OpenCPI
Rectangular to Polar CORDIC Component Data Sheet

OpenCPI Release: v2.4.7

Revision History

Revision	Description of Change	Date
v1.4		10/2018
v1.5		4/2019
v1.6	Convert Worker to Version 2 HDL API	5/2019
v1.6	Created rp_cordic_fsk_app.rcc worker	6/2019
v1.7	Table of Worker Configurations and Resource Utilization Table removed	5/2020

Summary - Rectangular to Polar CORDIC

Name	rp_cordic
Worker Type	Application
OpenCPI Release	v2.4.7
Last Update	11/2019
Component Library	ocpi.assets.dsp_comps
Workers	rp_cordic.hdl, rp_cordic_for_fskapp.rcc
Tested Platforms	alst4, E310(PL), isim, Matchstiq-Z1(PL), ml605, modelsim, xsim, ZedBoard(PL)

Functionality

The Rectangular to Polar CORDIC (Coordinate Rotation Digital Computer) worker implements an FM Discriminator circuit as shown in Figure 1. Complex samples are fed into the CORDIC, which output magnitude and phase values. A $d\phi$ circuit is applied to the phase to calculate real samples.

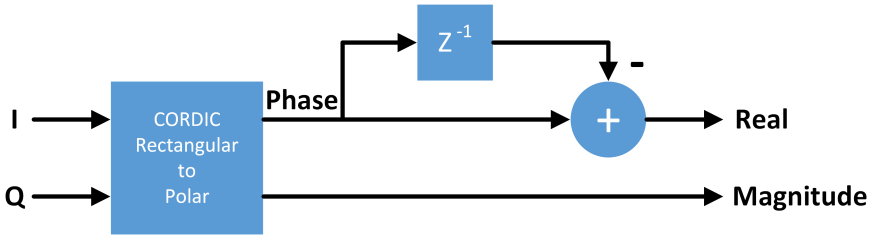


Figure 1: FM Discriminator Block Diagram

Worker Implementation Details

rp_cordic.hdl

The FM Discriminator circuit consists of two sub-circuits: one to calculate the phase and another to calculate the rate of change of the phase. The first circuit uses a CORDIC algorithm to implement the arc-tangent function to calculate the phase of a complex sinusoid. The CORDIC is also used to calculate the magnitude of the complex sinusoid. Equations 1 and 2 represent the equations used to calculate the magnitude and phase, respectively. The magnitude output is exposed as a read-only property of the worker, which could be useful for downstream gain control.

$$magnitude = \sqrt{I^2 + Q^2} \tag{1}$$

$$phase = atan(\frac{Q}{I}) \tag{2}$$

The second circuit simply uses a subtractor to implement a $d\phi$ function, which is a real signed number that is the difference in phase. Equations 3 and 4 show how to calculate $d\phi$.

$$\omega = \frac{d\phi}{dt} = 2\pi f \tag{3}$$

$$d\phi = 2\pi f dt = 2\pi f T_s = \frac{2\pi f}{F_s} = \frac{2\pi f \frac{2^{DATAWIDTH-1}}{\pi}}{F_s} = \frac{2f * 2^{DATAWIDTH-1}}{F_s} \tag{4}$$

rp_cordic_for_fskapp.rcc

This RCC worker is intended to be used in an all-RCC worker version of the FSK application as it is currently implemented. Where components are incorrectly named to reflect their implementation, rather than based on their functionality, i.e. rp_cordic vs rect_polar. Additionally, this worker implements a secondary function (i.e. FM discriminator), which does not respect the component (1 function) model. For these reasons, and those discussed below, it is highly recommended that the usage of this RCC worker be limited to the all-RCC FSK applications and not used for new applications.

This RCC (C++) worker is a work-a-like to the HDL worker, similarly named rp_cordic.hdl. However, it does NOT implement a CORDIC algorithm (i.e. stages of shifts & adds), but rather utilizes floating point cmath functions to calculate its output. Due to HDL worker implementation decisions, which affect performance, the RCC worker was required to implement additional (restrictive) functionality to emulate the HDL worker's behavior.

The calculation that it does first is the phase of all incoming iq samples. Then iterates over the phase values and FM discriminates based on the current and next phase value. The resulting output is given to the output port.

For the magnitude calculation, the HDL implements magnitude as a volatile property, however for RCC workers accessing the property is only viable at the completion of the run method. To reduce unnecessary computations the last sample that would be used to calculate the magnitude sets the property value.

The HDL worker will not output the final samples depending on the number of CORDIC stages and pipleline stages. To match the HDL implementation, the rcc worker implements a stage delay using a deque as a FIFO (m_trailing_data). The FIFO contains the previous input samples that would still be present in the HDL implementation's pipline. Calculations are made when data is present in the FIFO and enough data on the input port is present to push more data out. Then on the input data phase calculations are made. Finally the remaining input data that would be delayed is passed to the FIFO to be used in the next calculation.

Memory optimization was made to load the FIFO only with samples known to be used on the next run method call. Considerations were made to unload existing elements from the FIFO.

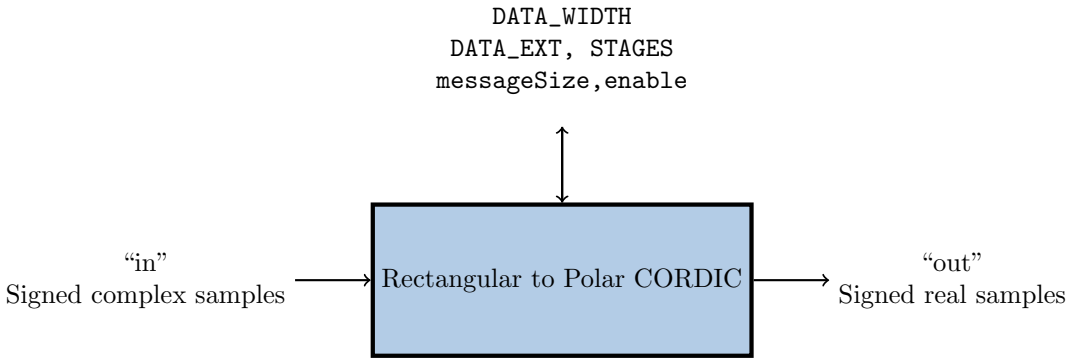
Limitations:

1) This worker currently does not work with stage delay set to zero. To do so the fm_discrimination calculation cannot be lookahead but instead looks back at the previous phase value for its calculation. A variable will have to be maintained of the previous phase value when fm discriminating.

2) This worker does not implement the CORDIC (shifts-add) algorithm, but equivalent floating point math with fixed-point adjustments (using float to integer truncation).

Block Diagrams

Top level



State Machine

None

Source Dependencies

rp_cordic.hdl

- projects/assets/components/dsp_comps/rp_cordic.hdl/rp_cordic.vhd
- projects/assets/hdl/primitives/dsp_prims/dsp_prims_pkg.vhd
  projects/assets/hdl/primitives/dsp_prims/cordic/src/cordic_rp.vhd
  projects/assets/hdl/primitives/dsp_prims/cordic/src/cordic.vhd
  projects/assets/hdl/primitives/dsp_prims/cordic/src/cordic_stage.vhd
- projects/assets/hdl/primitives/misc_prims/misc_prims_pkg.vhd
  projects/assets/hdl/primitives/misc_prims/round_conv/src/round_conv.vhd
- projects/assets/hdl/primitives/util_prims/util_prims_pkg.vhd
  projects/assets/hdl/primitives/util_prims/pd/src/peakDetect.vhd

rp_cordic_for_fskapp.rcc

- projects/assets/components/dsp_comps/rp_cordic_for_fskapp.rcc/rp_cordic_for_fskapp.rcc

Component Spec Properties

Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Usage
DATA_WIDTH	UChar	-	-		-	-	Data width of complex input and real output
DATA_EXT	UChar	-	-		-	-	Number of growth bits implemented by CORDIC
STAGES	UChar	-	-		-	-	Number of CORDIC stages to implement
messageSize	UShort	-	-	Writable	8192	8192	Number of bytes in output message (Not implemented by Version 2)
enable	Bool	-	-	Writable	Standard	true	Enable(true) or bypass(false) (Not implemented by Version 2)

Worker Properties

rp_cordic.hdl

Type	Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Usage
SpecProperty	DATA_WIDTH	-	-	-	Parameter	8-16	16	Real input and complex output data width
SpecProperty	DATA_EXT	-	-	-	Parameter	6	6	CORDIC requirement: Number of extension bits
SpecProperty	STAGES	-	-	-	Parameter	8-16	16	Number of CORDIC stages implemented
Property	PEAK_MONITOR	Bool	-	-	Parameter	Standard	true	Enable/Disable build-time inclusion of peak monitoring
Property	peak	Short	-	-	Volatile	Standard	-	Peak value of FM Discriminator output
Property	magnitude	Short	-	-	Volatile	Standard	-	Magnitude of I/Q vector. May be useful for gain control

rp_cordic_for_fskapp.rcc

Type	Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Usage
SpecProperty	DATA_WIDTH	-	-	-	Parameter	16	16	Added here to be a FSK workalike, the implementation does not use this.
SpecProperty	DATA_EXT	-	-	-	Parameter	6	6	CORDIC requirement, the implementation does not use this.
SpecProperty	STAGES	-	-	-	Parameter	8-16	16	Used to truncate data to match HDL implementation
Property	magnitude	Short	-	-	Volatile	Standard	-	Magnitude of I/Q vector. May be useful for gain control
Property	AdditionalDelay	UChar	-	-	Parameter	Standard	8	Additional number of delays over CORDIC stages (STAGES) to match HDL implementation.
Property	StageDelay	UChar	-	-	Parameter	Standard	STAGES + AdditionalDelay	Number of delays to match HDL implementation.

Component Ports

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

Worker Interfaces

rp_cordic.hdl

Type	Name	DataWidth	Advanced	Usage
StreamInterface	in	32	-	Signed complex samples
StreamInterface	out	16	InsertEOM=1	Signed real samples

Control Timing and Signals

The Rectangular to Polar CORDIC worker uses the clock from the Control Plane and standard Control Plane signals. This worker has a start-up transient delay of `STAGES`+8 valid samples. This means that once the input is ready and valid and the output is ready, it takes that many valid input samples before the first output sample is made available. The delays for this worker are itemized be as follows:

- 2 : rp_cordic.vhd
- 5 : rp_cordic.vhd/cordic_rp.vhd
- STAGES : rp_cordic.vhd/cordic_rp.vhd/cordic.vhd/cordic_stage.vhd
- 1 : cordic_rp.vhd/round_conv.vhd

Latency
`STAGES`+8 clock cycles

Worker Configuration Parameters

rp_cordic.hdl

Performance and Resource Utilization

rp_cordic.hdl

Test and Verification

One test case is implemented to validate the Rectangular to Polar CORDIC component:

1) Normal mode

The input file is a waveform with a single tone at 27 Hz sampled at 10 kHz. The complex waveform is then scaled to fixed-point signed 16-bit integers, using a maximal amplitude of 32,767. Time and frequency domain plots may be viewed in Figures 2 and 3 below, respectively.

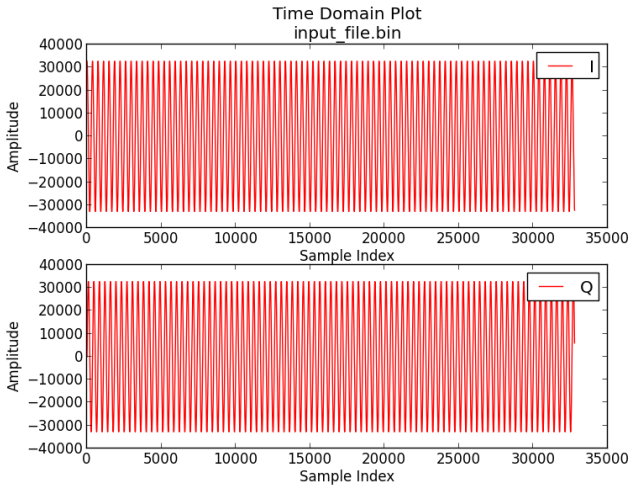


Figure 2: Time Domain Complex Tone

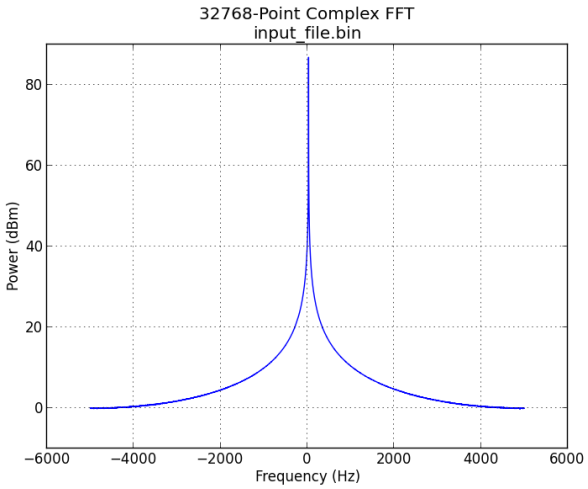


Figure 3: Frequency Domain Complex Tone

The output file is first checked that the data is not all zero and is then checked for the expected length. Once these quick checks are made, the complex input data is transformed into expected magnitude and phase arrays using Equations 1 and 2. The expected phase array then implements the FM discriminator subtractor to create an array of the expected phase difference. These two expected value arrays are compared sample-by-sample with the measured phase output array from the UUT and the single value magnitude property from the UUT. Error checks are then calculated for the average peak error, the magnitude peak error, and the phase peak error. Should any of these three error values be more than one (the difference between each expected and measured value is allowed to be no greater than one) the overall test fails. Figure 4 depicts the conversion results of the single tone input, which shows no more than a ±1 deviation from the expected result. Equation 4 is used to form Equation 5, which validates the result shown in Figure 4.

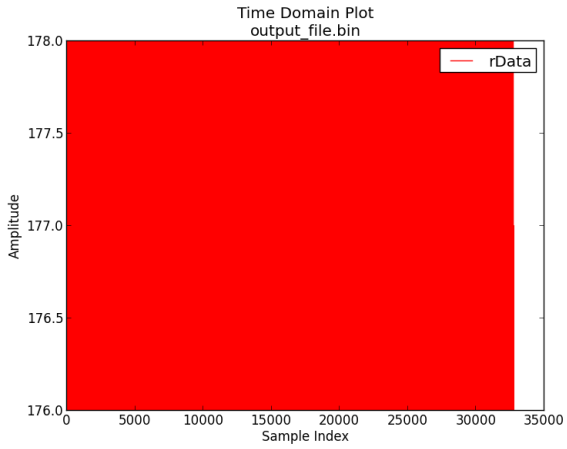


Figure 4: Time Domain Real Data

$$d\phi = \frac{2f * 2^{DATAWIDTH-1}}{F_s} = \frac{2 * 27 * 32,768}{10,000} = 176.9472 \quad (5)$$