

Summary - Polar to Rectangular CORDIC

| | |
|-------------------|--|
| Name | pr.cordic |
| Worker Type | Application |
| Version | v1.5 |
| Release Date | 4/2019 |
| Component Library | ocpi.assets.dsp_comps |
| Workers | pr_cordic.hdl |
| Tested Platforms | xsim, isim, modelsim, alst4, ml605, ZedBoard(PL), Matchstiq-Z1(PL) |

Functionality

The Polar to Rectangular CORDIC component functions to perform a polar to rectangular conversion using a Coordinate Rotation Digital Computer (CORDIC) algorithm.

Worker Implementation Details

pr_cordic.hdl

Traditionally, CORDIC algorithms are represented with three inputs, x_0 , y_0 , and z_0 and their respective three outputs x_N , y_N , and z_N . For this polar to rectangular algorithm, we drive the *magnitude* to x_0 and the *phase* to z_0 . This process is summarized below in Equation 1.

$$x_0 = R, y_0 = 0, z_0 = \theta \quad (1)$$

Both the *magnitude* and *phase* are normalized prior to the CORDIC processes. The *magnitude* is normalized from -1 to $+1$ and the *phase* is normalized from $-\pi$ to $+\pi$. Due to the iterative nature of CORDIC algorithms, there is a gain factor A_n that needs to be handled within the system. This system accounts for the gain in Equation 2 internally by scaling the *magnitude* by a Gain Correction Factor K_c as seen in Equation 3.

$$A_n = \prod_n \sqrt{1 + 2^{-2i}} \quad (2)$$

$$K_c = \frac{1}{A_n} = \prod_{i=0}^N \cos(\text{atan}(2^{-i})), n = STAGES - 1 \quad (3)$$

Additionally, the *phase* input is prescaled internally by S_c in Equation 4 to simplify the arithmetic. Once S_c is applied to θ the input range will have a maximum negative value corresponding to $-\pi$ and a maximum positive value corresponding to $+\pi - \delta$, where δ is defined in Equation 5.

$$S_c = 2^{(DATA.WIDTH-1)}/\pi \quad (4)$$

$$\delta = 2\pi/2^{(DATA.WIDTH)} \quad (5)$$

This worker implementation takes the inputs from Equation 1 and scales them by the gain adjustment and phase prescaled to yield Equation 6.

$$x_0 = K_c R, y_0 = 0, z_0 = S_c/\theta \quad (6)$$

Lastly, the expected outputs are found in Equation 7.

$$x_N = R\cos(\theta), y_N = R\sin(\theta), z_N = 0 \quad (7)$$

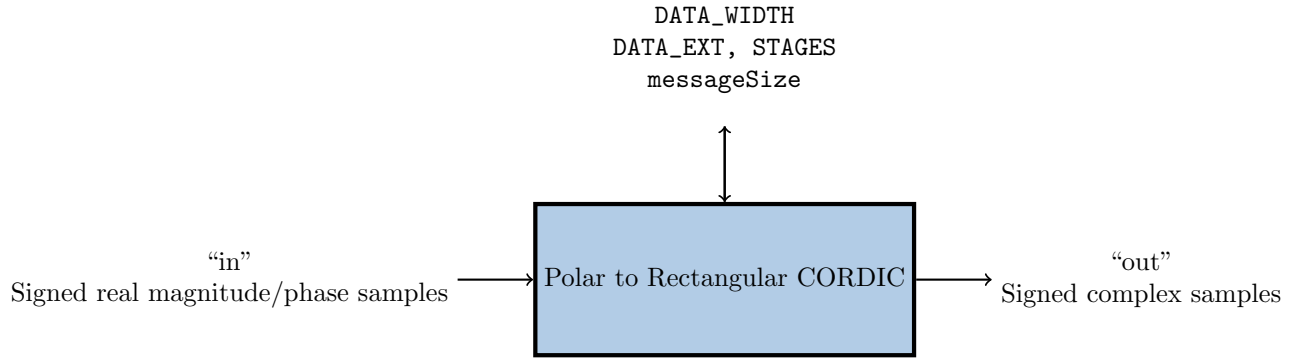
Theory

CORDIC algorithms are iterative algorithms used to calculate various trigonometric functions. This CORDIC algorithm specifically performs the conversion from the polar coordinate system to rectangular coordinate system. In polar coordinates, inputs to this component are the *radius* (or *magnitude*) and the *angle* θ , and expected outputs in the rectangular coordinate system are x and y . This process is summarized below in Equation 8.

$$R, \theta \rightarrow x, y \quad (8)$$

Block Diagrams

Top level



State Machine

Only one finite-state machine (FSM) is implemented by this worker. The FSM supports Zero-Length Messages.

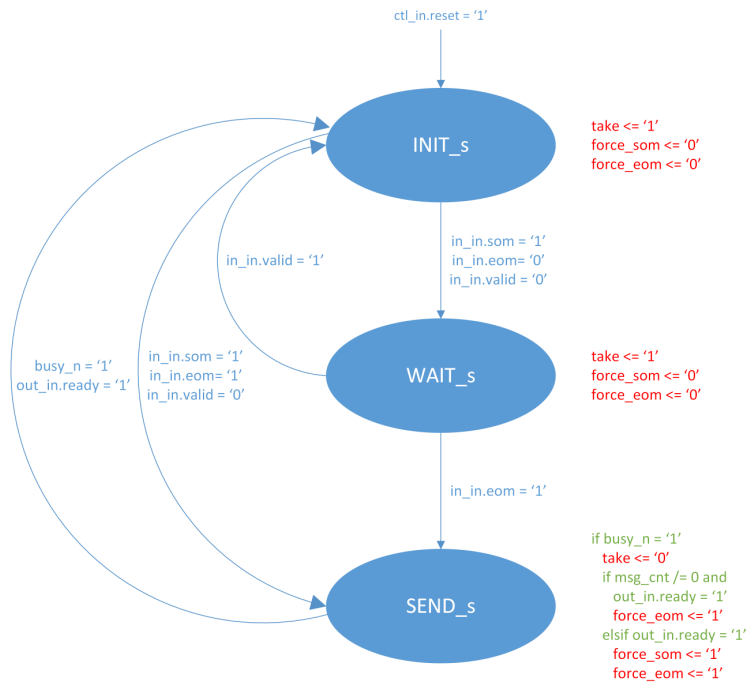


Figure 1: Zero-Length Message FSM

Note: In future releases, this worker will be converted to the HDL version 2 API which will remove the need for this state machine.

Source Dependencies

pr_cordic.hdl

- projects/assets/components/dsp_comps/pr_cordic.hdl/pr_cordic.vhd
- projects/assets/hdl/primitives/dsp_prims/dsp_prims_pkg.vhd
 - projects/assets/hdl/primitives/dsp_prims/cordic/src/cordic_pr.vhd
 - projects/assets/hdl/primitives/dsp_prims/cordic/src/cordic.vhd

projects/assets/hdl/primitives/dsp_prims/cordic/src/cordic.stage.vhd

Component Spec Properties

| Name | Type | SequenceLength | ArrayDimensions | Accessibility | Valid Range | Default | Usage |
|-------------|--------|----------------|-----------------|--------------------|-------------|---------|--|
| DATA_WIDTH | UChar | - | - | Readable | - | - | Real input and complex output data width |
| DATA_EXT | UChar | - | - | Readable | - | - | CORDIC requirement: # of extension bits |
| STAGES | UChar | - | - | Readable | - | - | Number of CORDIC stages implemented |
| messageSize | UShort | - | - | Writable, Readable | 8192 | 8192 | Number of bytes in output message |

Worker Properties

pr_cordic.hdl

| Type | Name | Type | SequenceLength | ArrayDimensions | Accessibility | Valid Range | Default | Usage |
|--------------|------------|------|----------------|-----------------|---------------|-------------|---------|--|
| SpecProperty | DATA_WIDTH | - | - | - | Parameter | 1-16 | 16 | Real input and complex output data width |
| SpecProperty | DATA_EXT | - | - | - | Parameter | 1-16 | 16 | CORDIC requirement: # of extension bits |
| SpecProperty | STAGES | - | - | - | Parameter | 16-32 | 16 | Number of CORDIC stages implemented |

Component Ports

| Name | Producer | Protocol | Optional | Advanced | Usage |
|------|----------|-------------------|----------|----------|------------------------|
| in | false | iqstream_protocol | false | - | Signed complex samples |
| out | true | iqstream_protocol | false | - | Signed complex samples |

Worker Interfaces

pr_cordic.hdl

| Type | Name | DataWidth | Advanced | Usage |
|-----------------|------|-----------|-------------------------|---|
| StreamInterface | in | 32 | ZeroLengthMessages=true | Signed magnitude samples (31:16), Signed phase samples (15:0) |
| StreamInterface | out | 32 | ZeroLengthMessages=true | Signed complex samples |

Control Timing and Signals

The CORDIC Polar-to-Rectangular HDL worker uses the clock from the Control Plane and standard Control Plane signals.

Worker Configuration Parameters

pr_cordic.hdl

Table 1: Table of Worker Configurations for worker: pr_cordic

| Configuration | DATA_EXT | DATA_WIDTH | STAGES | ocpi_endian | ocpi_debug |
|---------------|----------|------------|--------|-------------|------------|
| 0 | 6 | 16 | 18 | little | false |

Performance and Resource Utilization

pr_cordic.hdl

Table 2: Resource Utilization Table for worker "pr_cordic"

| Configuration | OCPI Target | Tool | Version | Device | Registers (Typ) | LUTs (Typ) | Fmax (MHz) (Typ) | Memory/Special Functions |
|---------------|-------------|---------|---------|------------------|-----------------|------------|------------------|--------------------------|
| 0 | stratix4 | Quartus | 17.1.0 | EP4SGX230KF40C2 | 1553 | 1995 | N/A | DSP18: 8 |
| 0 | zynq | Vivado | 2017.1 | xc7z020clg484-1 | 1518 | 3336 | N/A | DSP48E1: 2 |
| 0 | zynq_ise | ISE | 14.7 | 7z020clg484-1 | 1484 | 3369 | 162.946 | DSP48E1: 4 |
| 0 | virtex6 | ISE | 14.7 | 6vlx240tff1156-1 | 1484 | 3369 | 165.235 | DSP48E1: 4 |

Test and Verification

A single test case is implemented to validate the CORDIC Polar-to-Rectangular component. An input file is generated with a constant magnitude of 29760 in bits (31:16) (driven into CORDIC input x_0) and two cycles of ramp data centered about zero representing a constant phase in bits (15:0) (driven into CORDIC input z_0). The expected outputs are a cosine from I (the CORDIC x_N output) with a magnitude equal to that of x_0 and a sine from Q (the CORDIC y_N output) also with a magnitude equal to that of x_0 . Input magnitude and phase plots may be viewed in Figures 2 and 3 below.

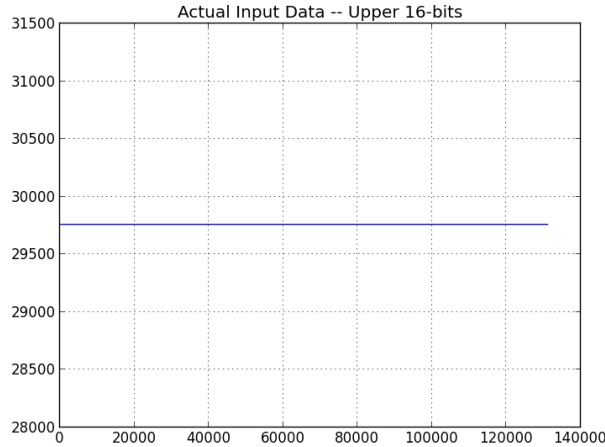


Figure 2: Time Domain Magnitude Input

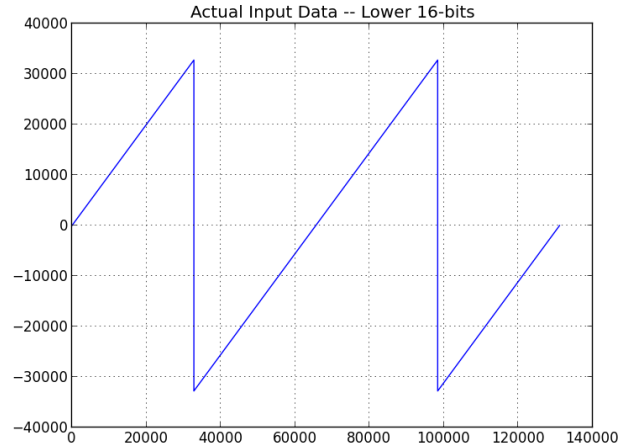


Figure 3: Time Domain Phase Input

Verification of output data is performed by creating two cycles of a complex sinusoid in python, and then comparing the worker output sample-by-sample with these expected results. A difference of ± 1.0 is tolerated to allow for differences in fixed-point implementation and hardware rounding. Figures 4 and 5 depict the output of the Polar to Rectangular CORDIC.

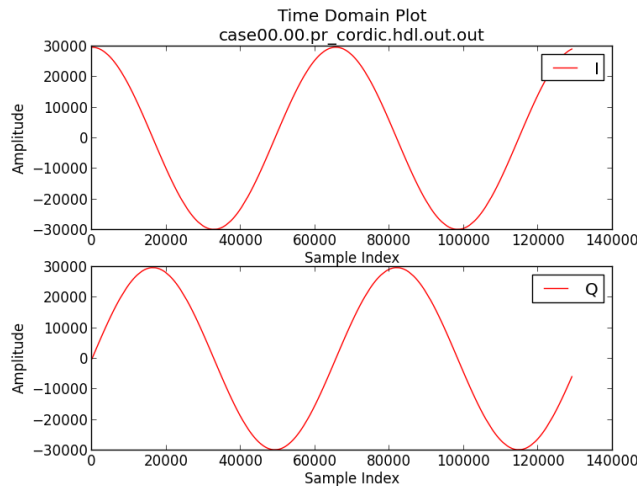


Figure 4: Time Domain Rectangular Output

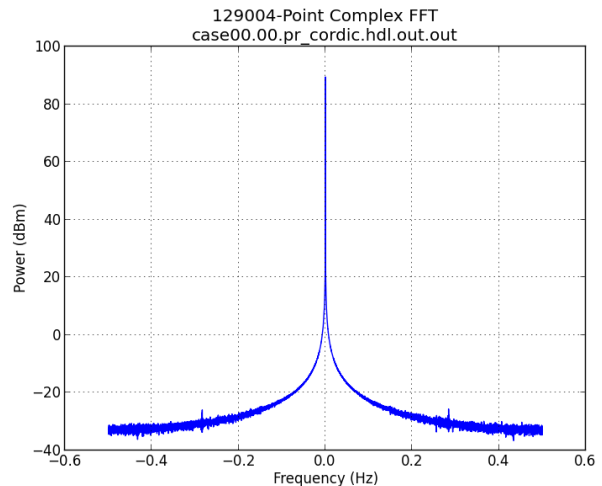


Figure 5: Frequency Domain Rectangular Output