.110000000 => -2.25

Phase Format Specification

- **Phase Signals**
  The phase signals are phasein and phaseout. The input phase signal, phasein, must be in the range [- ¼, ¼]. Input phase outside this range will produce unpredictable results.

  The phase signals, phasein and phaseout, are always the same representation.

  For N-bit phase signal, the (N-3) LSB represents the fractional component to the left of the decimal place and the MSB represents the sign bit.

  For example, when the DINWIDTH is 10, in the data format, +¼ and -¼ are represented:

  "0110010010" => 011.0010010 => +π
  "1001101110" => 100.1101110 => -π

  When the DINWIDTH is 13, in the data format, +¼ and -¼ are represented:

  "0110010010001" => 011.0010010001 => +π
  "1001101101111" => 100.1101101111 => -π

Synthesis Options Specification

There are two synthesis options for controlling IP generation flow, the "Frequency constraint" and "Pipelining and retiming". The "Pipelining and retiming" option is used to move existing registers in order to balance the delays between registers. Users can adjust these two options to optimize for timing and area.

Timing Specifications

Timing diagrams for the CORDIC IP core are given in the Figure 2-6, Figure 2-7, and Figure 2-8.

Figure 2-6. Timing Diagram for Parallel CORDIC (Rotation Mode) with Continuous Input

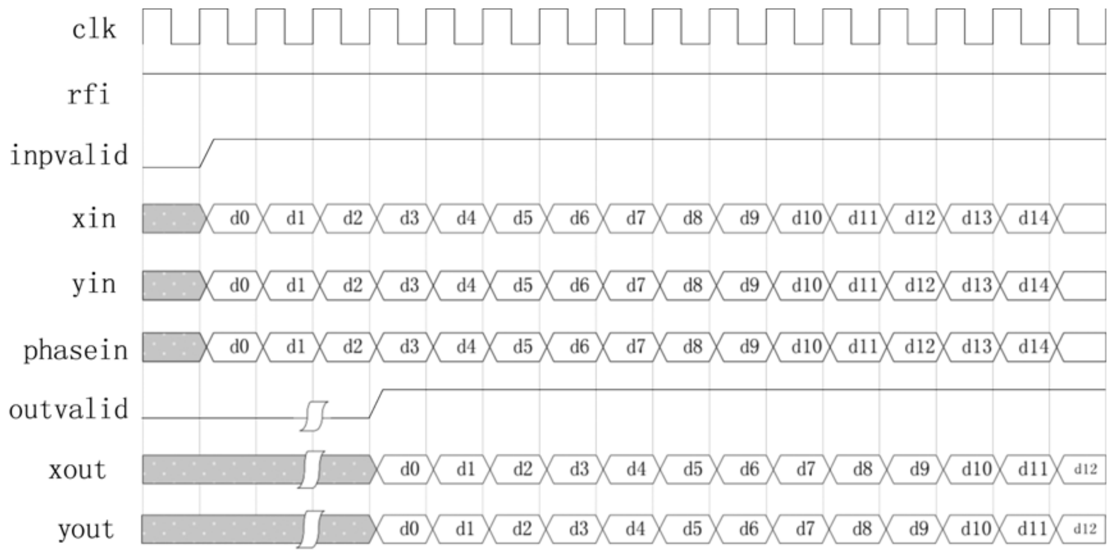


Figure 2-7. Timing Diagram for Parallel CORDIC (Sin/Cos Mode) with Gapped Inputs

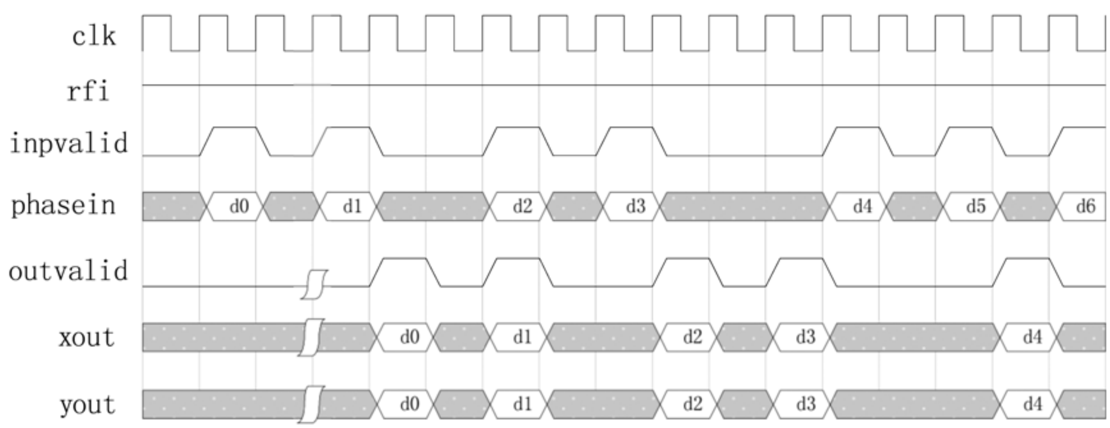
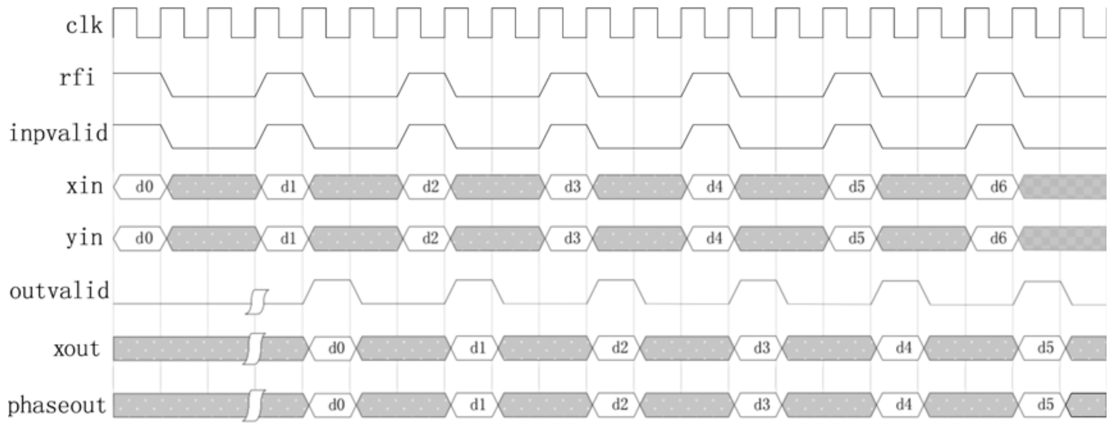


Figure 2-8. Timing Diagram for Serial CORDIC (Translation Mode)



Parameter Settings

The IPexpress tool is used to create IP and architectural modules in the Diamond and ispLEVER software. Refer to "IP Core Generation" on page 23 for a description on how to generate the IP.

Table 3-1 provides the list of user configurable parameters for the CORDIC IP core. The parameter settings are specified using the CORDICI IP core Configuration GUI in IPexpress.

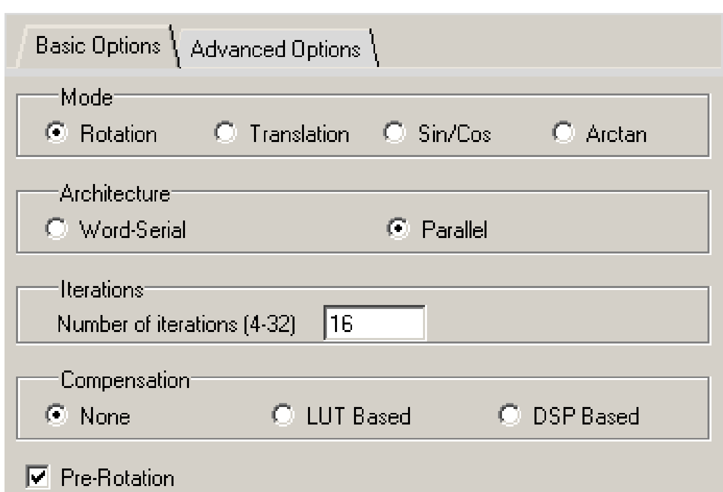
Table 3-1. Parameter Specifications for the CORDIC IP Core

Parameter	Range/Options	Default
CORDIC Specifications		
Mode	Rotate, Translate, Sin/Cos, Arctan	Rotate
Architecture	Word-Serial, Parallel	Parallel
Iterations	4 - 32	16
Compensation	None, LUT based, DSP based	None
Prerotation	Disable, Enable	Enable
I/O Specifications		
Input data width	8 - 32	16
Output data width	8 - 32	16
Precision Control		
Roundmethod	Truncation, Rounding up, Round away from zero, Convergent rounding	Truncation
Optional Ports		
Synchronous Reset	Disable, Enable	Disable
Clock Enable	Disable, Enable	Disable
Synthesis Options		
Frequency constraint	1- 400	250
Pipelining and retiming	Disable, Enable	Disable

Basic Options Tab

Figure 3-1 shows the CORDIC Basic Options tab in the IPexpress tool.

Figure 3-1. CORDIC Basic Options Tab



Mode

Specifies the CORDIC function to be performed.

Architecture

Specifies the architecture configuration for the CORDIC core: parallel (with single-cycle data throughput) or word-serial (with multiple-cycle throughput).

Iterations

Specifies the number of internal add-sub iterations to perform.

Compensation

Specifies CORDIC magnitude scaling compensation. The outputs are compensated using a LUT-based multiplier or the block multiplier.

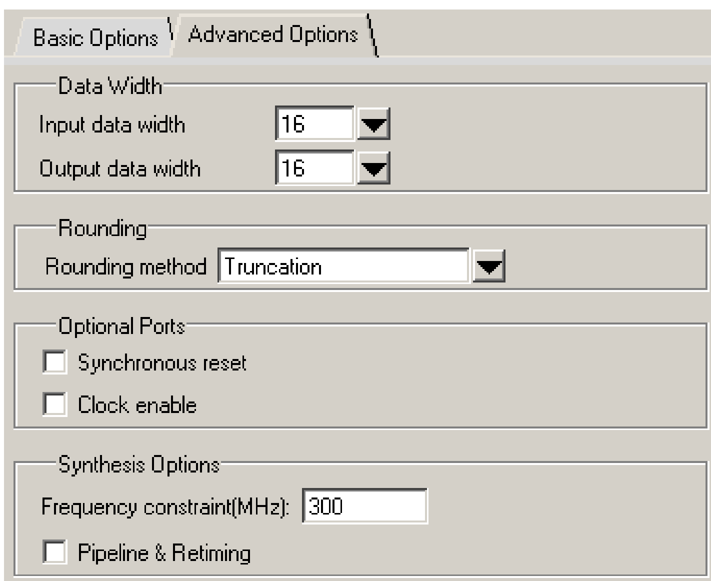
Pre-Rotation

Specifies whether the pre-rotation module is instantiated.

Advanced Options Tab

Figure 3-2 shows the CORDIC Advanced Options tab in the IPexpress tool.

Figure 3-2. CORDIC Advanced Options Tab



Data Width

Consists of two dropdown menus: Input Data Width and Output Data Width.

Rounding

Identifies the rounding method to be used when it is necessary to drop one or more LSBs from the true output.

Optional Ports

Synchronous Reset
Specifies whether a synchronous reset port is needed. A synchronous reset signal resets all the registers in the IP core.

Clock Enable

Specifies whether a clock enable port is needed in the IP. Clock enable control can be used for power saving when the core is not used. Use of clock enable port increases the resource utilization and may affect performance due to increased routing congestion.

Synthesis Options

Frequency Constraint (MHz)

Specifies frequency constraint for synthesis and PAR. The value specified here will be included in the .lpf file with an additional 50MHz overconstraining adjustment factor (overconstraining typically provides improved performance). For example, if this value is 250, the frequency constraint in the .lpf file will be “250MHz PAR_ADJ 50”.

Pipelining and Retiming

Specifies pipelining and retiming synthesis options for Synplify Pro. This option is not recommended to be selected.

IP Core Generation

This chapter provides information on how to generate the CORDIC IP core using the Diamond or ispLEVER software IPexpress tool, and how to include the core in a top-level design.

Licensing the IP Core

An IP core- and device-specific license is required to enable full, unrestricted use of the CORDIC IP core in a complete, top-level design. Instructions on how to obtain licenses for Lattice IP cores are given at:

http://www.latticesemi.com/products/intellectualproperty/aboutip/isplevercoreonlinepurchas.cfm

Users may download and generate the CORDIC IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The CORDIC IP core also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license. See "Hardware Evaluation" on page 28 for further details. However, a license is required to enable timing simulation, to open the design in the Diamond or ispLEVER EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

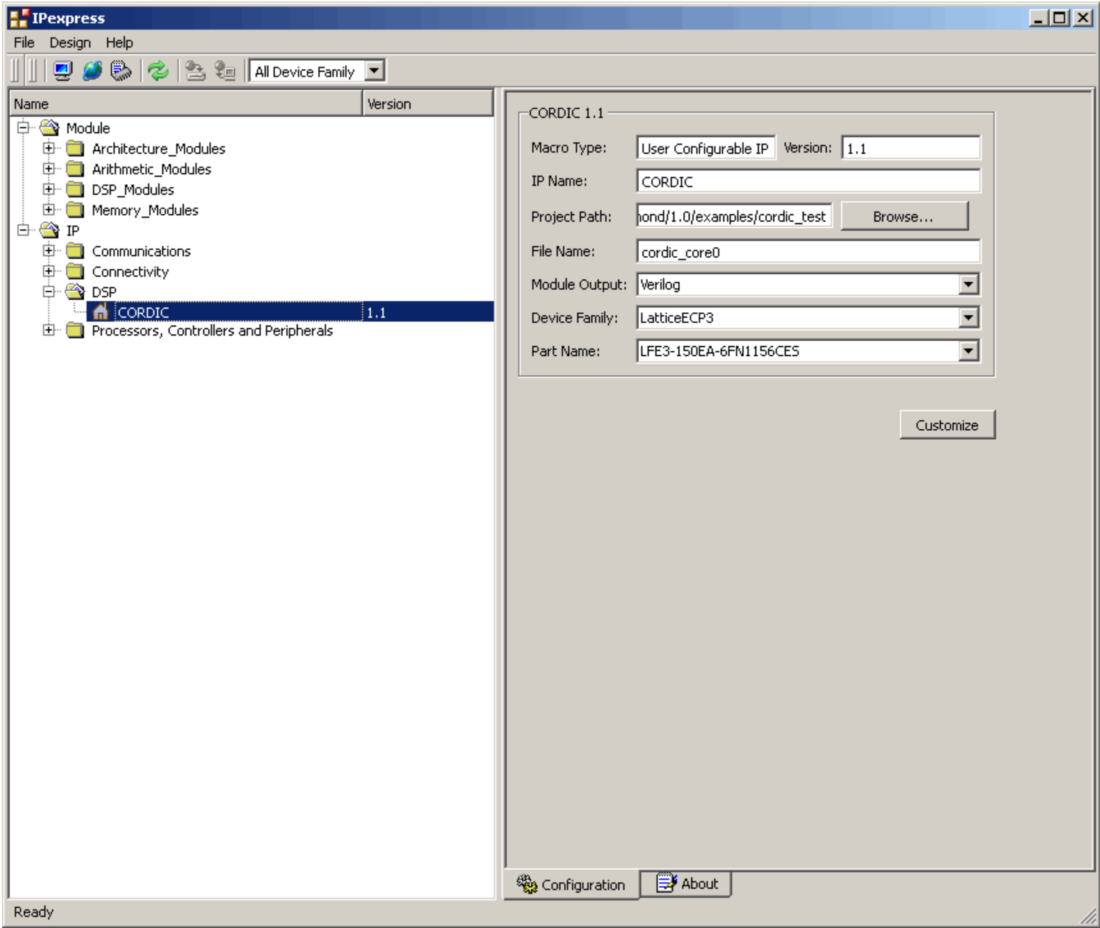
Getting Started

The CORDIC IP core is available for download from the Lattice IP Server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1.

The IPexpress tool GUI dialog box for the CORDIC IP core is shown in Figure 4-1. To generate a specific IP core configuration the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be located.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **(Diamond) Module Output** – Verilog or VHDL.
- **(ispLEVER) Design Entry Type** – Verilog HDL or VHDL.
- **Device Family** – Device family to which IP is to be targeted (e.g. LatticeSCM, Lattice ECP2M, LatticeECP3, etc.). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

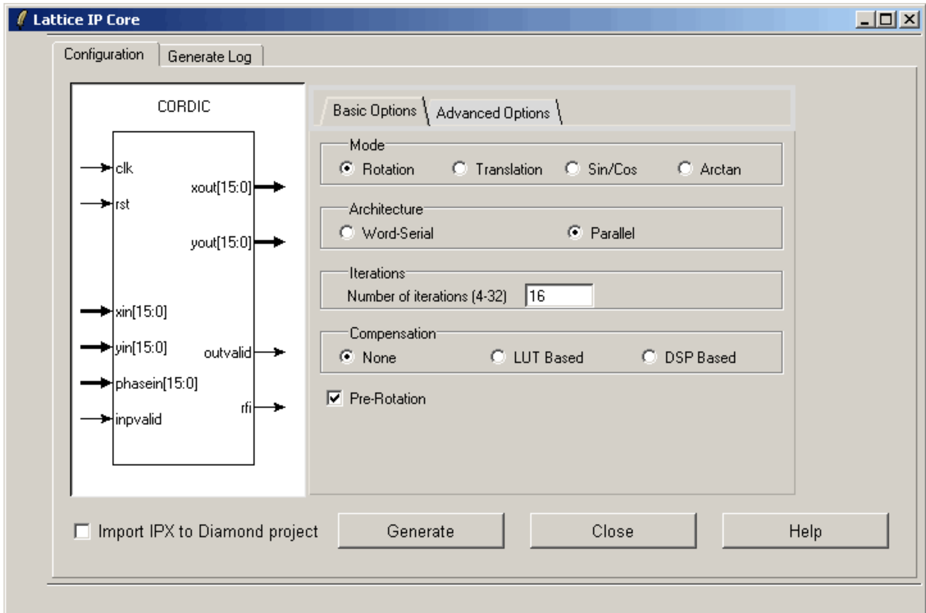
Figure 4-1. Pexpress Dialog Box (Diamond Version)



Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output (Design Entry in ispLEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the CORDIC IP core Configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to "Parameter Settings" on page 20 for more information on the CORDIC IP core parameter settings.

Figure 4-2. Configuration GUI (Diamond Version)



IPexpress-Created Files and Top Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified "Project Path" directory. The directory structure of the generated files is shown in Figure 4-3.