OpenCPI
FSK App Guide

OpenCPI Release: v2.3.4

Revision History

Revision	Description of Change	Date
v1.1	Initial Release	3/2017
v1.2	Updated for OpenCPI Release 1.2	8/2017
v1.3	Updated for OpenCPI Release 1.3	1/2018
v1.3.1	Updated for OpenCPI Release 1.3.1, including FMCOMMS2/3 support	3/2018
v1.4	Updated recommendations for configuring OCPI_LIBRARY_PATH	9/2018
v1.5	Deprecated Zipper; Moved to Appendix	3/2019
v1.7	Resource Utilization Table removed	5/2020

Table of Contents

1 Document Scope

This document describes the OpenCPI FSK demo application. It includes a description of the application, instructions to setup the hardware, build of bitstreams, and execution of the application itself on various platforms.

2 Supported Hardware Setups

This app is supported on the following hardware configurations:

- Zedboard/FMCOMMS2
- Zedboard/FMCOMMS3
- x86/ML605/FMCOMMS2 in FMC LPC slot
- x86/ML605/FMCOMMS3 in FMC LPC slot
- Matchstiq-Z1

3 Description

The FSK App may be run in one of five available modes. The modes are *filerw*, *rx*, *tx*, *txrx*, and *bbloopback*. The *filerw* mode uses file_read and file_write workers to process the input using only application workers (platform agnostic) in a purely digital fashion. A block diagram of the FSK App *filerw* mode can be seen in Figure 1.

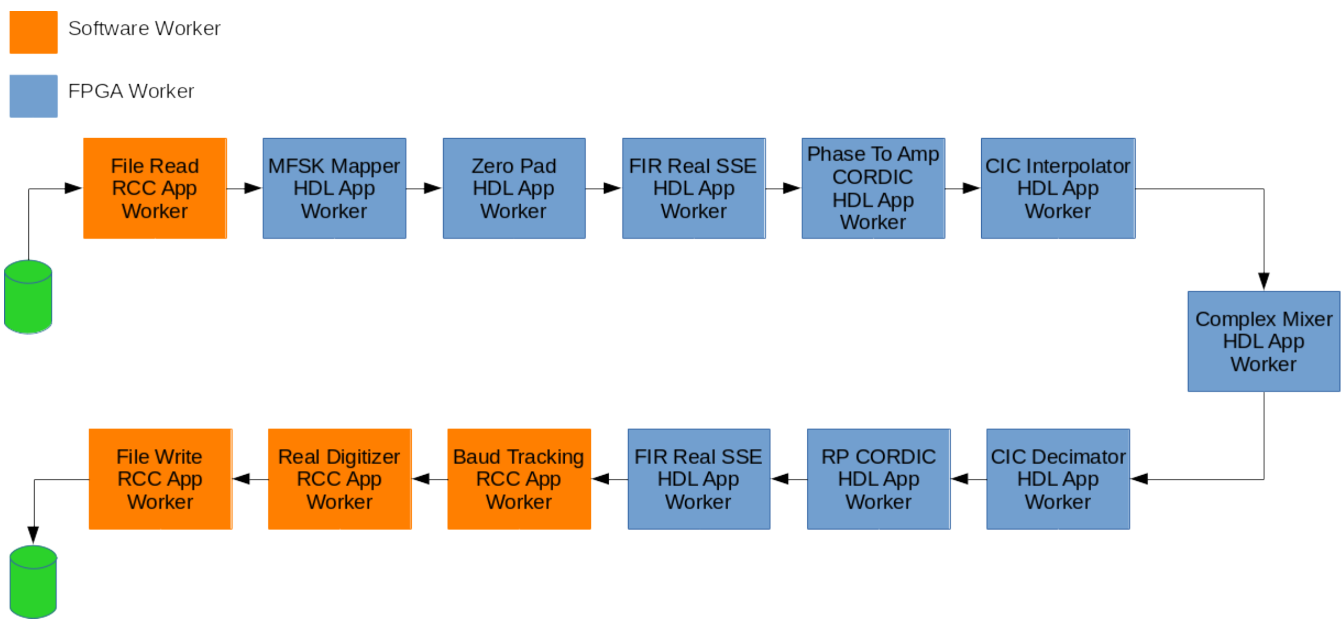

Figure 1: FSK App *filerw* mode Block Diagram

The *rx* mode inputs IQ data from the Lime ADC and processes the FSK signal down to bits that are written to file. A block diagram of the FSK App *rx* mode can be seen in Figure 2.

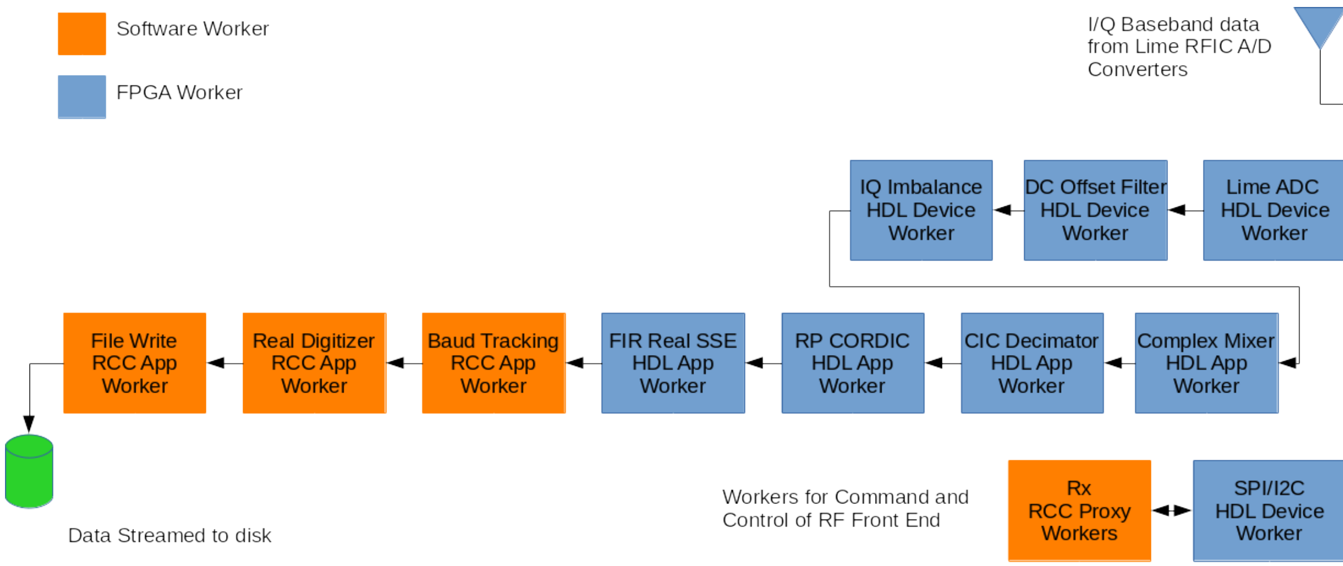

Figure 2: FSK App *rx* mode Block Diagram

The *tx* mode inputs a file from disk, modulates the input as a FSK signal, and transmits the input via the Lime DAC. A block diagram of the FSK App *tx* mode can be seen in Figure 3.

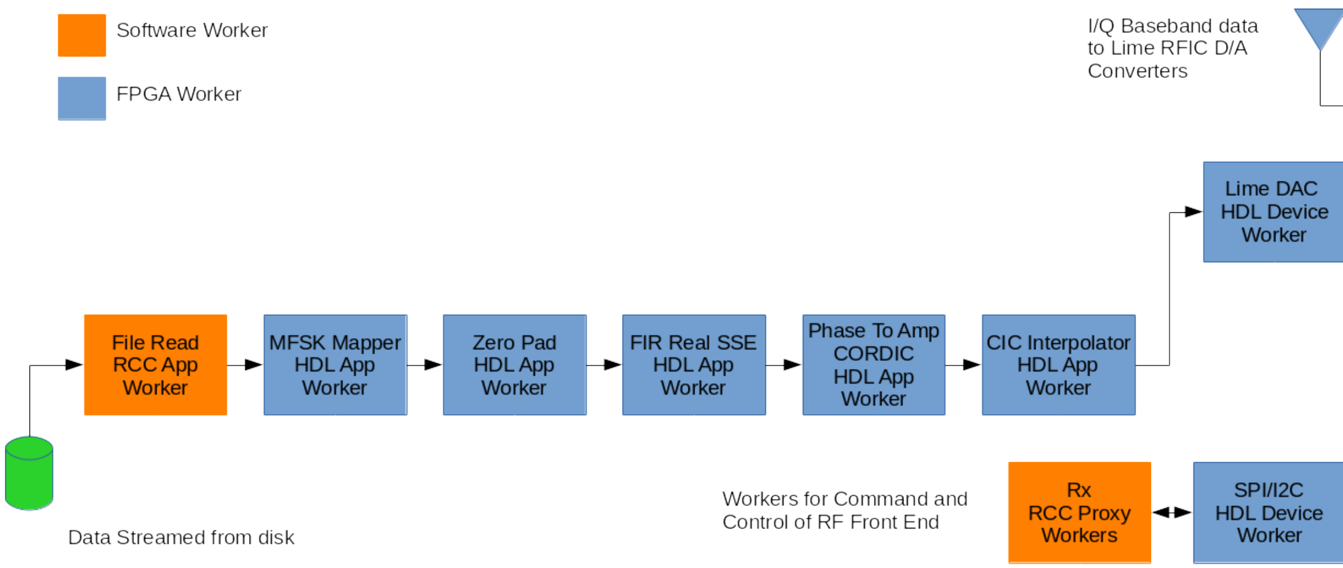

Figure 3: FSK App *tx* mode Block Diagram

The *txrx* mode is the full transceiver mode of the application which combines the functionality of Figures 2 and 3 into a single application. This mode transmits input file data as the radio RF TX output and inputs RF RX radio input that is written to file. The *bbloopback* mode utilizes the same HDL assembly and application XML as the *txrx* mode but utilizes a built-in test mode of the Lime transceiver to loopback analog data at baseband.

The *tx*, *txrx*, and *bbloopback* modes for the Matchstiq-Z1 contain the matchstiq_z1_gp_out device worker which can control the three GPIO pins present on the Matchstiq-Z1. The application XMLs give an example of how to use the device worker.

3.1 Application Specifications

3.1.1 Baud Rate

The baud rate for the application is based on the following properties:

- ADC/DAC Sample Rate (S)
- CIC Decimation/Interpolation Factor (R)
- Number of zeros inserted between symbols (Z)

It can be computed using the following equation:

$$\textit{Baud Rate} = S \ / \ R \ / \ (Z+1) \tag{1}$$

Using the default settings for the Matchstiq Z1 platform, the resulting baud rate is 64 kbaud.

$$\textit{Baud Rate} = S \ / \ R \ / \ (Z+1) = 39,936,000 \ / \ 16 \ / \ (38+1) \ = \ 64000 \tag{2}$$

3.1.2 Frequency Deviation

The theoretical FSK frequency deviation is calculated as follows. The bits of the input file are converted to Q0.15 symbols by the mfsk_mapper worker, where the symbol values are determined by the value of the worker's symbols property, which is an array. The following equation corresponds to the values mapped from bits, with s representing the bit value. Note that division by 2^{15} accounts for the Q0.15 value interpretation. The mfsk_mapper worker's symbols array property value is fixed to the values $symbols[0] = -32768$ and $symbols[1] = 32767$ in all OASs used by the ACI.

$$x_{mfsk\_mapper,out,s} = \frac{symbols[s]}{2^{15}}, s \in [0..1] \tag{3}$$

The value of these symbols is unmodified as they pass through the zero_pad worker.

$$x_{zero\_pad\_mapper,out,s} = x_{mfsk\_mapper,out,s} \tag{4}$$

The zero-padded symbols then pass through the fir_real_see worker whose taps are read from a file at runtime. The gen_rrcos_taps.py script runs at application buildtime to generate the tap file for a root-raised-cosine filter. The application Makefile passes an argument to the script that forces the filter taps to be scaled such that the maximum tap value amplitude $maxTap$ is 4096, which corresponds to the Q0.15 value $4096/2^{15} = 0.125$. Because root-raised-cosine filters do not obey the Nyquist ISI criterion, the incoming symbol values will undergo an amplitude adjustment that will vary per symbol, as shown in Figure 4. In order to to proceed with frequency deviation calculation, the nominal symbol values are approximated as being multiplied by the maximum Q0.15 tap value while passing through the fir_real_sse worker.

$$x_{fir\_real\_sse,out,s,nom} = \frac{maxTap}{2^{15}} * x_{zero\_pad\_mapper,out,s} \tag{5}$$

The frequency modulation performed by the phase_to_amp_cordic worker assigns frequency according to the following formula. Note that $x_{fir\_real\_sse,out}$ is in the Q0.15 format.

$$f_{phase\_to\_amp\_cordic,out} = \pi * x_{fir\_real\_sse,out} \ \frac{\text{radians}}{\text{sample}} \tag{6}$$

The nominal space/mark frequencies at the output of the phase_to_amp_cordic worker is given as:

$$f_{phase\_to\_amp\_cordic,out,s,nom} = \pi * \frac{maxTap}{2^{15}} * \frac{symbols[s]}{2^{15}} \ \frac{\text{radians}}{\text{sample}} \tag{7}$$

Figure 5 displays the nominal space and mark frequencies versus that actual instantaneous frequency of the output of the phase_to_amp_cordic worker that will occur at runtime. The output of phase_to_amp_cordic is upsampled by an interpolating CIC filter, which has the effect of dividing the space/mark frequencies. The interpolation factor is fixed via the parameter property/value of $R = 16$ within all assemblies used by the ACI and all its OASs.

$$f_{cic\_int,out,s,nom} = \frac{f_{phase\_to\_amp\_cordic,out,s,nom}}{R} \tag{8}$$

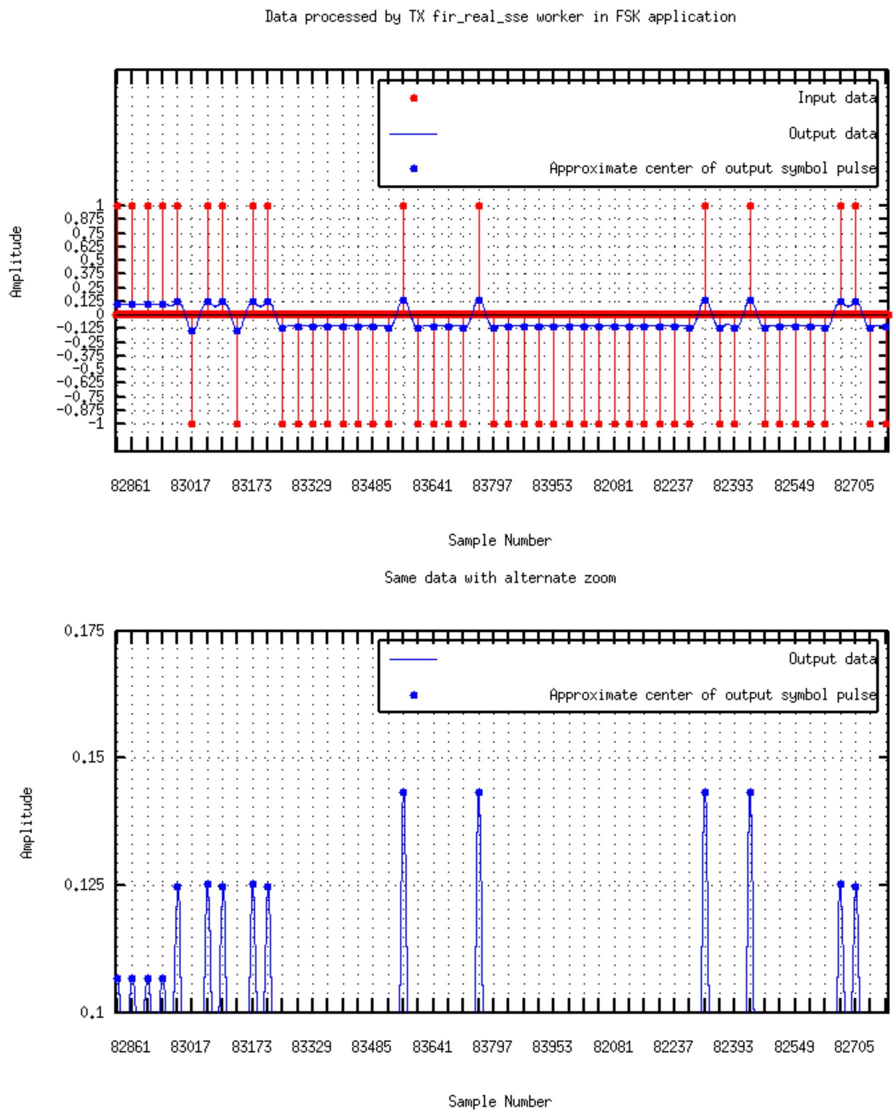

Figure 4: Data processed by TX fir_real_sse worker.

The output data of the CIC is passed to the DAC, which has a configurable sample rate f_s. The resulting space/mark frequencies at the output of the DAC is given as:

$$f_{DAC,out,s,nominal} = \left[f_{cic\_int,out,s,nom} \frac{radians}{sample} \right] * \left[fs \frac{samples}{second} \right] * \left[\frac{1}{2\pi \ radians} \right] = \frac{maxTap}{2^{15}} * \frac{symbols[s]}{2^{16}} * \frac{f_s}{R} \text{ Hz} \quad (9)$$

This results in a final frequency deviation calculation:

$$\begin{aligned} f_{\Delta,nominal} = f_{DAC,out,1,nominal} - f_{DAC,out,0,nominal} &= \frac{maxTap}{2^{15}} * \frac{symbols[1] - symbols[0]}{2^{16}} * \frac{f_s}{R} \\ = 0.125 * \frac{32767 - (-32768)}{2^{16}} * \frac{f_s}{16} &= \frac{65535}{2^{23}} f_s \text{ Hz} \end{aligned}$$

Using the ACI's default prompted setting of $f_s = 4,000,000$ Hz for the FMCOMMS2/3 transceivers, the resulting nominal frequency deviation would be approximately 31,249.5 Hz.

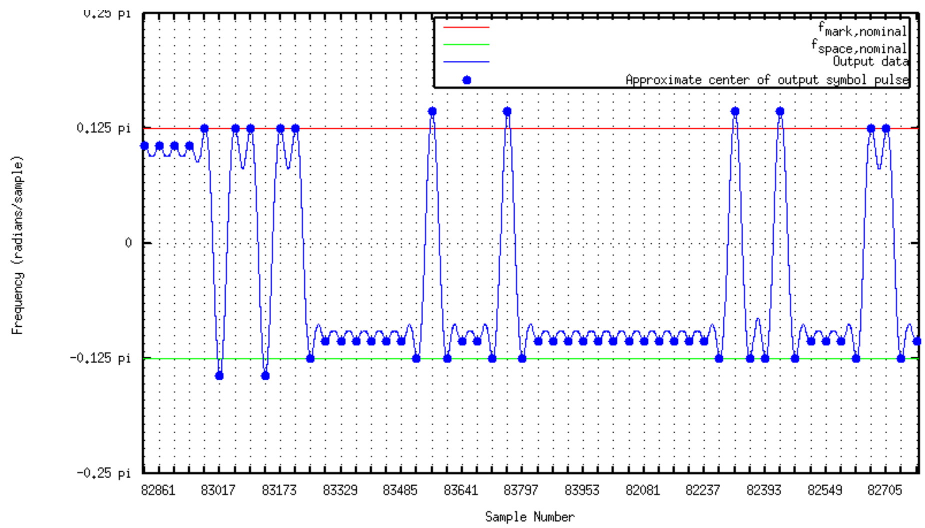

Figure 5: Instantaneous frequency of phase_to_amp_cordic output.

4 Building the Application

4.1 Dependencies

The tables below breakdown the workers used within the various platforms and modes of the FSK App. Appendix A shows the exact worker configurations used in the HDL assemblies. See the individual component data sheets for more information and build instructions. Similarly, the HDL platform worker and configurations for the intended radio must be compiled prior to building the various FSK bitstreams.

4.2 FSK Mode Configurations

4.2.1 Common to all Hardware

Application XML	filerw	rx	tx	txrx	bbloopback
app_fsk_filerw (dependency only, no build required)	x				
HDL Assemblies	filerw	rx	tx	txrx	bbloopback
fsk_filerw	x				
dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic_fir_real		x			
mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int			x		
fsk_modem				x	x
RX Path Workers	filerw	rx	tx	txrx	bbloopback
dc_offset_filter.hdl		x		x	x
iq_imbalance_fixer.hdl		x		x	x
complex_mixer.hdl	x	x		x	x
cic_dec.hdl	x	x		x	x
rp_cordic.hdl	x	x		x	x
fir_real_sse.hdl	x	x		x	x
baudTracking.rcc	x	x		x	x
real_digitizer.rcc	x	x		x	x
file_write.rcc	x	x		x	x
TX Path Workers	filerw	rx	tx	txrx	bbloopback
file_read.rcc	x		x	x	x
mfsk_mapper.hdl	x		x	x	x
zero_pad.hdl	x		x	x	x
fir_real_sse.hdl	x		x	x	x
phase_to_amp_cordic.hdl	x		x	x	x
cic_int.hdl	x		x	x	x

4.2.2 Additional Dependencies for FMCOMMS2

Application XML	filerw	rx	tx	txrx	bbloopback
app_fsk_rx_fmcomms2 (dependency only, no build required)		x			
app_fsk_tx_fmcomms2 (dependency only, no build required)			x		
app_fsk_txrx_fmcomms2 (dependency only, no build required)				x	
RX or TX Path Workers	filerw	rx	tx	txrx	bbloopback
ad9361_data_sub.hdl		x	x	x	
RX Path Workers	filerw	rx	tx	txrx	bbloopback
ad9361_adc.hdl		x		x	
ad9361_adc_sub.hdl		x		x	
TX Path Workers	filerw	rx	tx	txrx	bbloopback
ad9361_dac.hdl			x	x	
ad9361_dac_sub.hdl			x	x	
Endpoint Proxies	filerw	rx	tx	txrx	bbloopback
fmcomms_2_3_rx.rcc		x		x	
fmcomms_2_3_tx.rcc			x	x	
SPI Command and Control	filerw	rx	tx	txrx	bbloopback
ad9361_config.hdl		x	x	x	
ad9361_config_proxy.rcc		x	x	x	
ad9361_spi.hdl		x	x	x	
I2C Command and Control	filerw	rx	tx	txrx	bbloopback
fmcomms_2_3_i2c.hdl		x	x	x	

4.2.3 Additional Dependencies for FMCOMMS3

Application XML	filerw	rx	tx	txrx	bbloopback
app_fsk_rx_fmcomms3 (dependency only, no build required)		x			
app_fsk_tx_fmcomms3 (dependency only, no build required)			x		
app_fsk_txrx_fmcomms3 (dependency only, no build required)				x	
RX or TX Path Workers	filerw	rx	tx	txrx	bbloopback
ad9361_data_sub.hdl		x	x	x	
RX Path Workers	filerw	rx	tx	txrx	bbloopback
ad9361_adc.hdl		x		x	
ad9361_adc_sub.hdl		x		x	
TX Path Workers	filerw	rx	tx	txrx	bbloopback
ad9361_dac.hdl			x	x	
ad9361_dac_sub.hdl			x	x	
Endpoint Proxies	filerw	rx	tx	txrx	bbloopback
fmcomms_2_3_rx.rcc		x		x	
fmcomms_2_3_tx.rcc			x	x	
SPI Command and Control	filerw	rx	tx	txrx	bbloopback
ad9361_config.hdl		x	x	x	
ad9361_config_proxy.rcc		x	x	x	
ad9361_spi.hdl		x	x	x	
I2C Command and Control	filerw	rx	tx	txrx	bbloopback
fmcomms_2_3_i2c.hdl		x	x	x	

4.2.4 Additional Dependencies for Matchstiq-Z1

Application XML	filerw	rx	tx	txrx	bbloopback
app_fsk_rx_matchstiq_z1 (dependency only, no build required)		x			
app_fsk_tx_matchstiq_z1 (dependency only, no build required)			x		
app_fsk_txrx_matchstiq_z1 (dependency only, no build required)				x	x
RX Path Workers	filerw	rx	tx	txrx	bbloopback
lime_adc.hdl		x		x	x
TX Path Workers	filerw	rx	tx	txrx	bbloopback
lime_dac.hdl			x	x	x
Endpoint Proxies	filerw	rx	tx	txrx	bbloopback
matchstiq_z1_rx.rcc		x		x	x
matchstiq_z1_tx.rcc			x	x	x
SPI Command and Control	filerw	rx	tx	txrx	bbloopback
lime_rx_proxy.rcc		x		x	x
lime_rx.hdl		x		x	x
lime_tx_proxy.rcc			x	x	x
lime_tx.hdl			x	x	x
lime_spi.hdl		x	x	x	x
I2C Command and Control	filerw	rx	tx	txrx	bbloopback
si5338_proxy.rcc		x	x	x	x
si5338.hdl		x	x	x	x
matchstiq_z1_avr_proxy.rcc		x	x	x	x
matchstiq_z1_avr.hdl		x	x	x	x
tmp100_proxy.rcc		x	x	x	x
tmp100.hdl		x	x	x	x
matchstiq_z1_pca9535_proxy.rcc		x	x	x	x
pca9535.hdl		x	x	x	x
matchstiq_z1_i2c.hdl		x	x	x	x

4.3 Performance and Resource Utilization

4.3.1 filerw

4.3.2 tx

4.3.3 rx

4.3.4 txrx/bbloopback

4.4 Executable

The software portion of the application consists of a C++ program written using the OpenCPI C++ API as well as RCC proxy workers for command and control functionality. The program references the appropriate application XML file for the requested hardware and mode. The app XML files contain all of the property settings for the components in each application, except for the configuration of the endpoint proxy(ies). These endpoint proxy-related settings are passed via command-line prompted values to the appropriate endpoint proxy workers.
To build for the host platform (which is the case if ML605 or Stratix IV platform is intended to be used), run the following commands from the FSK directory:

```
ocpidev build
```

To build for the Zedboard or Matchstiq-Z1 (which run the xilinx13_3 PetaLinux operating system), run the following command from the FSK directory:

```
ocpidev build --rcc-platform xilinx13_3
```

5 Testing the Application

5.1 Baud Synchronization

The input filename for the application is idata/Os.jpeg. It is modified from an original JPEG image (see Figure 6) with data prepended to it for baud synchronization purposes (240 bytes of an alternating 1-0 pattern followed by 2 bytes are the value 0xFACE). In the receive data stream, real_digitizer.rcc worker makes symbol/bit decisions and only sends out bits that occur after the first detected 0xFACE bit pattern (b1111101011001110). The image is also padded with 40 bytes of zeros. This padding ensures that all image data is pushed through the signal processing chain.

5.2 Sample test setup

The test setup varies per mode of operation. The base test setup includes a hardware platform of choice and appropriate power and USB cables (Matchstiq-Z1 uses the USB cable to access the terminal via a program such as *screen* or over a USB-over-Ethernet connection; Stratix IV and ML605 require a USB cable for JTAG loading). All other test modes expand upon this base configuration, and may or may not include additional RF cabling or external equipment.

The *filerw* mode requires only the base test setup since no transceiver operations are actuated (data passes to and from the FPGA in a "loopback" fashion). Upon application execution, the expected result is written to odata/out_app_fsk_filerw.bin, which is a transmitted copy of the input file idata/Os.jpeg, without the prepended synchronizing pattern.

The *bbloopback* mode is similar to the *filerw* mode, but data goes beyond the FPGA through the radio's built-in analog baseband loopback, and back into the FPGA. Because the data never reaches the TX/RX connectors, no external RF cabling is required. The expected result is a transmitted copy of the idata/Os.jpeg file as the output odata/out_app_fsk_bbloopback.bin, without the prepended synchronizing pattern. Since this test is executed for a duration, rather than total image recognition, it is not uncommon that the output file will be larger than the actual size of the input image.

The next mode is the *rx* mode, which requires the base test setup with an RX antenna, as well as another transmitter (such as another platform running the *tx* mode) in order to broadcast a known FSK signal. Optionally, a spectrum analyzer may be connected to the transmitter to visually verify that the signal being fed into the radio's RX input is an FSK signal at the correct RF frequency and bandwidth. The output is written to the file odata/out_app_fsk_rx.bin, without the prepended synchronizing pattern. Since this test is executed for a duration, rather than total image recognition, it is not uncommon that the output file will be larger than the actual size of the input image.

The next mode is the *tx* mode, which requires the base test setup with a TX antenna, as well as some hardware to verify the transmission, such as a spectrum analyzer or an additional radio running in *rx* mode. In this mode the input file idata/Os.jpeg is transmitted out of the RF TX output of the radio.

The final mode is the *txrx* mode. This mode requires a base test setup with either a SMA loopback cable connecting the TX output of the radio to the RX input of the radio, or separate RX/TX antennas if RF usage is desired. An

RF splitter can also be used to optionally connect a spectrum analyzer to the RF signal for visual verification. The default values for RF gain assume that an RF splitter is being used. The output is written to the file odata/out_app_fsk_txrx.bin, without the prepended synchronizing pattern. Since this test is executed for a duration, rather than total image recognition, it is not uncommon that the output file will be larger than the actual size of the input image.

5.3 make show

In order to test the application using the various modes mentioned above, `make show` can be run from the `applications/FSK` directory. This provides instructions (for Zynq-Based Platforms) for setting `OCPI_LIBRARY_PATH` on the hardware platform and then running the application. Finally, it explains how to verify the output data on the development computer. The following sections provide further insight into these instructions.

5.4 Artifacts

Before running the application, the location of the required deployable artifacts must be specified in the `OCPI_LIBRARY_PATH` environment variable. Separate artifacts are needed for each RCC worker, and one artifact for the required FPGA image. Furthermore, artifacts differ depending on which mode the application is to be run in. Appendix B includes a list of the artifacts required for each platform and mode.

5.5 Arguments to executable

There is only one required initial argument to the FSK App executable, which defines the mode of execution. There is an optional initial argument to the FSK App executable that defines whether or not to run in debug mode. The app prompts the user at runtime for additional values, which vary depending upon the selected mode of operation. Running the application without any arguments prints the usage instructions. The non-initial optional arguments prompt the user to override the default value(s), and primarily configure the RF front end of the given platform using one or more endpoint proxies.

The arguments to the executable are summarized in the following table:

Argument	Mode	Description
mode	n/a	filerw, rx, tx, txrx, bbloopback
debug_mode	optional - ‘d’ or blank	enables initial and final dump of all properties

The prompts performed by the executable are summarized in the following table:

Argument	Mode	Description
RF frontend (ML605 and zed only)	rx, txrx	zipper, FMCOMMS2, or FMCOMMS3
runtime	filerw, rx, tx, txrx, bbloopback	run time of application in seconds
rx_sample_rate	rx, txrx, bbloopback	RX RF sample rate in Msps
rx_rf_center_freq	rx, txrx, bbloopback	RX RF tuning frequency in MHz
rx_rf_bw	rx, txrx, bbloopback	RX RF bandwidth in MHz
rx_rf_gain	rx, txrx, bbloopback	RX RF gain in dB
rx_bb_bw	rx, txrx, bbloopback	RX baseband bandwidth in MHz
rx_bb_gain	rx, txrx, bbloopback	RX baseband gain in dB
rx_if_center_freq	rx, txrx, bbloopback	RX IF tuning frequency in MHz. 0 disables IF tuning
tx_sample_rate	tx, txrx, bbloopback	TX RF sample rate in Msps
tx_rf_center_freq	tx, txrx, bbloopback	TX RF tuning frequency in MHz
tx_rf_gain	tx, txrx, bbloopback	TX RF gain in dB
tx_bb_bw	tx, txrx, bbloopback	TX baseband bandwidth in MHz
tx_bb_gain	tx, txrx, bbloopback	TX baseband gain in dB

Example arguments for the **FMCOMMS2** card using the txrx mode with an SMA loopback cable between FMCOMMS2 SMA ports RX1A and TX1A:

Parameter	Value
RF frontend	FMCOMMS2
Runtime (s)	20
RX SMA channel	RX1A
TX SMA channel	TX1A
rx_sample_rate	4
rx_rf_center_freq	2400 (default)
rx_rf_bw	-1 (default)
rx_rf_gain	24
rx_bb_bw	4
rx_bb_gain	-1 (default)
rx_if_center_freq	0
tx_sample_rate	4
tx_rf_center_freq	2400 (default)
tx_rf_bw	-1 (default)
tx_rf_gain	-34
tx_bb_bw	4
tx_bb_gain	-1 (default)

Example arguments for the **FMCOMMS3** card using the txrx mode with an SMA loopback cable between FMCOMMS3 SMA ports RX1A and TX1A:

Parameter	Value
RF frontend	FMCOMMS3
Runtime (s)	20
RX SMA channel	RX1A
TX SMA channel	TX1A
rx_sample_rate	4
rx_rf_center_freq	2400 (default)
rx_rf_bw	-1 (default)
rx_rf_gain	24
rx_bb_bw	4
rx_bb_gain	-1 (default)
rx_if_center_freq	0
tx_sample_rate	4
tx_rf_center_freq	2400 (default)
tx_rf_bw	-1 (default)
tx_rf_gain	-34
tx_bb_bw	4
tx_bb_gain	-1 (default)

5.6 Library Path Requirements

Prior to running the application, the environment variable OCPI_LIBRARY_PATH must be configure, such that, all of the FSK application's run-time artifacts can be located. OpenCPI conveniently provides access to a project's run-time artifacts at the top-level of each project in a directory called artifacts. Reference the OpenCPI Application Development Guide for more about OCPI_LIBRARY_PATH.

Note that the Stratix IV GX230 and ML605 hardware setups require the intended slot-specific bitstream's file location to be first in OCPI_LIBRARY_PATH. This is necessary because ocpirun's aritifact compatibility test does not currently differentiate between slot-connected device workers for multiple bitstreams that contain the same device worker, in the scenario where what differentiates the bitstreams is the device worker's slot connectivity.

The following are recommendations for configuring the OCPI_LIBRARY_PATH based on the mode of FSK, platform, the use of a daughter card and specific slot that card is installed. For all recommendations:

- All paths are relative to the applications/FSK/ directory.
- It is assumed (for PCI/host platforms) that the `core` and `assets` projects are named as such and exist in the same parent directory.

Recommended Library Path for Matchstiq-Z1 or Zedboard

For these platforms, follow the instructions contained in the FSK application's Makefile. They can be viewed by opening the Makefile in an editor, or by executing "`make show`" from within the assets/applications/FSK/.

Recommended Library Path *filerw* mode for all host/PCI platforms
`OCPI_LIBRARY_PATH=../../core/artifacts:../../artifacts`

Recommended Library Path for ML605/FMCOMMS2/3-HPC

Not Supported

Recommended Library Path for ML605/FMCOMMS2/3-LPC

`OCPI_LIBRARY_PATH=../../core/artifacts:../../artifacts`

5.7 Expected results

In the case of the *filerw*, *rx*, *txrx*, and *bbloopback* modes, assuming transmission of the idata/Os.jpeg input file, the expected result is a transmitted copy of the JPEG file. A Linux program such as Eye of GNOME (eog) may be used to display the JPEG file. The file is shown in Figure 6.

In the case of the *tx* mode, verification is obtained by viewing the RF spectrum on a spectrum analyzer. An example of the transmitted spectrum may be seen in Figure 7.

Figure 6: FSK input file

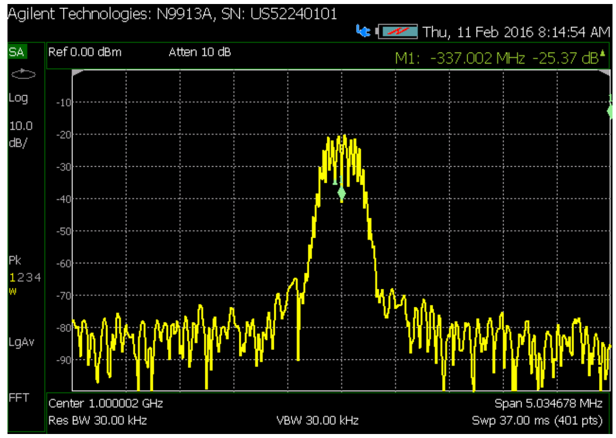

Figure 7: Output of FSK App RF transmit

5.8 Known Issues

- The demodulation algorithm currently suffers from limited carrier recovery ability which can cause the output image to be corrupted or non-existent when using the *rx* or *txrx* mode. If using an SMA loopback for txrx mode on an RF transceiver where the RX and TX data stream's LOs are sourced by the same clock, carrier recovery is not expected to be an issue.

Appendices

A Worker Parameters

Common to all hardware

- dc_offset_filter.hdl
  DATA_WIDTH_p = 16
  PEAK_MONITOR_p = true
- iq_imbalance_fixer.hdl
  DATA_WIDTH_p = 16
  ACC_PREC_p = 38
  PEAK_MONITOR_p = true
- complex_mixer.hdl
  CHIPSCOPE_p = false
  NCO_DATA_WIDTH_p = 12
  INPUT_DATA_WIDTH_p = 12
  CORDIC_STAGES_p = 16
  PEAK_MONITOR_p = true
- cic_dec.hdl
  N = 3
  M = 1
  R = 16
  DIN_WIDTH = 16
  ACC_WIDTH = 28
  DOUT_WIDTH = 16
- rp_cordic.hdl
  DATA_WIDTH = 16
  DATA_EXT = 6
  STAGES = 16
- fir_real_sse.hdl (rx_fir_real)
  NUM_TAPS_p = 64
  DATA_WIDTH_p = 16
  COEFF_WIDTH_p = 16
- mfsk_mapper.hdl
  M_p = 2
- zero_pad.hdl
  DWIDTH_p = 16

ML605 (with FMCOMMS2/3 card in FMC HPC slot)

- fmcomms_2_3_i2c.hdl
  CP_CLK_FREQ_p = 125e6
  FMC_GA1 = 0
  FMC_GA0 = 0
- ad9361_spi.hdl
  CP_CLK_FREQ_HZ_p = 125e6
- ad9361_data_sub.hdl
  LVDS_p = true
  DATA_CLK_Delay = 2
  RX_Data_Delay = 0
  FB_CLK_Delay = 7
  TX_Data_Delay = 0

ML605 (with FMCOMMS2/3 card in FMC LPC slot)

- fmcomms_2_3_i2c.hdl
  CP_CLK_FREQ_p = 125e6
  FMC_GA1 = 1
  FMC_GA0 = 0
- ad9361_spi.hdl
  CP_CLK_FREQ_HZ_p = 125e6
- ad9361_data_sub.hdl
  LVDS_p = true
  DATA_CLK_Delay = 2
  RX_Data_Delay = 0
  FB_CLK_Delay = 7
  TX_Data_Delay = 0

Zedboard FMCOMMS2/3 configurations)

- fmcomms_2_3_i2c.hdl
  CP_CLK_FREQ_p = 100e6
  FMC_GA1 = 0
  FMC_GA0 = 0
- ad9361_spi.hdl
  CP_CLK_FREQ_HZ_p = 100e6
- ad9361_data_sub.hdl
  LVDS_p = true
  DATA_CLK_Delay = 2
  RX_Data_Delay = 0
  FB_CLK_Delay = 7
  TX_Data_Delay = 0

Matchstiq-Z1 configurations

- lime_adc.hdl
  DRIVE_CLK_p = false
  USE_CLK_IN_p = false
  USE_CTL_CLK_p = false
  USE_CLK_OUT_p = true
- si5338.hdl
  CLKIN_PRESENT_p = true
  CLKIN_FREQ_p = 3.072e7
  XTAL_PRESENT_p = false
  XTAL_FREQ_p = 0
  OUTPUTS_PRESENT_p = true, false
  INTR_CONNECTED_p = false
- matchstiq_z1_i2c.hdl
  NUSERS_p = 5
  SLAVE_ADDRESS_p = 0x45, 0x71, 0x48, 0x21, 0x20
  CLK_CNT_p = 199

Zipper-related platforms (Zedboard, Stratix IV, ML605)

- lime_adc.hdl
  DRIVE_CLK_p = false
  USE_CLK_IN_p = true
  USE_CTL_CLK_p = false
  USE_CLK_OUT_p = false
- si5351.hdl
  CLKIN_PRESENT = true
  CLKIN_FREQ = 3.072e7
  XTAL_PRESENT = false
  XTAL_FREQ = 0
  VC_PRESENT = false
  OUTPUTS_PRESENT = 0,0,1,1,1,1,0,0
  OEB_MODE = low
  INTR_CONNECTED = false
- zipper_i2c.hdl
  NUSERS_p = 2

B Artifacts

B.1 Zedboard/FMCOMMS2/3

filerw (FMCOMMS2/3 not required)

- fsk_filerw_zed_base.bitz
- target-xilinx13_3/file_read.so
- target-xilinx13_3/Baudtracking_simple.so
- target-xilinx13_3/real_digitizer.so
- target-xilinx13_3/file_write.so

rx

- dc_offset_iq_imbalance_mixer_cic_dec_timestamper_zed_cfg_1rx_0 tx_fmcomms_2_3_lpc_lvds_cnt_1rx_0tx_thruasm_fmcomms_2_3_lpc_LVDS_zed.bitz
- target-xilinx13_3/Baudtracking_simple.so
- target-xilinx13_3/real_digitizer.so
- target-xilinx13_3/file_write.so
- target-xilinx13_3/ad9361_config_proxy.so
- target-xilinx13_3/fmcomms_2_3_rx.so

tx

- mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int_zed_cfg_0rx_1tx_fmcomms_2_3_lpc_lvds_cnt_0rx_1tx_thruasm_fmcomms_2_3_lpc_LVDS_zed.bitz

- target-xilinx13_3/file_read.so
- target-xilinx13_3/zipper_tx.so
- target-xilinx13_3/ad9361_config_proxy.so
- target-xilinx13_3/fmcomms_2_3_tx.so

txrx

- fsk_modem_zed_cfg_1rx_1tx_fmcomms_2_3_lpc_lvds_cnt_1rx_1tx_thruasm_fmcomms_2_3_lpc_LVDS_zed.bitz
- target-xilinx13_3/file_read.so
- target-xilinx13_3/Baudtracking_simple.so
- target-xilinx13_3/real_digitizer.so
- target-xilinx13_3/file_write.so
- target-xilinx13_3/ad9361_config_proxy.so
- target-xilinx13_3/fmcomms_2_3_rx.so
- target-xilinx13_3/fmcomms_2_3_tx.so

B.2 Matchstiq-Z1

filerw

- fsk_filerw_matchstiq_z1_base.bitz
- target-xilinx13_3/file_read.so
- target-xilinx13_3/Baudtracking_simple.so
- target-xilinx13_3/real_digitizer.so
- target-xilinx13_3/file_write.so

rx

- dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic_fir_real_matchstiq_z1_matchstiq_z1_rx_cnt_1rx_0tx_thruasm_matchstiq_z1.bitz

- target-xilinx13_3/Baudtracking_simple.so
- target-xilinx13_3/real_digitizer.so
- target-xilinx13_3/file_write.so
- target-xilinx13_3/matchstiq_z1_rx.so
- target-xilinx13_3/lime_rx_proxy.so
- target-xilinx13_3/si5338_proxy.so
- target-xilinx13_3/matchstiq_z1_avr_proxy.so
- target-xilinx13_3/tmp100_proxy.so
- target-xilinx13_3/matchstiq_z1_pca9535_proxy.so

tx

- mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int_matchstiq_z1_matchstiq_z1_tx_cnt_0rx_1tx_thruasm_matchstiq_z1.bitz

- target-xilinx13_3/file_read.so
- target-xilinx13_3/matchstiq_z1_tx.so
- target-xilinx13_3/lime_tx_proxy.so
- target-xilinx13_3/si5338_proxy.so
- target-xilinx13_3/matchstiq_z1_avr_proxy.so
- target-xilinx13_3/tmp100_proxy.so
- target-xilinx13_3/matchstiq_z1_pca9535_proxy.so

txrx/bbloopback

- fsk_modem_matchstiq_z1_matchstiq_z1_rx_tx_cnt_1rx_1tx_thruasm_matchstiq_z1.bitz

- target-xilinx13_3/file_read.so
- target-xilinx13_3/Baudtracking_simple.so
- target-xilinx13_3/real_digitizer.so
- target-xilinx13_3/file_write.so
- target-xilinx13_3/matchstiq_z1_rx.so
- target-xilinx13_3/matchstiq_z1_tx.so
- target-xilinx13_3/lime_rx_proxy.so
- target-xilinx13_3/lime_tx_proxy.so
- target-xilinx13_3/si5338_proxy.so
- target-xilinx13_3/matchstiq_z1_avr_proxy.so
- target-xilinx13_3/tmp100_proxy.so
- target-xilinx13_3/matchstiq_z1_pca9535_proxy.so

B.3 ML605/FMCOMMS2/3

filerw (FMCOMMS2/3 not required)

- fsk_filerw_ml605_base.bitz
- target-centos7/file_read.so
- target-centos7/Baudtracking_simple.so
- target-centos7/real_digitizer.so
- target-centos7/file_write.so

rx

- target-centos7/file_read.so
- target-centos7/Baudtracking_simple.so
- target-centos7/real_digitizer.so
- target-centos7/ad9361_config_proxy.so
- target-centos7/fmcomms_2_3_rx.so

For FMCOMMS2/3 plugged into FMC LPC:

- dc_offset_iq_imbalance_mixer_cic_dec_timestamper_ml605_cfg_1rx_0tx _fmcomms_2_3_lpc_lvds_cnt_1rx_0tx_thruasm_fmcomms_2_3_lpc_LVDS_ml605.bitz

For FMCOMMS2/3 plugged into FMC HPC:

- dc_offset_iq_imbalance_mixer_cic_dec_timestamper_ml605_cfg_1rx_0tx _fmcomms_2_3_hpc_lvds_cnt_1rx_0tx_thruasm_fmcomms_2_3_hpc_LVDS_ml605.bitz

tx

- target-centos7/file_read.so
- target-centos7/ad9361_config_proxy.so
- target-centos7/fmcomms_2_3_tx.so

txrx

- target-centos7/file_read.so
- target-centos7/Baudtracking_simple.so
- target-centos7/real_digitizer.so
- target-centos7/ad9361_config_proxy.so
- target-centos7/fmcomms_2_3_rx.so
- target-centos7/fmcomms_2_3_tx.so

C Deprecated Zipper

Beginning with OpenCPI Version 1.5, Support for Lime Microsystems' Zipper card is now deprecated, and the following have been removed from the main body of this document:

C.1 Supported Hardware Setups

This app is supported on the following hardware configurations:

- Zedboard/Zipper/MyriadRF
- x86/Stratix IV GX development kit (230 Edition)/Zipper/MyriadRF in HSMC A slot
- x86/Stratix IV GX development kit (230 Edition)/Zipper/MyriadRF in HSMC B slot
- x86/ML605/Zipper/MyriadRF in FMC LPC slot
- x86/ML605/Zipper/MyriadRF in FMC HPC slot

C.2 Known Issues

- For more information on known limitations when using the Zipper-related platforms (Zedboard, Stratix IV, ML605), see the document Myriad-RF_1_Zipper_Limitations included with this project.
- On x86 host machines with more than one Stratix IV and/or ML605s plugged into PCIe slots, this app will assume that the first found Stratix IV/ML605 has a Zipper/MyriadRF plugged in. The first found Stratix IV/ML605 will be used during execution. While there are means to address this issue, they have not been implemented for the current release.

C.3 Additional Dependencies for Zipper-related platforms (Zedboard, Stratix IV, ML605)

Application XML	filerw	rx	tx	txrx	bbloopback
app_fsk_rx_zipper (dependency only, no build required)		x			
app_fsk_tx_zipper (dependency only, no build required)			x		
app_fsk_txrx_zipper (dependency only, no build required)				x	x
RX Path Workers	filerw	rx	tx	txrx	bbloopback
lime_adc.hdl		x		x	x
TX Path Workers	filerw	rx	tx	txrx	bbloopback
lime_dac.hdl			x	x	x
Endpoint Proxies	filerw	rx	tx	txrx	bbloopback
zipper_rx.rcc		x		x	x
zipper_tx.rcc			x	x	x
SPI Command and Control	filerw	rx	tx	txrx	bbloopback
lime_rx_proxy.rcc		x		x	x
lime_rx.hdl		x		x	x
lime_tx_proxy.rcc			x	x	x
lime_tx.hdl			x	x	x
lime_spi.hdl		x	x	x	x
I2C Command and Control	filerw	rx	tx	txrx	bbloopback
si5351_proxy.rcc		x	x	x	x
si5351.hdl		x	x	x	x

C.4 Zipper Recommended Library Path

Recommended Library Path for Stratix IV GX230/Zipper in HSMC A

rx
```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.dc_offset_iq_imbalance_mixer_cic_dec_\
rp_cordic_fir_real_alst4_alst4_zipper_hsmc_alst4_port_a_rx_cnt_1rx_0tx_thruasm_zipper_\
hsmc_a_alst4.hdl.0.alst4.gz\
:../../core/artifacts:../../artifacts
```

tx
```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int_\
alst4_alst4_zipper_hsmc_alst4_port_a_tx_cnt_0rx_1tx_thruasm_zipper_hsmc_a_alst4.hdl.0.alst4.gz\
:../../core/artifacts:../../artifacts
```

txrx/bbloopback
```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.fsk_modem_alst4_alst4_zipper_hsmc_alst4_\
port_a_rx_tx_cnt_1rx_1tx_thruasm_zipper_hsmc_a_alst4.hdl.0.alst4.gz\
:../../core/artifacts:../../artifacts
```

Recommended Library Path for Stratix IV GX230/Zipper in HSMC B

rx
```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic_fir\
_real_alst4_alst4_zipper_hsmc_alst4_port_b_rx_cnt_1rx_0tx_thruasm_zipper_hsmc_b_alst4.hdl.0.alst4.gz\
:../../../core/artifacts:../../artifacts
```

tx
```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int\
_alst4_alst4_zipper_hsmc_alst4_port_b_tx_cnt_0rx_1tx_thruasm_zipper_hsmc_b_alst4.hdl.0.alst4.gz\
:../../../core/artifacts:../../artifacts
```

txrx/bbloopback
```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.fsk_modem_alst4_alst4_zipper_hsmc_alst4_\
port_b_rx_tx_cnt_1rx_1tx_thruasm_zipper_hsmc_b_alst4.hdl.0.alst4.gz\
:../../../core/artifacts:../../artifacts
```

Recommended Library Path for ML605/Zipper in FMC LPC

rx
```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic\
_fir_real_ml605_ml605_zipper_fmc_lpc_rx_cnt_1rx_0tx_thruasm_zipper_lpc_ml605.hdl.0.ml605.gz\
:../../../core/artifacts:../../artifacts
```

tx
```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.fsk_modem_ml605_ml605\
_zipper_fmc_lpc_rx_tx_cnt_1rx_1tx_thruasm_zipper_lpc_ml605.hdl.0.ml605.gz\
:../../../core/artifacts:../../artifacts
```

txrx/bbloopback
```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.fsk_modem_ml605_ml605\
_zipper_fmc_lpc_rx_tx_cnt_1rx_1tx_thruasm_zipper_lpc_ml605.hdl.0.ml605.gz\
:../../../core/artifacts:../../artifacts
```

Example ML605/Zipper in FMC HPC

rx

```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic\
_fir_real_ml605_ml605_zipper_fmc_hpc_rx_cnt_1rx_0tx_thruasm_zipper_hpc_ml605.hdl.0.ml605.gz\
:../../../core/artifacts:../../artifacts
```

tx

```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int\
_ml605_ml605_zipper_fmc_hpc_tx_cnt_0rx_1tx_thruasm_zipper_hpc_ml605.hdl.0.ml605.gz\
:../../../core/artifacts:../../artifacts
```

txrx/bbloopback

```
OCPI_LIBRARY_PATH=../../artifacts/ocpi.assets.fsk_modem_ml605_ml605\
_zipper_fmc_hpc_rx_tx_cnt_1rx_1tx_thruasm_zipper_hpc_ml605.hdl.0.ml605.gz\
:../../../core/artifacts:../../artifacts
```

C.5 Artifacts: Zedboard/Zipper

filerw (zipper not required)

- fsk_filerw_zed_base.bitz
- target-xilinx13_3/file_read.so
- target-xilinx13_3/Baudtracking_simple.so
- target-xilinx13_3/real_digitizer.so
- target-xilinx13_3/file_write.so

rx

- dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic_fir_real_zed_base_cnt_1rx_0tx_thruasm_zipper_lpc_zed.bitz

- target-xilinx13_3/Baudtracking_simple.so
- target-xilinx13_3/real_digitizer.so
- target-xilinx13_3/file_write.so
- target-xilinx13_3/zipper_rx.so
- target-xilinx13_3/lime_rx_proxy.so
- target-xilinx13_3/si5351_proxy.so

tx

- mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int_zed_base_cnt_0rx_1tx_thruasm_zipper_lpc_zed.bitz

- target-xilinx13_3/file_read.so
- target-xilinx13_3/zipper_tx.so
- target-xilinx13_3/lime_tx_proxy.so
- target-xilinx13_3/si5351_proxy.so

txrx/bbloopback

- fsk_modem_zed_base_cnt_1rx_1tx_thruasm_zipper_lpc_zed.bitz

- target-xilinx13_3/file_read.so
- target-xilinx13_3/Baudtracking_simple.so
- target-xilinx13_3/real_digitizer.so
- target-xilinx13_3/file_write.so
- target-xilinx13_3/zipper_rx.so
- target-xilinx13_3/zipper_tx.so
- target-xilinx13_3/lime_rx_proxy.so
- target-xilinx13_3/lime_tx_proxy.so
- target-xilinx13_3/si5351_proxy.so

C.6 Artifacts: Stratix IV/Zipper

filerw (zipper not required)

- fsk_filerw_alst4_base.bitz
- target-centos7/file_read.so
- target-centos7/Baudtracking_simple.so
- target-centos7/real_digitizer.so
- target-centos7/file_write.so

rx

- target-centos7/file_read.so
- target-centos7/Baudtracking_simple.so
- target-centos7/real_digitizer.so
- target-centos7/zipper_rx.so
- target-centos7/lime_rx_proxy.so
- target-centos7/si5351_proxy.so

For Zipper plugged into HSMC Port A:

- dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic_fir_real_alst4_alst4_zipper_hsmc_alst4_port_a_rx_cnt_1rx_0tx_thruasm_zipper_hsmc_a_alst4.bitz

For Zipper plugged into HSMC Port B:

- dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic_fir_real_alst4_alst4_zipper_hsmc_alst4_port_b_rx_cnt_1rx_0tx_thruasm_zipper_hsmc_b_alst4.bitz

tx

- target-centos7/file_read.so
- target-centos7/zipper_tx.so
- target-centos7/lime_tx_proxy.so
- target-centos7/si5351_proxy.so

For Zipper plugged into HSMC Port A:

- mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int_alst4_alst4_zipper_hsmc_alst4_port_a_txcnt_0rx_1tx_thruasm_zipper_hsmc_a_alst4.bitz

For Zipper plugged into HSMC Port B:

- mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int_alst4_alst4_zipper_hsmc_alst4_port_b_tx_cnt_0rx_1tx_thruasm_zipper_hsmc_b_alst4.bitz

txrx/bbloopback

- target-centos7/file_read.so
- target-centos7/Baudtracking_simple.so
- target-centos7/real_digitizer.so
- target-centos7/zipper_rx.so
- target-centos7/zipper_tx.so
- target-centos7/lime_rx_proxy.so
- target-centos7/lime_tx_proxy.so
- target-centos7/si5351_proxy.so

For Zipper plugged into HSMC Port A:

- fsk_modem_alst4_alst4_zipper_hsmc_alst4_port_a_rx_tx_cnt_1rx_1tx_thruasm_zipper_hsmc_a_alst4.bitz

For Zipper plugged into HSMC Port B:

- fsk_modem_alst4_alst4_zipper_hsmc_alst4_port_b_rx_tx_cnt_1rx_1tx_thruasm_zipper_hsmc_b_alst4.bitz

C.7 Artifacts: ML605/Zipper

filerw (zipper not required)

- fsk_filerw_ml605_base.bitz
- target-centos7/file_read.so
- target-centos7/Baudtracking_simple.so
- target-centos7/real_digitizer.so
- target-centos7/file_write.so

rx

- target-centos7/file_read.so
- target-centos7/Baudtracking_simple.so
- target-centos7/real_digitizer.so
- target-centos7/zipper_rx.so
- target-centos7/lime_rx_proxy.so
- target-centos7/si5351_proxy.so

For Zipper plugged into FMC LPC:

- dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic_fir_real_ml605_cfg_1rx_0tx_fmcomms_2_3_lpc_lvds_cnt_1rx_0tx_thruasm_fmcomms_2_3_lpc_LVDS_ml605.bitz

For Zipper plugged into FMC HPC:

- dc_offset_iq_imbalance_mixer_cic_dec_rp_cordic_fir_real_ml605_cfg_1rx_0tx_fmcomms_2_3_hpc_lvds_cnt_1rx_0tx_thruasm_fmcomms_2_3_hpc_LVDS_ml605.bitz

tx

- target-centos7/file_read.so
- target-centos7/zipper_tx.so
- target-centos7/lime_tx_proxy.so
- target-centos7/si5351_proxy.so

For Zipper plugged into FMC LPC:

- mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int_ml605_ml605_zipper_fmc_lpc_tx_cnt_0rx_1tx_thruasm_zipper_lpc_ml605.bitz

For Zipper plugged into FMC HPC:

- mfsk2_zp16_fir_real_phase_to_amp_cordic_cic_int_ml605_ml605_zipper_fmc_hpc_tx_cnt_0rx_1tx_thruasm_zipper_hpc_ml605.bitz

txrx/bbloopback

- target-centos7/file_read.so
- target-centos7/Baudtracking_simple.so
- target-centos7/real_digitizer.so
- target-centos7/zipper_rx.so
- target-centos7/zipper_tx.so
- target-centos7/lime_rx_proxy.so
- target-centos7/lime_tx_proxy.so
- target-centos7/si5351_proxy.so

For Zipper plugged into FMC LPC:

- fsk_modem_ml605_ml605_zipper_fmc_hpc_rx_tx_cnt_1rx_1tx_thruasm_zipper_lpc_ml605.bitz

For Zipper plugged into FMC HPC:

- fsk_modem_ml605_ml605_zipper_fmc_hpc_rx_tx_cnt_1rx_1tx_thruasm_zipper_hpc_ml605.bitz