Rate This Document

CoreCORDIC v3.0

Handbook

Actel®
POWER MATTERS

Table of Contents

Introduction

CoreCORDIC is an Actel FPGA–optimized CORDIC engine. The Coordinate Rotation Digital Computer (CORDIC) algorithm by J. Volder provides an iterative method of performing vector rotations using only shifts and adds. The top-level interface of data source and receiver with a CORDIC engine is shown in Figure 1.

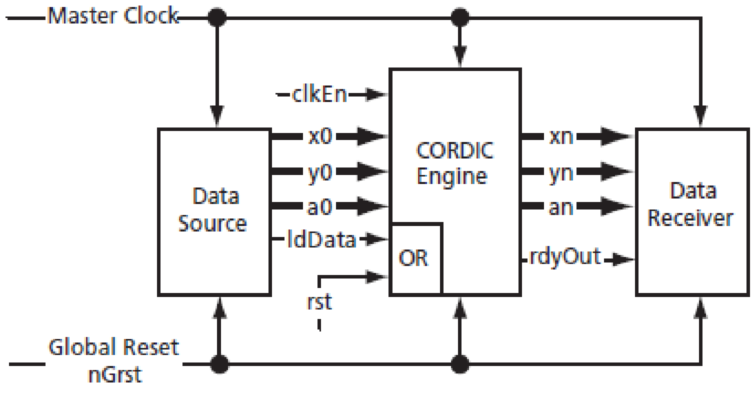
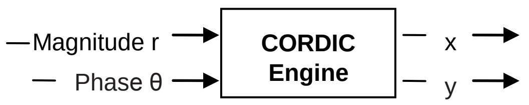


Figure 1 Top-Level Interface of Data source and receiver with CORDIC Engine

Depending on the configuration defined by you, the resulting module implements pipelined parallel, word-serial, or bit-serial architecture in one of two major modes: rotation or vectoring. In rotation mode, the CORDIC rotates a vector by a specified angle. This mode is used to convert polar coordinates to Cartesian coordinates, for general vector rotation, and also to calculate sine and cosine functions (see Figure 2).

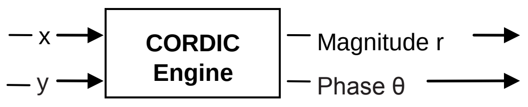


where $X = K \cdot r \cdot \cos x\theta$

$Y = K \cdot r \cdot \sin\theta$

Figure 2 Top CORDIC Engine in Rotation Mode

In vectoring mode, the CORDIC rotates the input vector towards the *x* axis while accumulating a rotation angle. Vectoring mode is used to convert Cartesian vector coordinates to polar coordinates; i.e., to calculate the magnitude and phase of the input vector (see Figure 3).



where $r = K \cdot \sqrt{(x^2 + y^2)}$

$\theta = \arctan(Y/X)$

Figure 3 CORDIC Engine in Vectoring Mode

The CORDIC results, such as *x, y,* and *r,* are scaled by the inherent processing gain, *K,* which depends on the number of iterations and converges to about 1.647 after a few iterations. The gain can be compensated for in many applications when the system includes the CORDIC engine. To assist a user in doing so, the CoreCORDIC generator computes the precise value of the gain and displays it on a screen. In the cases when only relative magnitude is of importance—for example, spectrum analysis and AM demodulation—the constant gain can be neglected. When calculating sine/cosine, the CORDIC gets initialized with a constant reciprocal value of the processing gain $r = 1/K$.

The input and output data is represented as q-bit words, where ‘q’ is a user-defined number in the range from 8 to 48. The number of iterations is also defined by a user in the same range. The CORDIC result accuracy improves when the number of iterations is increased, as long as the number of iterations does not exceed data bit width. In other words, the bit width limits the number of meaningful iterations.

Key Features

- Vector rotation – Conversion of polar coordinates to rectangular coordinates
- Vector translation – Conversion of rectangular coordinates to polar coordinates
- Sine and cosine calculation
- Vector (X, Y) magnitude and phase (arctan[X/Y]) calculation
- 8-bit to 48-bit configurable word size
- 8 to 48 configurable number of iterations
- Parallel pipelined architecture for the fastest calculation
- Bit-serial architecture for the smallest area
- Word-serial architecture for moderate speed and area
- Word parallel data I/Os

Core Version

This handbook supports CoreCORDIC version 3.0.

Supported Actel FPGA Families

Families supported by this core are:

- IGLOO®
- IGLOOe
- IGLOO PLUS
- ProASIC®3
- ProASIC3E
- ProASIC3L
- SmartFusion®
- Fusion
- ProASIC$^{\text{PLUS}}$®
- Axcelerator®
- RTAX-S
- SX-A
- RTSX-S

Utilization and Performance

CoreCORDIC can be implemented in several Actel devices. A summary of CoreCORDIC utilization and performance for various devices is listed in Table 1. The CORDIC engine bit resolution is set to 24 bits and the number of iterations set to 24. Device utilization and performance varies depending on the architecture chosen and the configuration parameters used. Timing-driven settings were used when synthesizing parallel architectures; area optimization settings were used in other cases. CoreCORDIC does not utilize on-chip RAM blocks.

Table 1 CoreCORDIC Device Utilization and Performance

Device	Engine Architecture	Mode	Cell or Tiles			Utilization %	Clock Rate, MHz	Transform Time, nsec
			Comb	Seq	Total			
Fusion				**Speed Grade-2**				
AFS600	Bit-serial	Rotate	288	97	385	3%	94	6,151
		Vector	287	96	383	3%	105	5,509
AFS600	Word-serial	Rotate	1,247	87	1334	10%	63	397
		Vector	1,249	85	1334	10%	64	391
AFS600	Parallel	Rotate	11,432	1,808	13240	96%	73	14
ProASIC3/E				**Speed Grade-2**				
A3P250	Bit-serial	Rotate	286	96	382	6%	97	5,961
		Vector	283	96	379	6%	100	5,783
A3P250	Word-serial	Rotate	1,245	87	1332	22%	62	403
		Vector	1,266	87	1,353	22%	64	391
A3P1000	Parallel	Rotate	9,372	1,797	11,169	45%	76	13
		Vector	9,645	1,824	11,469	47%	70	14
IGLOO/e				**Speed Grade-STD**				
AGL600V2	Bit-serial	Rotate	313	99	412	3%	28	20,594
		Vector	322	100	422	3%	29	19,885
AGL600V2	Word-serial	Rotate	1,407	83	1,490	11%	18	1,389
		Vector	1,476	84	1,560	11%	17	1,471
AGL1000V2	Parallel	Rotate	12,711	1,817	14,528	59%	18	56
		Vector	15,714	1,849	17,563	71%	19	53
ProASIC^{PLUS}				**Speed Grade-STD**				
APA150	Bit-serial	Rotate	391	105	496	8%	29	19,885
		Vector	392	104	496	8%	38	15,181
APA150	Word-serial	Rotate	1,359	104	1,463	24%	31	806
		Vector	1,386	106	1,492	24%	30	833
APA600	Parallel	Rotate	14,430	1,947	16,386	76%	33	30
		Vector	18,312	1,995	20,307	94%	29	34

Axcelerator			Speed Grade-2					
AX125	Bit-serial	Rotate	178	105	283	14%	132	4,387
		Vector	179	103	282	14%	132	4,387
AX125	Word-serial	Rotate	401	128	529	26%	111	225
		Vector	395	125	520	26%	107	234
AX1000	Parallel	Rotate	4,221	1,788	6,009	33%	130	8
		Vector	4,126	1,817	5,943	33%	128	8
RTAX-S			**Speed Grade-1**					
RTAX250S	Bit-serial	Rotate	177	105	282	7%	94	6,151
		Vector	177	105	282	7%	94	6,151
RTAX250S	Word-serial	Rotate	401	126	527	13%	80	313
		Vector	390	126	516	12%	74	338
RTAX1000S	Parallel	Rotate	4221	1788	6009	33%	97	10
		Vector	4126	1817	5943	33%	97	10
SX-A			**Speed Grade-2**					
A54SX32A	Bit-serial	Rotate	197	107	304	11%	86	6,721
		Vector	194	105	299	10%	87	6,644
A54SX32A	Word-serial	Rotate	666	131	797	27%	65	385
		Vector	697	120	817	28%	61	410
RTSX-S			**Speed Grade-1**					
RT54SX32	Bit-serial	Rotate	191	107	298	10%	26	2,2177
		Vector	191	106	297	10%	27	2,1356
RT54SX32	Word-serial	Rotate	690	138	828	29%	17	1,471
		Vector	697	136	833	29%	16	1,563
SmartFusion			**Speed Grade-1**					
A2F200M3F	Bit-serial	Rotate	299	202	501	11%	94	6,128
		Vector	299	202	501	11%	100	5,760
A2F200M3F	Word-serial	Rotate	1312	83	1395	31%	59	424
		Vector	1218	83	1301	28%	62	403

Note: The above data were obtained by typical synthesis and place-and-route methods. Other core parameter settings can result in different utilization and performance values.

Design Description

Verilog/VHDL Parameters

CoreCORDIC generates the CORDIC engine RTL code based on parameters set by the user when generating the module. The core generator supports the variations specified in Table 2.

Table 2 CoreCORDIC Generator Parameters

Parameter Name	Valid Range	Default	Description
FAMILY	0 to 99	11	Must be set to match the supported FPGA family. 9: RTSX-S 11: Axcelerator 12: RTAX-S 14: ProASICPLUS 15: ProASIC3 16: ProASIC3E 17: Fusion 18: SmartFusion 20: IGLOO 21: IGLOOe 22: ProASIC3L 23: IGLOO+
ARCHITECTURE	0 to 2	0	0: Bit-serial 1: Word-serial 2: Word parallel architecture
MODE	0,1	0	0: Vector rotation (polar to rectangular coordinate conversion and sine/cosine calculation) 1: Vector translation (rectangular to polar conversion)
BIT_WIDTH	8 to 48	48	I/O data bit width
ITERATIONS	8 to BIT_WIDTH	48	Shows the number of iterations. This value must be less than or equal to BIT_WIDTH.

I/O Signals

The data input is composed of a combination of X0, Y0, and A0. The type of data inputs rely on the parameters selected as explained in Table 3. The same goes for the output where, depending on your inputs and parameters, a variation of outputs could be produced.

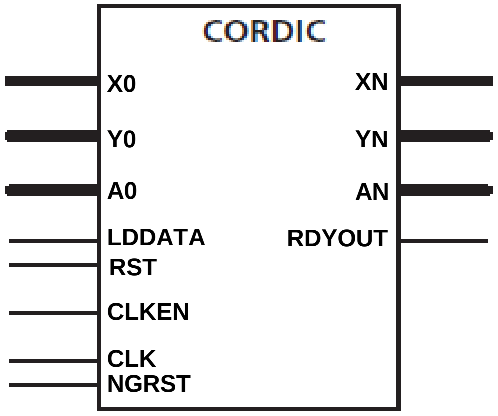


Figure 4 CoreCORDIC I/O Block Diagram

Table 3 CoreCORDIC I/O Signals

Port Name	Type	Description
X0	In	• Input data bus X0 • The abscissa of the input vector in the vectoring mode or the magnitude of the input vector in rotation mode should be placed on this bus • Bit [BIT_WIDTH – 1] is the Most Significant Bit (MSB) • Data is assumed to be presented in two’s complement format • The other vector coordinates must be supplied simultaneously • The value of X0 must always be positive
Y0	In	• Input data bus Y0 • The ordinate of the input vector in the vectoring mode should be placed on this bus • In rotation mode, the bus should be grounded or left idle • Bit [BIT_WIDTH – 1] is the MSB • Data is assumed to be presented in two’s complement format • The other vector coordinates must be supplied simultaneously
A0	In	• Input angle data bus A0 • The phase of the input vector in the rotation mode should be placed on this bus • In vectoring mode, the bus should be grounded or left idle • Bit [BIT_WIDTH – 1] is the MSB • Data is assumed to be presented in two’s complement format • The other vector coordinates must be supplied simultaneously.

LDDATA	In	• Load input data • Indicates that input vector coordinates are ready for the CORDIC engine to be processed • Active high • Valid in word-serial and bit-serial architectures
RST	In	• System/module synchronous reset • Active high • Valid in Word-serial and parallel architecture only • Resets all registers of the core
CLKEN	In	• Clock enable signal • Active high. Valid in word-serial and bit-serial architectures
CLK	In	• System clock. Active rising edge
NGRST	In	• System asynchronous reset. Active low
XN	Out	• Output data bus XN • The abscissa of the output vector in rotation mode or the magnitude of the output vector in the vectoring mode appears on this bus • Bit [BIT_WIDTH – 1] is the MSB • Data is presented in two's complement format • The other vector coordinates emerge on their respective output buses simultaneously
YN	Out	• Output data bus YN • The ordinate of the output vector in rotation mode • Bit [BIT_WIDTH– 1] is the MSB • Data is presented in two's complement format • The other vector coordinates emerge on their respective output buses simultaneously
AN	Out	• Output data bus AN • The phase of the output vector in vectoring mode • Bit [BIT_WIDTH – 1] is the MSB • Data is presented in two's complement format • The other vector coordinates emerge on their respective output buses simultaneously
RDYOUT	Out	• Output data (vector coordinates or sine/cosine values) are ready for the data receiver to read • Active high • Valid in word-serial and bit-serial architectures

Note: All signals are Active High (logic 1) unless otherwise noted.

Design Details

Architecture

Three different architectures are used that vary in speed and device utilization.

Word-Serial Architecture

Direct implementation of the CORDIC iterative equations yields the block diagram shown in Figure 5. The vector coordinates to be converted, or initial values, are loaded via multiplexers into registers RegX, RegY, and RegA. RegA, along with an adjacent adder/subtractor, multiplexer, and a small arctan look-up table (LUT), is often called an angle accumulator. Then on each of the following clock cycles, the registered values are passed through adders/subtractors and shifters. They are loaded back to the same registers. Each iteration takes one clock cycle, so that in *n* clock cycles, *n* iterations are performed and the converted coordinates are stored in the registers.

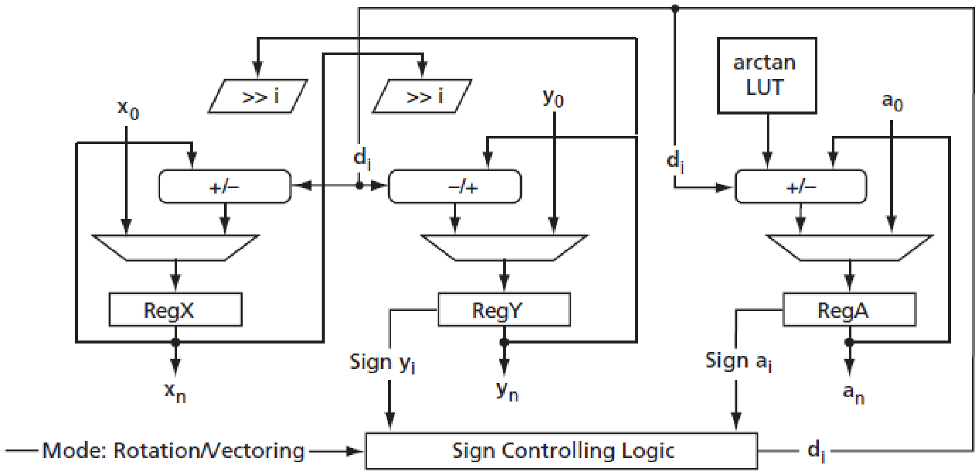


Figure 5 Word-Serial CORDIC Block Diagram

Depending on the CORDIC mode (rotation or vectoring), the sign-controlling logic watches either the RegY or the RegA sign bit. It determines what type of operation (addition or subtraction) needs to be performed at every iteration. The arctan LUT keeps a pre-computed table of the *arctan(2-i)* values. The number of entries in the arctan LUT equals the desirable number of iterations, *n*.

The word-serial CORDIC engine takes *n + 1* clock cycles to complete a single vector coordinate conversion.

Parallel Pipelined Architecture

This architecture presents an unrolled version of the sequential CORDIC algorithm above. Instead of reusing the same hardware for all iteration stages, the parallel architecture has a separate hardware processor for every CORDIC iteration. An example of the parallel CORDIC architecture configured for rotation mode is shown in Figure 6 on page 14.

Each of the *n* processors performs a specific iteration, and a particular processor always performs the same iteration. This leads to a simplification of the hardware. All the shifters perform the fixed shift, which means these can be implemented in the FPGA wiring. Every processor utilizes a particular arctan value that can also be hardwired to the input of every angle accumulator. Yet another simplification is an absence of a state machine.

The parallel architecture is obviously faster than the sequential architecture described in the Word-Serial Architecture. It accepts new input data and puts out the results at every clock cycle. The architecture introduces a latency of *n* clock cycles. Actel recommends sending a test signal to determine the latency and use that value to match your input data to the outputs.

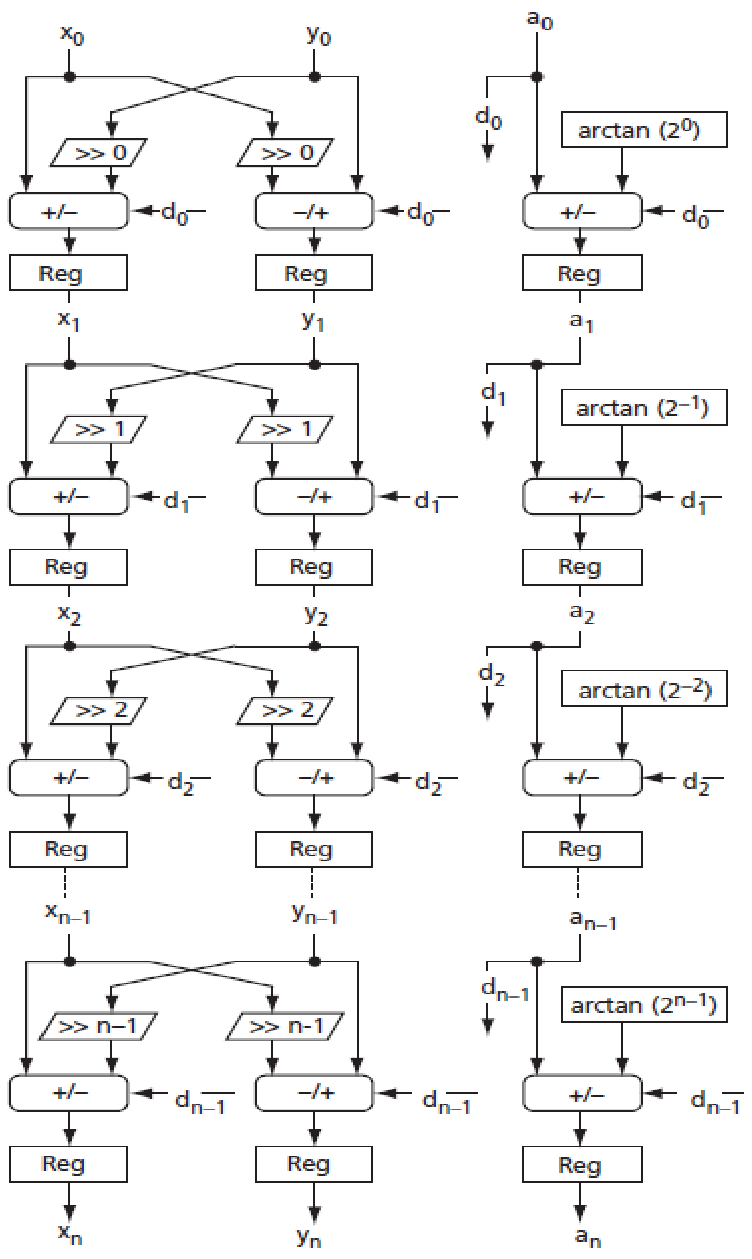


Figure 6 Parallel CORDIC Architecture

Bit-Serial Architecture

Whenever the CORDIC conversion speed is not an issue, this architecture provides the smallest FPGA implementation. For example, in order to initialize a sine/cosine LUT, the bit-serial CORDIC is the solution. Figure 7 depicts the simplified block diagram of the bit-serial architecture. The shift registers get loaded with initial data presented in bit-parallel form; that is, all bits at once. The data then shifts to the right, before arriving to the serial adders/subtractors. Every iteration takes *m* clock cycles, where *m* is the CORDIC bit resolution. Serial shifters are implemented by properly tapping the bits of the shift registers. The control circuitry (not shown in Figure 7) provides sign-padding of the shifted serial data to realize its correct sign extension. The results from the serial adders return back to the shift registers, so that in *m* clock cycles the results of another iteration are stored in the shift registers.

A single full CORDIC conversion takes *n×m+2* clock cycles, where *n* is the number of iterations.

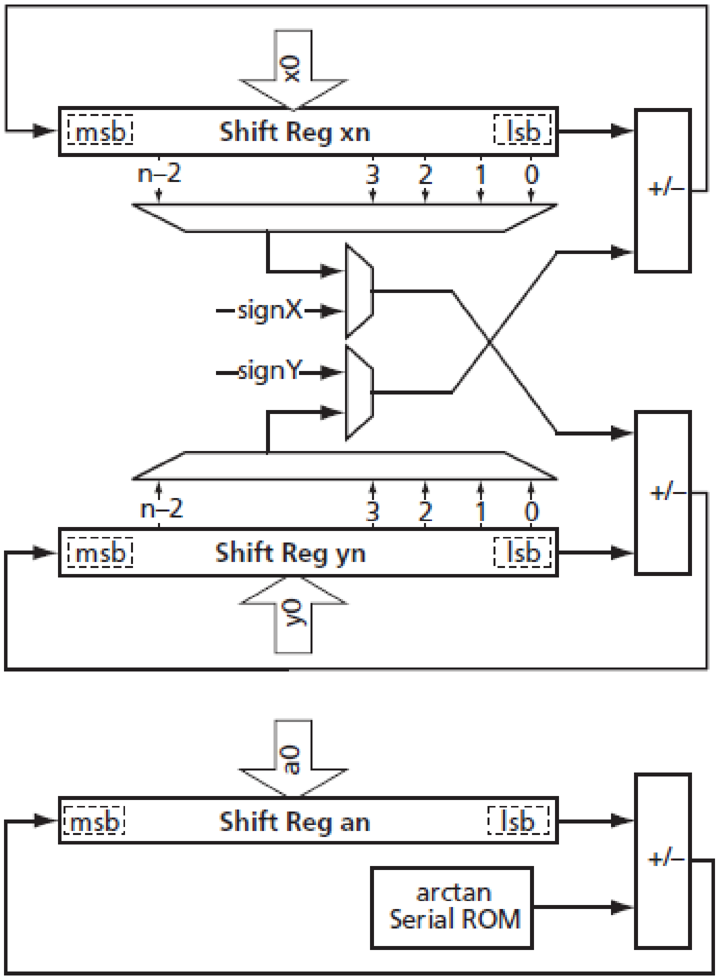


Figure 7 Bit-Serial CORDIC Architecture

I/O Formats – Q Format Fixed-Point Numbers

CoreCORDIC utilizes fixed-point arithmetic. In particular, the numbers the core operates with are presented as two's complement signed fractional numbers. To identify the position of a binary point separating the integer and fractional portions of the number, the Q format is commonly used. An mQn format number is an (n+1)-bit signed two's complement fixed-point number: a sign bit followed by n significant bits with the binary point placed immediately to the right of the m most significant bits. The m MSBs represent the integer part, and (n–m) LSBs represent the fractional part of the number, called the mantissa. 0 depicts an example of a 1Qn format number.

Table 4 1Qn Format Number

Bit 2^n	Bits 2^{n-1}	Position of the Binary Point	Bits $[2^{n-2} : 2^0]$
Sign	Integer bit		Mantissa

Table 5 Qn Format Number

Bit 2^n	Position of the Binary Point	Bits $[2^{n-1} : 2^0]$
Sign		Mantissa

The following sections explain in detail the formats of the input and output signals. The linear signals include Cartesian coordinates and a vector magnitude. These come to the CORDIC engine inputs *X0* and *Y0*, or appear on its outputs *XN* and *YN*. The angular signals include the vector phase that comes to the CORDIC engine input *A0*, or appears on its output *AN*. Both linear and angular signals utilize mQn formats and appropriate conversion rules from floating-point to the mQn formats.

I/O Linear Format

The CoreCORDIC engine utilizes the 1Qn format shown in 0. Though the 1Qn format numbers are capable of expressing fixed-point numbers in the range from (–2n) to (2n – 2m–n), the input linear data must be limited to fit the smaller range from (–2n–1) to (2n–1). In terms of floating-point numbers, the input must fit the range from –1.0 to +1.0. For example, the 1Q9 format input data range is limited by the following 10-bit numbers:

Max input negative number of –1.0:

1100000000 ⇔ 11.00000000

Max input positive number of +1.0:

0100000000 ⇔ 01.00000000

This precaution is taken to prevent the data overflow that otherwise could occur as a result of the CORDIC inherent processing gain. Also, in vector mode, the square root of $x^2 + y^2$ multiplied by gain must be less than 2 ($\mathrm{sqrt}(x^2+y^2)$*gain < 2), or else overflow occurs as well. To convert floating-point linear input data to the 1Qn format, follow the simple rule in Equation 1.

Equation 1: 1Qn Fixed-Point Data = $2^{n-1} \times \text{FloatingPoint Data}$

Here it is assumed the floating-point Data is presented in the range from –1.0 to 1.0. The product on the right-hand side of Equation 1 contains integer and fractional parts. The fractional part has to be truncated or rounded. Table 6 shows a few examples of converting the floating-point numbers to the 1Q9 format. To convert the 1Qn format back to the floating-point format, use Equation 2.

Equation 2: Floating-Point Data = $1\text{Qn FixedPoint Data}/2^{n-1}$

Table 6 Floating-Point to 1Q9 Format Conversion

Floating-Point Number X	P = X × $2^{(n–1)}$	P Rounded	Common Binary Format	1Q9 Format
1.00	256	256	0100000000	01.00000000
0.678915	173.80224	174	0010101101	00.10101101
0.047216	12.087296	12	0000001100	00.00001100
–1.00	–256	–256	1100000000	11.00000000
-0.678915	-173.80224	-174	1101010011	11.01010011
-0.047216	-12.087296	-12	1111110100	11.11110100

I/O Angular Format

The angle (phase) signals are *A0* and *AN* (seen in Figure 4). They are presented in Qn format, as shown in Table 5. The relation between the floating-point angular value expressed in radians and the Qn format is shown in Equation 3.

Equation 3: Qn Fixed-Point Angle = $2^n \times \mathrm{Floating} - \mathrm{Point\ Angle}/\pi$

In Equation 3, the floating-point angle is measured in radians. The product on the right-hand side of Equation 1 contains integer and fractional parts. The fractional part must be truncated or rounded. Equation 4 presents a rule for the conversion from the Qn format back to the floating-point radian measure.

Equation 4: Floating-Point Angle = $\mathrm{Qn\ FixedPoint\ Angle} \times \pi/2^n$

The conversion formulae (Equation 3 and Equation 4) support an important feature that greatly simplifies sine and cosine table calculations. Such tables usually have power of two entries (lines). At the same time, they often span angular values from –π/2 to π/2 radians. Therefore, it is beneficial to represent the angle of π/2 radians with the power of two fixed-point numbers. In particular, when having the CORDIC engine calculate the sin(θ) and cos(θ) table, it is sufficient to increment the fixed-point angular argument θ at each cycle. The angular value range is from –π/2 to π/2, or in Q9 format:

Max input negative number of –π/2:

1100000000 ⇔.1100000000

Max input positive number of +π/2:

0100000000 ⇔ .0100000000

0 shows a few examples of converting floating-point numbers to Q9 format.

Table 7 Examples of Angular Value to Fixed-Point Conversion

Floating-Point Angle A (rad)		P = A × 2^n	Common Binary Format	Q9 Format (sign.mantissa)
π/2	1.5707963268	256	0100000000	0.100000000
π/4	0.7853981634	128	0010000000	0.010000000
π/256	0.0122718463	2	0000000010	0.000000010
-π/2	-1.5707963268	-256	1100000000	1.100000000
-π/4	-0.7853981634	-128	1110000000	1.110000000
-π/256	-0.0122718463	-2	1111111110	1.111111110

Interface Definitions

Upon reset, the CORDIC core returns to its initial state. Signal NGRST asynchronously resets any architecture. Other I/O interfaces and timing depend on core architecture.

Bit-Serial Architecture Interface and Timing

Figure 8 depicts a typical timing diagram for the bit-serial architecture. Signal LDDATA resets the bit-serial CORDIC module and loads a set of data present on the A0, X0, and Y0 input busses. The set of input data is shown in Figure 8 as In0. Normally, the next LDDATA signal has to come after the end of a current CORDIC cycle; that is, after the RDYOUT signal appears on the module output. In case the next LDDATA signal is issued prior to the end of the current cycle, the CORDIC engine starts a new computation cycle and discards the incomplete results of the interrupted cycle. Once the CORDIC engine completes calculating the result, it generates an RDYOUT signal one clock period in width. The result on the output busses (AN, XN, and YN) is valid while the RDYOUT signal is active. The next LDDATA signal can coincide with the RDYOUT signal. Obviously a valid, fresh set of input data, shown as In1 in Figure 8, must be ready by then. One cycle of CORDIC computation = (BIT_WIDTH × ITERATIONS + 2) clock cycles. Signal CLKEN can be manipulated as desired. While this signal is low, the CORDIC engine retains all the data it has collected or processed so far. Normally, the bit-serial CORDIC engine is used to fill up the LUT on a power-on event.

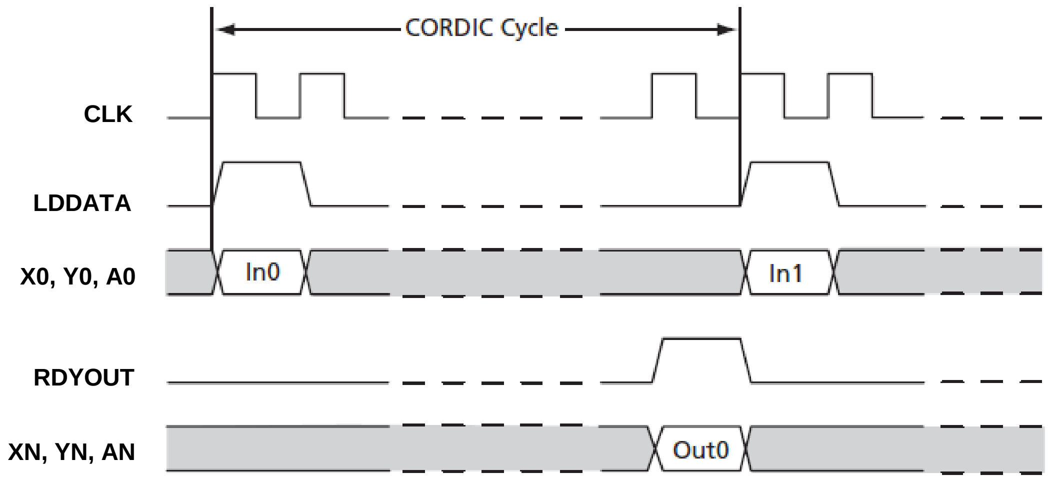


Figure 8 Bit-Serial Architecture Timing Diagram

Word-Serial Architecture Interface and Timing

Figure 9 depicts a timing diagram for the word-serial architecture. It is very similar to the bit-serial timing diagram. Signal LDDATA resets the word-serial CORDIC module and loads the set of data present on the A0, X0, and Y0 input busses. The set of input data is shown in Figure 9 as In0. Normally the next LDDATA signal must come after the end of the current CORDIC cycle; that is, after the RDYOUT signal appears on the module output. In the case that the next LDDATA signal is issued prior to the end of a current cycle, the CORDIC engine starts a new computation cycle and discards the incomplete results of the interrupted cycle.

Once the CORDIC engine completes calculating the result, it generates an RDYOUT signal one clock period in width. The result on the output busses (AN, XN, and YN) is valid while the RDYOUT signal is active. The next LDDATA signal can immediately follow the RDYOUT signal. A valid, fresh set of input data, shown as In1, must be ready by then.

One cycle of CORDIC computation = (iterations + 1) clock cycles.

Signal CLKEN can be manipulated as desired. While this signal is low, the CORDIC engine retains all the data it has collected or processed so far. As an example, the word-serial CORDIC engine is used to fill up the LUT on a power-on event. Once the CORDIC completes the task, a high-level state machine may disable the CLKEN signal.

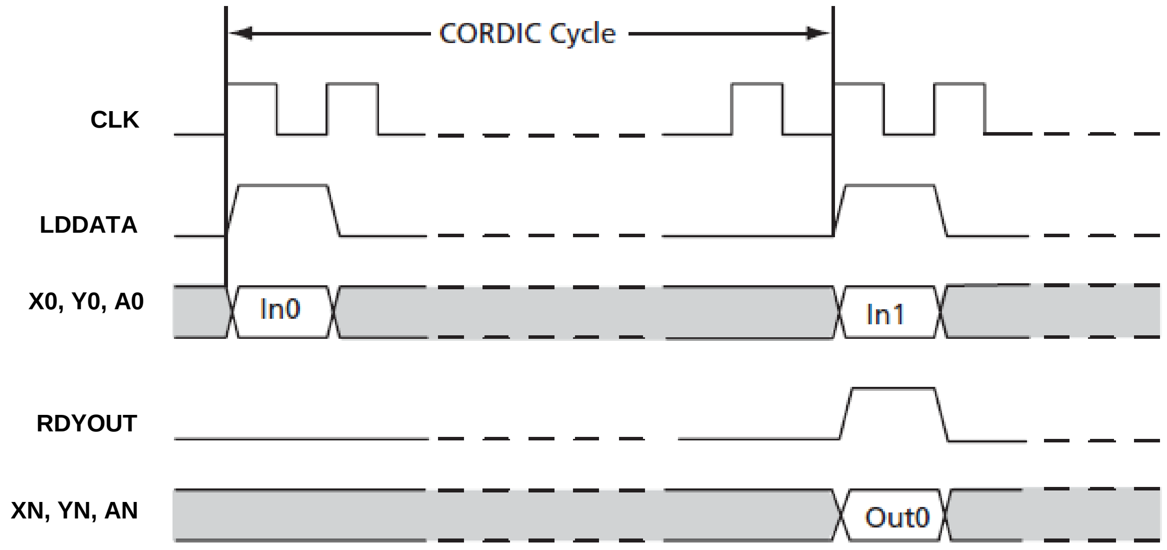


Figure 9 Word-Serial Architecture Timing Diagram

Parallel Architecture Interface and Timing

Figure 10 depicts a timing diagram for the parallel architecture. At the beginning of every clock cycle, a fresh set of input arguments A0, X0, and Y0 enters the CORDIC engine. No control signals accompany the input data. The CORDIC engine puts out the results at the beginning of every clock cycle with the latency of ITERATIONS clock cycles. Signal RST synchronously resets the parallel architecture (registers of the parallel engine).

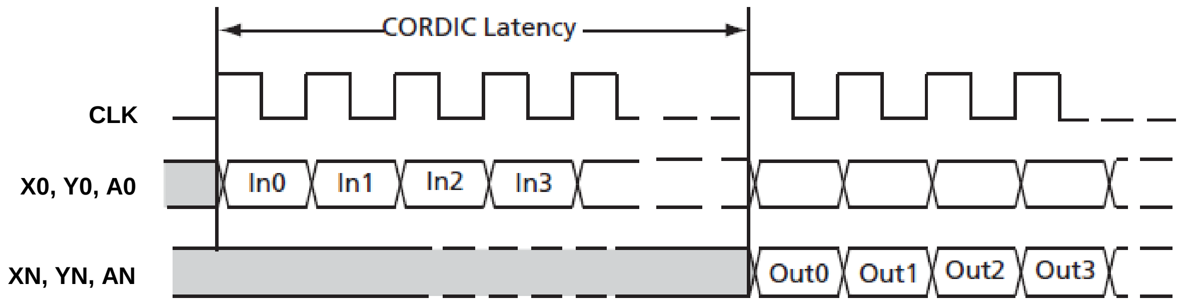


Figure 10 Parallel Architecture Timing Diagram

Tool Flows

Licensing

CoreCORDIC is only licensed to allow generating complete RTL source code. In this version the Generator allows generation, simulation, synthesis, and place-and-route of a user-defined configuration.

SmartDesign

CoreCORDIC is preinstalled in the SmartDesign IP Deployment design environment. For information on using SmartDesign to instantiate and generate cores, refer to the Using DirectCore in Libero® IDE User's Guide.

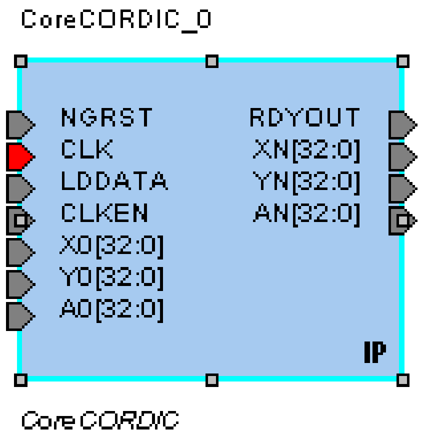
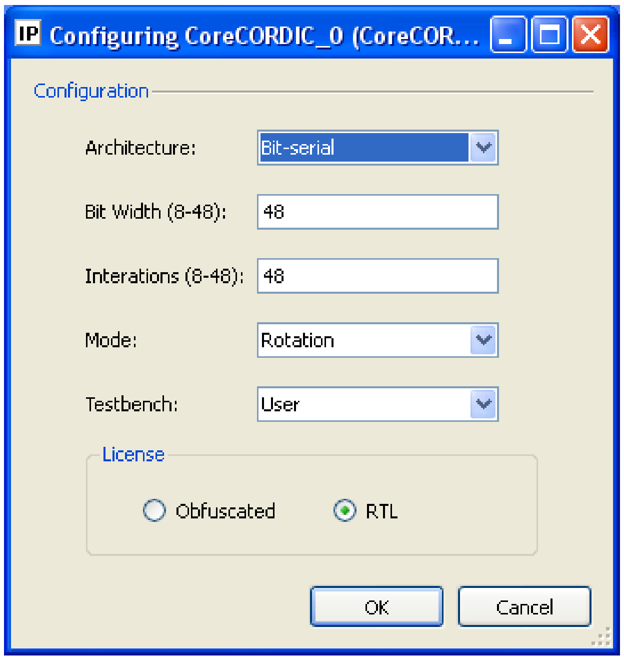


Figure 11 CoreCORDIC Full I/O View (Bit-Serial example)



Architecture: Bit-serial

Bit Width (8-48): 48

Interations (8-48): 48

Mode: Rotation

Testbench: User

License: Obfuscated [✓ RTL]

Figure 12 CoreCORDIC SmartDesign Configuration with Callouts to Associated Parameters

Once the design is generated an executable is run. This can be done automatically by running the simulation in Modelsim or through manual execution. Within the project directory (typically C:\Actelprj\<*project_name*>) the executable and configuration file can be found through the extension \component\Actel\DirectCore\CoreCORDIC\<*version_number*>\<*OS*> (OS is either 'win' for Windows or 'lin' for Linux).

Run the executable by typing the following command:

For Windows in a Command Prompt:

cd C:\Actelprj\<*project_name*>\component\Actel\DirectCore\CORECORDIC\<*version_number*>\<*OS*>

coreCORDIC.exe

For linux in a Terminal:

cd <*project_name*>/component/Actel/DirectCore/CORECORDIC/<*version_number*>/<*OS*>

./corecordic **

**(the permissions on your computer may not allow you to execute this program so change the permissions by typing "chmod +x corecordic" and run the previous command again)

Simulation Flows

The user testbench for CoreCORDIC is included in the release.

To run simulations, select the user testbench flow within SmartDesign and click **Save & Generate** on the Generate pane. The user testbench is selected through the Core Testbench Configuration GUI.

When SmartDesign generates the Libero IDE project, it installs the user testbench files.

To run the user testbench, set the design root to the CORECORDIC instantiation in the Libero IDE Design Hierarchy pane and click the Simulation icon in the Libero IDE Design Flow window. This invokes ModelSim® and automatically runs the simulation.

User Testbench

CoreCORDIC automatically generates two testbenches used for pre-synthesis simulations and post-synthesis simulations.

The first testbench implements a sine/cosine table test. It generates sixteen angle values (A0) in a range from –π/2 rad to π/2 rad, and the golden values of sin(A0) and cos(A0). The testbench applies the angle values one by one to the CORDIC engine input and compares the actual results generated by the CORDIC engine with the golden sine and cosine values. The testbench is used to verify the CORDIC engine in rotation mode. Figure 13 depicts functional block diagrams of the testbench along with the CORDIC engine.

The other testbench runs the CORDIC engine in vectoring mode. It generates a set of sixteen vectors (X0, Y0), as well as the golden set of their polar coordinates, magnitude, and phase (R,AN). The testbench applies the Cartesian values of (X0, Y0) to the CORDIC engine inputs and compares the actual results that the CORDIC engine generates to the golden values of the magnitude and phase. Figure 14 on page 24 shows a block diagram of the testbench.

Both testbenches print the input test vector values and the actual output values in floating point format.

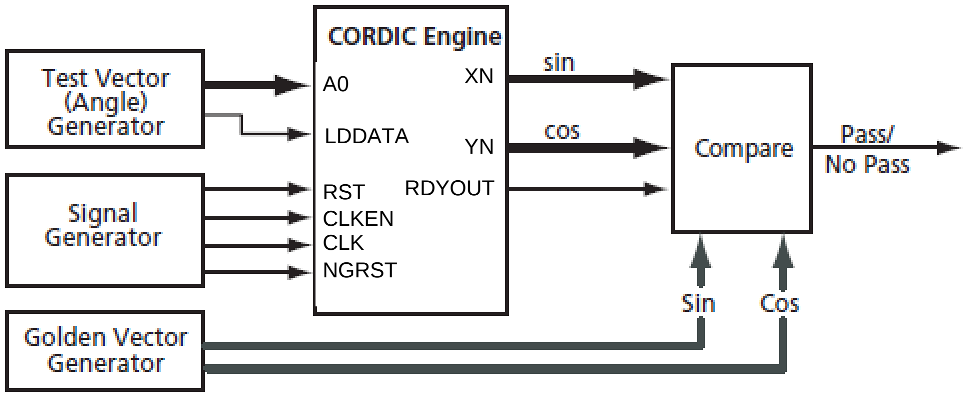


Figure 13 Sin/Cos Table Testbench Functional Block Diagram

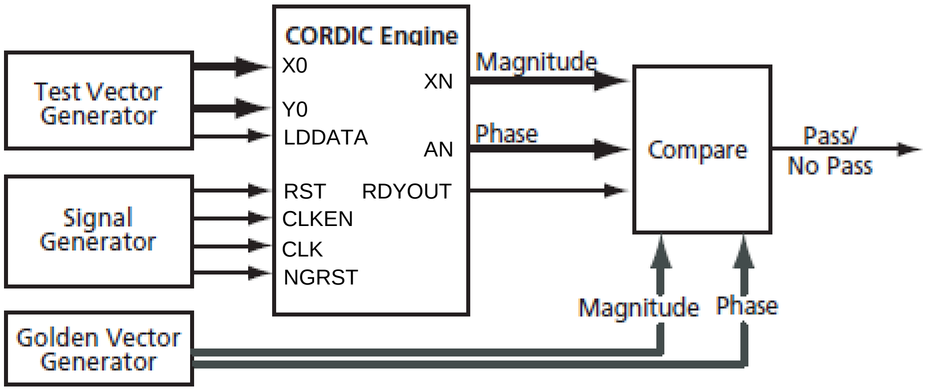


Figure 14 Magnitude/Phase Testbench Functional Block Diagram

Synthesis in Libero IDE

To run Synthesis on the core with parameters set in SmartDesign, set the design root appropriately and click the **Synthesis** icon in Libero IDE. The Synthesis window appears, displaying the Synplicity® project. Set Synplicity to use the Verilog 2001 standard if Verilog is being used. To run Synthesis, select the **Run** icon.

Place-and-Route in Libero IDE

Having set the design route appropriately and run Synthesis, click the Layout icon in the Libero IDE to invoke Designer. CoreCORDIC requires no special place-and-route settings.

Ordering Information

Ordering Codes

CoreCORDIC can be ordered through your local Actel sales representative. It should be ordered using the following number scheme: CoreCORDIC-XX, where XX is listed in Table 8.

Table 8 Ordering Codes

XX	**Description**
RM	RTL for RTL source — multiple-use license

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650. 318.8044**

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Actel Technical Support

Visit the Actel Customer Support website (http://www.actel.com/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at http://www.actel.com/.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/company/contact/default.aspx.

Actel Corporation *• 2061 Stierlin Court • Mountain View, CA 94043 • USA*
Phone 650.318.4200 • Fax 650.318.4600 • Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

Actel Europe Ltd*. • River Court, Meadows Business Park • Station Approach, Blackwater • Camberley Surrey GU17 9AB • United Kingdom*
Phone +44 (0) 1276 609 300 • Fax +44 (0) 1276 607 540

Actel Japan *• EXOS Ebisu Building 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • http://jp.actel.com

Actel Hong Kong *• Room 2107, China Resources Building • 26 Harbour Road • Wanchai • Hong Kong*
Phone +852 2185 6460 • Fax +852 2185 6488 • www.actel.com.cn

51700064-1/05.10