SigFPGA™
Hardware and Software
Quick Start Guide

Red Rapids

797 North Grove Rd, Suite 101
Richardson, TX 75081
Phone: 972-671-9570
www.redrapids.com

Red Rapids reserves the right to alter product specifications or discontinue any product without notice. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment. This product is not designed, authorized, or warranted for use in a life-support system or other critical application.

All trademark and registered trademarks are the property of their respective owners.

Table of Contents

List of Figures

1.0 Introduction

The SigFPGA™ product family provides the ideal platform to rapidly field application specific signal acquisition and generation functions minus the expense of custom hardware development. The products share a common FPGA processing architecture and code base with different interface options tailored to a variety of market needs.

The latest product documentation and software is available for download from the Red Rapids web site (www.redrapids.com).

1.1 Model Number Options Decoder

The SigFPGA™ product family offers several analog interface options coupled to a common digital signal processing architecture. Each product is assigned a six-digit model number (Model XXX-YYY) that uniquely identifies the specific hardware features of that unit. The first three digits are the primary designator used to convey information about the structure of the analog front-end. The last three digits define the build options selected by a customer to tailor the product to a specific application. Consult the SigFPGA™ Model Number Options Decoder manual (REF-007-000-Rxx) for a current list.

1.2 Conventions

This manual uses the following conventions:

- Hexadecimal numbers are prefixed by “0x” (e.g. 0x00058C).
- Blue font is used for names of directories, files and OS commands.
- Green font is used to designate source code.

☞	Text in this format highlights useful or important information.

!	Text shown in this format is a warning. It describes a situation that could potentially damage your equipment. Please read each warning carefully.

The following are some of the acronyms used in this manual.

- **ADC** Analog to Digital Converter
- **API** Application Program Interface
- **CCXMC** Conduction Cooled Express Mezzanine Card
- **CLK** External Clock
- **DAC** Digital to Analog Converter
- **GPIO** General Purpose I/O
- **LED** Light Emitting Diode
- **PCIe** Peripheral Component Interconnect Express
- **REF** External 10 MHz Clock Reference
- **RX** Receiver
- **TX** Transmitter
- **TRIG** External Trigger
- **XMC** Express Mezzanine Card

1.3 Revision History

Version	Date	Description
R02	10/01/2018	Updated to reflect new SDK.
R01	2/28/2018	Software archives don't have to be collocated on Linux hosts.
R00	2/1/2018	Initial release.

2.0 Hardware Installation

Every SigFPGA™ product is available in both XMC and PCI Express (PCIe) half-length form factors. The hardware ships with a x8 physical and x8 electrical PCIe connector. It is possible to load alternate firmware that reduces the number of electrical connections if desired.

2.1 XMC Hardware Installation

The following instructions provide a general guide for mounting a XMC module to a baseboard or carrier host socket. Consult your baseboard documentation for product specific guidance.

> **!** This is a static sensitive electronic device; please follow standard ESD guidelines when installing the device.

Power Down: Power to the host must be off during hardware installation. Permanent damage may result if the card is plugged into a hot socket.

Identify Socket: Find an open XMC socket on the host. The socket may be keyed for legacy PMC operation. The key consists of a metal post located at the centerline of the socket between the electrical connectors. The position of the post will indicate if the legacy PCI bus is designed for 3.3 Volt or 5 Volt signaling. The SigFPGA™ XMC product does not support legacy PCI operation, but the card is designed to accommodate a 3.3 Volt key for convenience.

Insert Card: Angle the XMC bezel into the host front panel cut-out. The EMI gasket around the bezel may offer resistance, be careful not to dislodge the gasket from the groove. Gently press the XMC down onto the host until the connectors mate.

Secure Card: Secure the XMC hardware with four mounting screws (provided). Be sure that the card does not bow as the screws are fastened.

Check Obstructions: Verify that the card is securely mounted in the XMC socket and not in contact with other components on the host.

Boot Computer: The host will detect the presence of new hardware on the PCIe bus when power is applied. Load the driver for your operating system before attempting to communicate with the device.

> ☞ The expansion card can pick up electrical interference from other devices or directly through the host power supply. Try moving the card away from other devices or try a different host platform if you are experiencing interference.

2.2 PCIe Hardware Installation

The following instructions provide a general guide for PCIe expansion card installation. Consult your host documentation for product specific guidance.

> **!** This is a static sensitive electronic device; please follow standard ESD guidelines when installing the device.

Power Down: Power to the host must be off during hardware installation. Permanent damage may result if the card is plugged into a hot socket.

Identify Slot: Find an open PCIe slot on the host backplane and verify that the socket is compatible with the card you are installing. The PCI Express specification defines each

slot by a physical size and electrical size. It is not uncommon to find a socket electrically wired to fewer lanes than the physical size allows. The specific configuration is usually silkscreened on the motherboard.

The interoperability of PCIe cards and slots is summarized in Table 2-1. The SigFPGA™ card ships with a x8 physical connector.

Table 2-1 PCIe Card Interoperability

Slot / Card	x1	x4	x8	x16
x1	Required	Required	Required	Required
x4	No	Required	Allowed	Allowed
x8	No	No	Required	Allowed
x16	No	No	No	Required

Remove Metal Insert: The expansion slot opening will typically be covered by a metal insert held in with a single screw. The metal insert may have to be punched out of some cases. Unscrew or punch out the appropriate metal insert. It may help to align the expansion card over the slot to determine which insert to remove.

Insert Card: Align the expansion card edge connector with the PCIe slot making sure the bottom edge of the metal faceplate clears the edge of the backplane motherboard. Apply firm pressure to seat the card in the slot. You may need to rock the card slightly from front to back to get the unit seated properly. *Do not force the card or significantly flex the motherboard*. The expansion board should not require much force to insert provided everything is lined up correctly.

Secure Card: The top side of the faceplate should be flush or close to flush with the card retention bar. If the plate is not close to the bar, verify that the board is not canted in the slot. Secure the expansion card to the chassis by inserting a screw into the top of the metal faceplate.

Check Obstructions: Verify that the card is securely mounted in the PCIe slot and not in contact with other items inside the chassis.

Boot Computer: The host will detect the presence of new hardware on the PCIe bus when power is applied. Load the driver for your operating system before attempting to communicate with the device.

The expansion card can pick up electrical interference from other devices or directly through the host power supply. Try moving the card away from other devices or try a different host platform if you are experiencing interference.

3.0 Front Panel Connectors

The face of each SigFPGA™ product contains an array of coaxial and digital connectors to connect signals, clocks and external triggers. If a bezel or face plate is present, each connector will be marked with one of the following designators:

RX#: Analog receiver channel number # input.

TX#: Analog transmitter channel number # output.

GPIO: General purpose I/O connector for external triggers.

REF/CLK: External 10 MHz reference or sample clock input.

TRIG: Coaxial trigger input or 1 PPS timing input.

There are two light emitting diodes (LEDs) located on either side of the GPIO connector. They can be illuminated by writing to BAR2 address x0020 from software. Writing a one to bit 0 will illuminate the yellow LED and writing a one to bit 1 will illuminate the green LED.

The connector placement and LED location for each product is shown below. The three available form factors are illustrated for each product.

3.1 Model 371

The Model 371 front panel includes four coaxial connectors and one GPIO connector as shown in Figure 3-1. The coaxial receiver channel inputs are numbered RX1 and RX2. The remaining two coaxial connectors are the REF/CLK input and TRIG input.

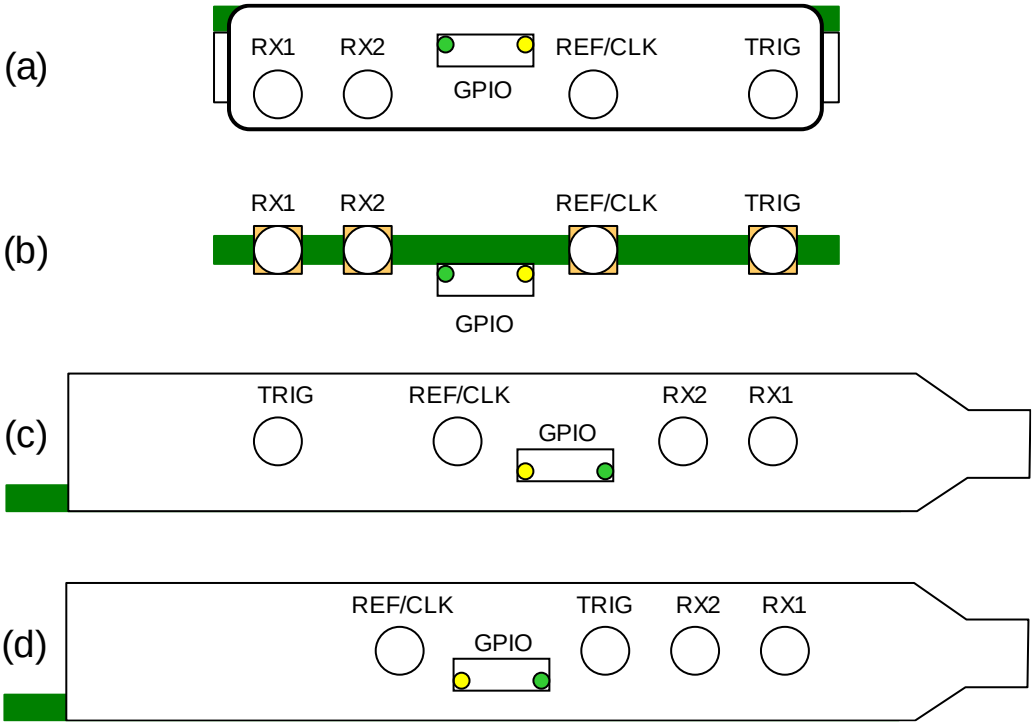

Figure 3-1 Model 371 (a) XMC, (b) CCXMC, (c/d) PCIe Front Panel

3.2 Model 372

The Model 372 front panel includes five coaxial connectors and one GPIO connector as shown in Figure 3-2. The coaxial receiver channel inputs are numbered RX1 and RX2, while the transmitter outputs are numbered TX3 and TX4. The fifth coaxial connector can either be used as a REF/CLK input or a TRIG input.

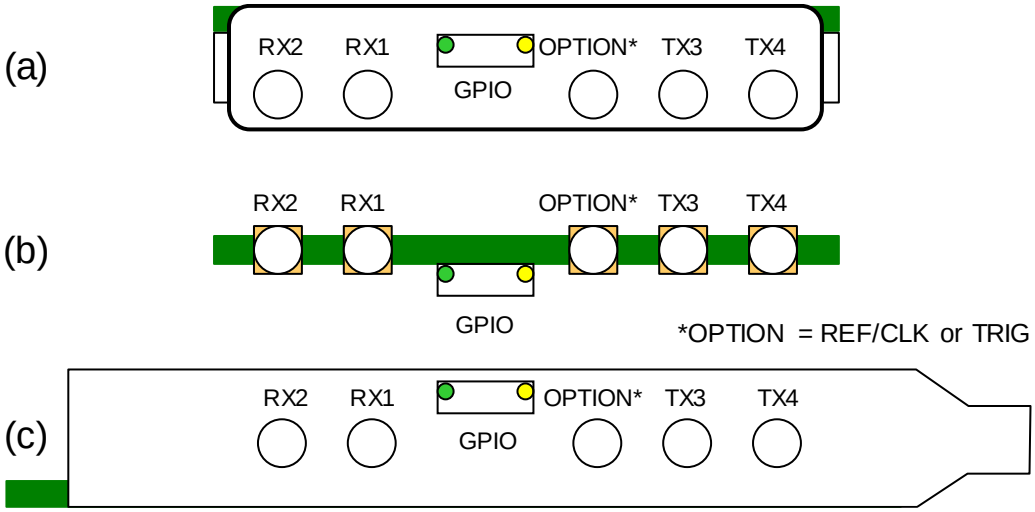

Figure 3-2 Model 372 (a) XMC, (b) CCXMC, (c) PCIe Front Panel

3.3 Model 373

The Model 373 front panel includes four coaxial connectors and one GPIO connector as shown in Figure 3-3. The coaxial receiver channel inputs are numbered RX1 and RX2. The remaining two coaxial connectors are the REF/CLK input and TRIG input.

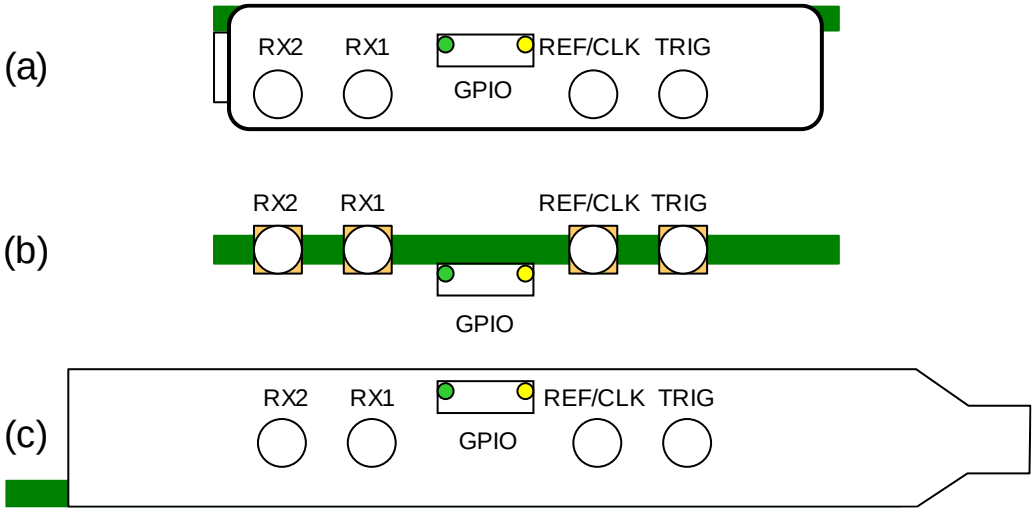

Figure 3-3 Model 373 (a) XMC, (b) CCXMC, (c) PCIe Front Panel

3.4 Model 374

The Model 374 front panel includes four coaxial connectors and one GPIO connector as shown in Figure 3-4. The coaxial transmitter channel outputs are numbered TX1 and TX2. The remaining two coaxial connectors are the REF/CLK input and TRIG input.

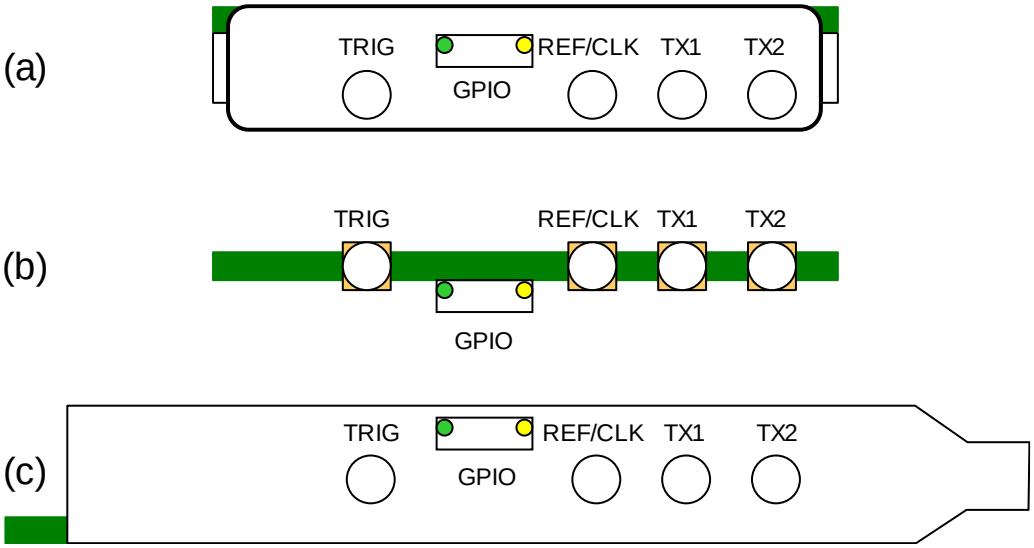

Figure 3-4 Model 374 (a) XMC, (b) CCXMC, (c) PCIe Front Panel

3.5 Model 376

The Model 376 front panel includes four coaxial connectors and one GPIO connector as shown in Figure 3-5. The coaxial receiver channel inputs are numbered RX1 and RX2. The remaining two coaxial connectors are the REF/CLK input and TRIG input.

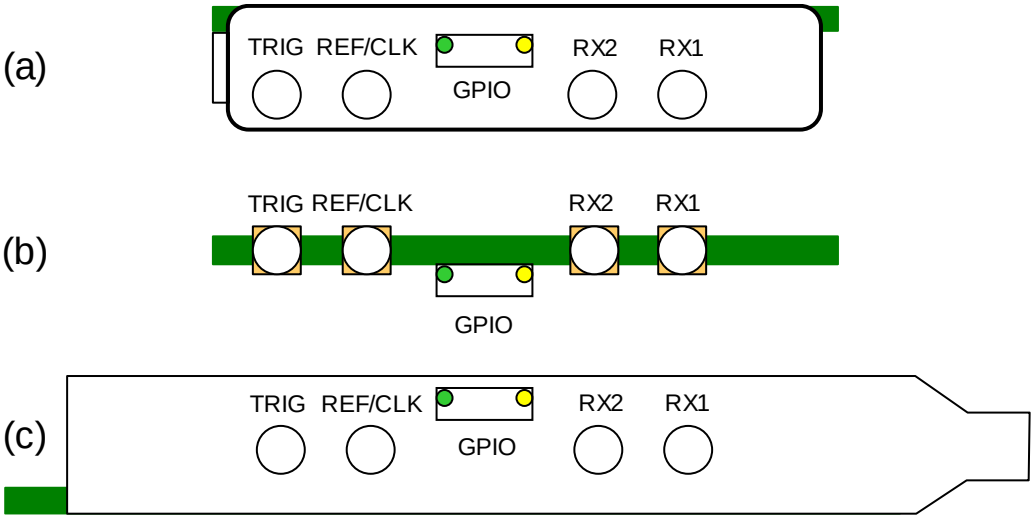

Figure 3-5 Model 376 (a) XMC, (b) CCXMC, (c) PCIe Front Panel

3.6 Model 377

The Model 377 front panel includes five or six coaxial connectors and one GPIO connector as shown in Figure 3-6. The coaxial receiver channel inputs are consecutively numbered RX1 through RX4. Units equipped with six coaxial connectors have dedicated REF/CLK and TRIG inputs, otherwise the fifth coaxial connector can perform either function.

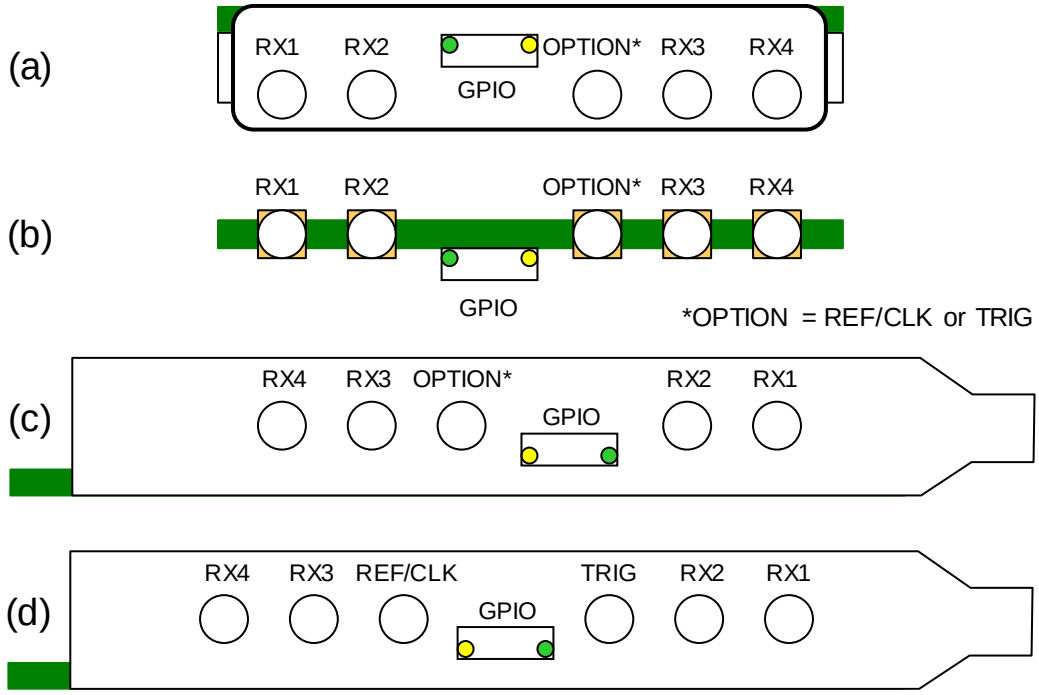

Figure 3-6 Model 377 (a) XMC, (b) CCXMC, (c/d) PCIe Front Panel

3.7 Model 378

The Model 378 front panel includes nine coaxial connectors and one GPIO connector as shown in Figure 3-6. The coaxial receiver channel inputs are consecutively numbered RX1 through RX8. The ninth coaxial connector can either be used as a REF/CLK input or a TRIG input.

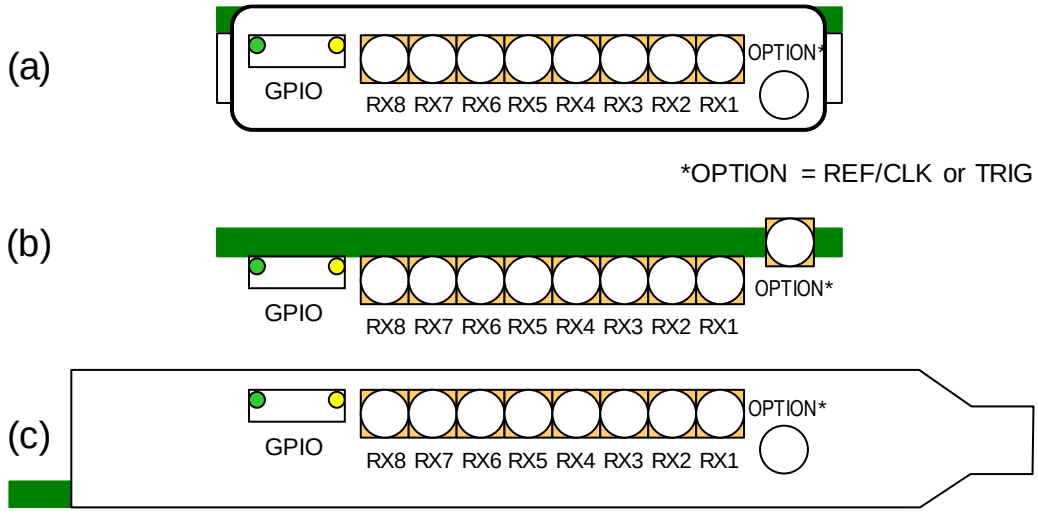

Figure 3-7 Model 378 (a) XMC, (b) CCXMC, (c) PCIe Front Panel

4.0 Software Installation

The SigFPGA™ software development kit for Windows or Linux can be download from the Red Rapids website (www.redrapids.com). There is a separate distribution provided for each operating system as listed below:

SDK-370-001-Rxx: Windows SDK Distribution (Zip Archive)

SDK-370-002-Rxx: Linux SDK Distribution (Tar Archive)

The distribution number of each archive is also the name of the root directory created when it is extracted. The archive can be extracted to any working directory on the host computer and the name of the root directory can be changed without impacting operation.

The driver must be installed on the host computer before any application will execute. A Red Rapids device must be present on the bus before the driver will load.

4.1 Windows Software Development Kit (SDK-370-001-Rxx)

The Windows software and API are distributed on Red Rapids zip archive number SDK-370-001-Rxx. The directory structure of the archive is outlined below:

\

The root directory includes four subdirectories containing relevant documentation (docs), the driver software (driver), a product demonstration application (sigfpga), and flash programmer (flash) to install firmware updates.

\docs

The docs subdirectory contains the following three documents associated with the software distribution:

REF-007-001-Rxx: SigFPGA Quick Start Guide

REF-806-902-Rxx: RRadapter Device Driver and API Reference Manual

REF-005-003-Rxx: Application Programming Interface (API) Reference Manual

\driver

The driver subdirectory contains the Windows driver (RRadapter) that must be installed on the host computer to communicate with a Red Rapids SigFPGA™ device. Document number REF-806-905-Rxx provides installation instructions and a description of the low-level API.

\sigfpga

The sigfpga subdirectory contains an application example that demonstrates all product API functions. The VSbuild.bat batch file can be executed from a Visual Studio command prompt to build a 32-bit or 64-bit Windows executable from the sigfpga.c source code. A precompiled executable is included in the distribution under the exe-32bit and exe-64bit folders to immediately run the demonstration.

\sigfpga\exe-32bit

The exe-32bit folder is the repository for library and executable files generated when the VSbuild.bat batch file is run from a 32-bit operating system. The two RRadapter driver libraries are also copied to this folder so that they can be found by sigfpa.exe at runtime. Any files generated by the application during demonstration will be created in this folder.

\sigfpga\exe-64bit

The exe-64bit folder is the repository for library and executable files generated when the VSbuild.bat batch file is run from a 64-bit operating system. The two RRadapter driver libraries are also copied to this folder so that they can be found by sigfpa.exe at runtime.

\sigfpga\include

The include folder contains the header files used to access the SigFPGA™ API.

sigfpga\obj

The obj folder is the repository for object files generated when the VSbuild.bat batch file is executed.

\sigfpga\src

The src folder contains source code to all SigFPGA™ API functions.

\flash

The flash subdirectory contains a software programmer that is used to load product firmware updates. The VSbuild.bat batch file can be executed from a Visual Studio command prompt to build a 32-bit or 64-bit Windows executable from the flash.c source code. A precompiled executable is included in the distribution under the exe-32bit and exe-64bit folders to immediately run the programmer.

\flash\exe-32bit

The exe-32bit folder is the repository for library and executable files generated when the VSbuild.bat batch file is run from a 32-bit operating system. The two RRadapter driver libraries are also copied to this folder so that they can be found by flash.exe at runtime. Any firmware file (MCS format) that will be accessed by the application should be copied to this folder.

\flash\exe-64bit

The exe-64bit folder is the repository for library and executable files generated when the VSbuild.bat batch file is run from a 32-bit operating system. The two RRadapter driver libraries are also copied to this folder so that they can be found by flash.exe at runtime. Any firmware file (MCS format) that will be accessed by the application should be copied to this folder.

\flash\include

The include folder contains the header files used to access the SigFPGA™ API.

\flash\obj

The obj folder is the repository for object files generated when the VSbuild.bat batch file is executed.

\flash\src

The src folder contains source code to the programmer functions.

4.2 Linux Software Development Kit (SDK-370-002-Rxx)

The Linux software and API are distributed on Red Rapids tape archive (tar) number SDK-370-002-Rxx. The directory structure of the archive is outlined below:

\

The root directory includes four subdirectories containing relevant documentation (docs), the driver software (driver), a product demonstration application (sigfpga), and flash programmer (flash) to install firmware updates.

\docs

The docs subdirectory contains the following three documents associated with the software distribution:

REF-007-001-Rxx: SigFPGA Quick Start Guide

REF-806-910-Rxx: Linux Device Drivers and API Reference Manual

REF-005-003-Rxx: Application Programming Interface (API) Reference Manual

\driver

The driver subdirectory contains the Linux device drivers (rrdev and rrdma) that must be installed on the host computer to communicate with a Red Rapids SigFPGA™ device. Document number REF-806-910-Rxx provides installation instructions.

\sigfpga

The sigfpga subdirectory contains an application example that demonstrates all product API functions. A Makefile is provided to build a 32-bit or 64-bit binary from the sigfpga.c source code. The Makefile must be executed before the demonstration can be run.

\sigfpga\bin

The bin folder is the repository for the binary file generated when the Makefile is run. Any files generated by the application during demonstration will be created in this folder.

\sigfpga\include

The include folder contains the header file used to access the SigFPGA™ API.

sigfpga\obj

The obj folder is the repository for object files generated when the Makefile is executed.

\sigfpga\src

The src folder contains source code to all SigFPGA™ API functions.

\flash

The flash subdirectory contains a software programmer that is used to load product firmware updates. A Makefile is provided to build a 32-bit or 64-bit binary from the flash.c source code. The Makefile must be executed before the programmer can be run.

\flash\bin

The bin folder is the repository for the binary file generated when the Makefile is run. Any firmware file (MCS format) that will be accessed by the application should be copied to this folder.

\flash\include

The include folder contains the header file used to access the SigFPGA™ API.

\flash\obj

The obj folder is the repository for object files generated when the VSbuild.bat batch file is executed.

\flash\src

The src folder contains source code to the programmer functions.

5.0 Product Demonstration Application (sigfpga)

Each SigFPGA™ product is accompanied by a software application that demonstrates hardware functionality using calls to the supplied API. Products equipped with receiver channels will digitize an analog signal, transfer samples to host memory using DMA transactions, and store a snapshot of the collection to a text file. Products equipped with transmitter channels will retrieve digital samples from host memory via DMA transactions and produce an analog reproduction of the digital signal. The transmitter samples are precomputed to produce a sinusoid of frequency Fs/8.

The demonstration software is provided in the SDK distribution. The application can be run from a command prompt by executing the following instructions from the /sigfpga directory.

32-bit Windows	64-bit Windows	32-bit Linux	64-bit Linux
> cd exe-32bit > sigfpga	> cd exe-64bit > sigfpga	> make > cd bin > ./sigfpga	> make > cd bin > ./sigfpga

5.1 Receiver Demonstration

Figure 5-1 illustrates the hardware path exercised by the receiver demonstration software. The application enables all available channels and issues a software trigger to initiate signal acquisition.

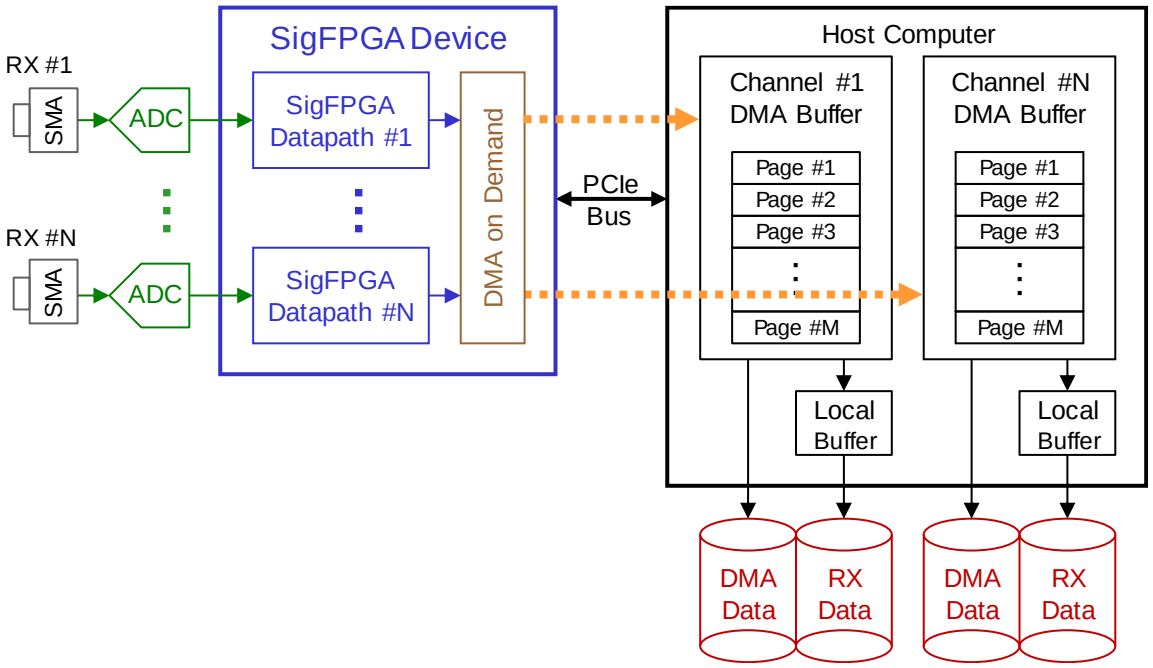

Figure 5-1 Receiver Data Flow

The SigFPGA™ datapath processes the number of samples requested by the channel configuration. All channels are configured identically for demonstration purposes.

Each channel continuously writes digital data samples to a dedicated DMA buffer in host memory. The DMA transactions are mastered by the SigFPGA™ device, no software

intervention is required. Consult the DMA on Demand Operating Guide (REF-004-000-Rxx) for further details about DMA transactions. The application software continuously transfers the contents of each DMA buffer to another local memory buffer to demonstrate how a customer application would manage the incoming data.

The demonstration will create two output files per channel. The first set of files are named dmabuf#–#.txt, where the first # is the device number and the second # is the receiver channel number. This file contains the first page of sample data in the DMA buffer. The number of pages that are written to the file can be changed in the call to the DMASave() API function, but must not be any greater than the total number of pages in the DMA buffer. The second set of files are named rxdata#–#.txt, where the first # is the device number and the second # is the receiver channel number. This file contains the first page of sample data stored in the local buffer. The number of pages written can be increased, but must not be any greater than the number of pages between DMA markers. The data in the rxdata#–#.txt and dmabuf#–#.txt files will only match if processing terminated while writing to a DMA page number between the first and second DMA marker.

No external equipment is required to run the demonstration code, but the receiver channels will only collect noise samples if there is no input signal present. Each SigFPGA™ product is assigned an analog Front End Reference Manual (REF-000-YYY-Rxx) that contains information about the expected input signal power.

5.2 Transmitter Demonstration

Figure 5-2 illustrates the hardware path exercised by the transmitter demonstration software. The application enables all available channels and issues a software trigger to initiate signal generation.

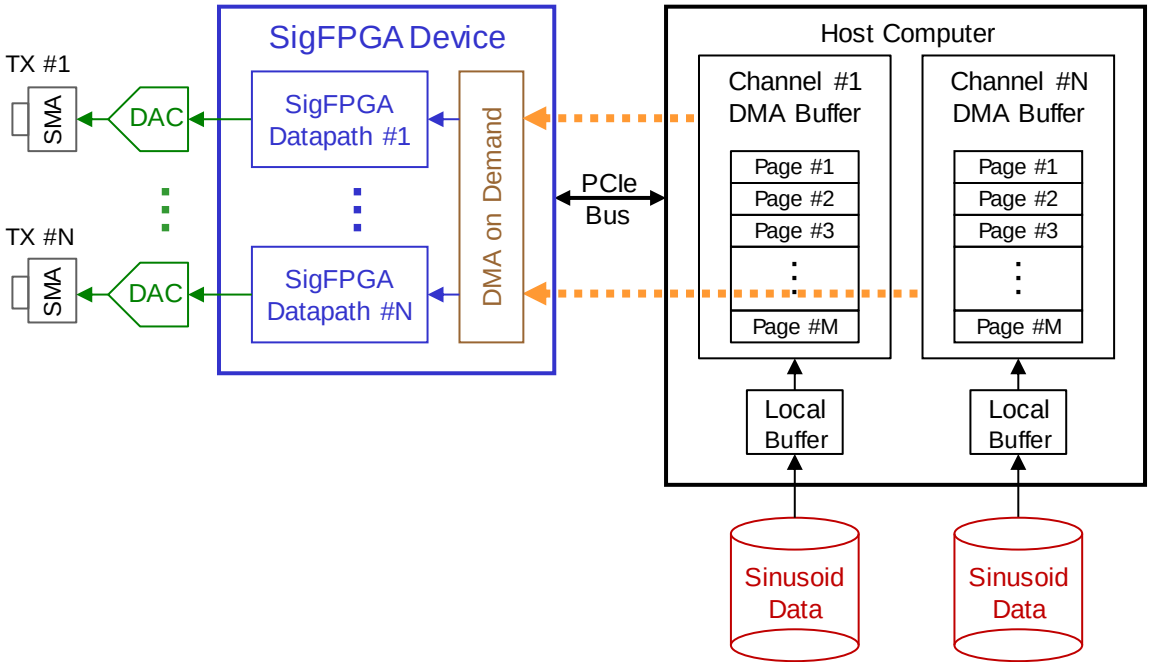

Figure 5-2 Transmitter Data Flow

The SigFPGA™ datapath processes the number of samples requested by the channel configuration. All channels are configured identically for demonstration purposes.

The demonstration software first generates sinusoid data samples that are stored in a set of files named txdata#–#.txt, where the first # is the device number and the second # is the transmitter channel number. The frequency of the sinusoid is set to Fs/8 by default, where Fs is the sample clock frequency produced by the on-board synthesizer or supplied as an external clock. These samples are loaded into a local memory buffer before the channels are enabled.

After processing begins, the application continuously transfers the contents of the local buffer to a DMA buffer to demonstrate how a customer application would supply outgoing data. The SigFPGA™ device continuously reads samples from the DMA buffer in host memory and supplies them to the datapath. The DMA transactions are mastered by the SigFPGA™ device, no software intervention is required. Consult the DMA on Demand Operating Guide (REF-004-000-Rxx) for further details about DMA transactions.

It is important to note that the waveform stored in the local buffer does not get updated once channel processing begins. The same block of samples will be repeated through the datapath if the requested number of samples exceeds the size of the local buffer. This will result in phase discontinuities if the final waveform sample does not represent the end of a period.

No external equipment is required to run the demonstration code, but a spectrum analyzer can be used to monitor the output waveform of each channel. Each SigFPGA™ product is assigned an analog Front End Reference Manual (REF-000-yyy-Rxx) that contains information about the expected output signal power.

5.3 Transceiver Demonstration

SigFPGA™ products that are equipped with both receiver and transmitter channels will run both demonstrations concurrently. A single software trigger will initiate processing on all channels simultaneously, regardless of direction.

5.4 Runtime Messages

The SigFPGA™ application reports status, warning, and error messages to the screen as it executes. Appendix 7.0 contains an example report that was generated by a system equipped with a transceiver device.

5.4.1 Open Device

Three messages are generated when a hardware device is successfully opened by the application.

(1) Firmware revision reported by the hardware.

(2) Model number reported by the hardware.

(3) Clock status.

The first two messages simply convey information to identify the hardware. The third message reports the health of the clock network and the specific clock source detected by the hardware. This message will change depending on whether there is an external reference or external sample clock connected through the front panel. An error will be reported if any type of clock problem is detected.

5.4.2 Assign DMA Buffers

The assignment of DMA buffers to RX and TX channels will only produce a message if an error is encountered, usually caused by an invalid configuration setting. The most

serious error occurs when the operating system encounters a problem allocating the requested buffer size in protected memory.

5.4.3 Load Configuration Settings

Loading configuration settings is normally a silent process since most of the values are simply written to registers within the device. A message may be displayed if there is any problem loading waveform data into the DMA buffer assigned to a TX channel.

5.4.4 QDR II+ SRAM Test (Optional)

If the optional QDR II+ SRAM is present, a test is performed to verify that it is functioning correctly. A unique value is written to each memory address and read back to compare against the expected value. The number of errors detected for each SRAM is reported.

5.4.5 PCIe Performance Measurement

It is important to know whether the host computer can supply the PCIe bus bandwidth required by a SigFPGA™ device. A performance test is run by the application to measure actual performance with all available channels continuously making DMA requests. The number reported by the benchmark must be above the expected throughput of the device to avoid losing data.

The performance benchmark is only provided for convenience, it can be eliminated from the code once the information is collected.

5.4.6 Starting Channels

A message is generated as every channel on the device is started in sequence. Channel processing does not begin immediately, the channels are simply armed and waiting for the requested trigger.

5.4.7 Start/Stop Processing

A software trigger is issued to start processing samples on all channels simultaneously. The application waits silently for each channel to complete the requested number of samples and stop. Terminating the process while the channels are active (CTRL-C) may crash the computer since the hardware is still accessing protected DMA buffer space in host memory.

The demonstration software copies fresh data out of the receiver DMA buffers into local memory while the channel is active. Similarly, fresh data is copied into the transmitter DMA buffers. The DMA buffers can be monitored by either polling the status bits or waiting for an interrupt. The demonstration software defaults to polling.

5.4.8 Retrieve Temperature and Power Status

There are multiple temperature sensors and power monitors available on the SigFPGA™ device. The application queries all available sensors and reports the results when processing is complete. Any temperature above 90 degrees C indicates inadequate cooling for the hardware.

5.4.9 Review Channel Status

There are several error conditions monitored by the device throughout operation. These are reviewed and reported when channel processing completes.

(1) Read or write to an illegal BAR address.

(2) Loss of stable clocks.

(3) ADC over-range.

(4) FIFO overflow (receiver) or underflow (transmitter).

(5) Loss of DMA data.

The first message simply indicates that software issued a PCIe read or write command to a register address that does not exist in the hardware. A clock error would only occur if there were some type of hardware event that causes a clock to lose lock. The ADC over-range occurs when the input signal exceeds the maximum amplitude supported by the device, but it is worth noting that not all ADC chips offer this status. A FIFO error will usually occur when the PCIe bus is not fast enough to meet the throughput demands of all active channels competing for resources. A loss of DMA data indicates that the software servicing the DMA buffers in host memory is not fast enough to keep up with the refresh rate demanded by the hardware. This might be resolved by increasing the size of the DMA buffers or changing the number of pages between markers. However, it is always possible that the host processor is simply not fast enough to keep up with the transfer of data between the device and host.

5.4.10 Save Captured Data (RX Only)

Every receiver channel will produce two text files as discussed in Section 5.1. Both files will contain signed integer values. A real data sample occupies a single line in the file while a complex sample occupies two lines. The real (in-phase) value always precedes the imaginary (quadrature) value of a complex sample. The maximum allowed positive or negative value is dictated by the data item size selected in the channel configuration.

6.0 Firmware Update Application (flash)

Each SigFPGA™ product includes a byte peripheral interface (BPI) flash to store the FPGA configuration bitstream. A detailed description can be found in document REF-002-000-Rxx (BPI Configuration Flash Design Guide).

The configuration Flash can be loaded through the JTAG port located on the GPIO connector of every SigFPGA™ product using a Xilinx programmer. The firmware is supplied in an MCS format file created with the Xilinx Vivado™ tool. This same MCS file can also be used to load the firmware over the PCIe bus using the flash application without a hardware programmer.

The application can be run from a command prompt by executing the following instructions from the /flash directory.

32-bit Windows	64-bit Windows	32-bit Linux	64-bit Linux
> cd exe-32bit > flash [option] [file]	> cd exe-64bit > flash [option] [file]	> make > cd bin > ./flash [option] [file]	> cd exe-64bit > flash [option] [file]

The following is a list of all the options available to the flash application:

flash [-p -g -a1 -a2] [-b -e -u <filename> -v <filename>] [-d <optional device number>]

One of the following four flash segments must be selected:

-p	Apply directive to the primary segment.
-g	Apply directive to the golden segment.
-a1	Apply directive to alternate segment #1.
-a2	Apply directive to alternate segment #2.

One of the following four directives must be selected:

-b	Blank check the entire selected segment.
-e	Erase the entire selected segment.
-u <filename>	Update the selected segment <MCS file required>.
-v <filename>	Verify the selected segment <MCS file required>.

The device number defaults to zero, but can be changed to address additional units:

-d <device number>	Select a target device number other than zero.

The first option will almost always be set to the primary segment (-p) since it is unlikely that a SigFPGA™ application would attempt to use any other part of the flash memory. The second option determines what action is to be performed on the flash. The third option is only used if there is more than one SigFPGA™ product connected to the PCIe bus and a device number other than zero is targeted for programming.

6.1 Blank Check Command

The blank check directive (-b) will simply report whether the flash memory is already programmed or is in the erased state.

Windows: flash -p -b Linux: ./flash -p -b

6.2 Erase Command

The erase directive (-e) will erase any existing programming, returning the flash to a blank state.

Windows: flash -p -e Linux: ./flash -p -e

6.3 Update Command

The update directive (-u) will overwrite any existing flash programming with the firmware supplied in the designated MCS file.

Windows: flash -p -u <filename> Linux: ./flash -p -u <filename>

6.4 Verify Command

The verify directive (-v) will compare the existing flash programming to firmware supplied in the designated MCS file.

Windows: flash -p -v <filename> Linux: ./flash -p -v <filename>

7.0 Appendix A – Example Runtime Reports

The following transcripts were produced by the SigFPGA™ demonstration software running on a computer with a Model 372 transceiver connected to the PCIe bus.

7.1 Example Windows Runtime Report

```
******************************** SigFPGA ********************************

Communicating with Jungo driver...

Initializing device DEV0...
  [INFO] Firmware revision (MMDDYYYY): 01102018
  [INFO] Identified Red Rapids Model 372.
  [INFO] Sample clock locked to internal reference, converter clocks locked.

Assigning DMA buffers to device DEV0...

Loading device DEV0 configuration settings...

Checking device DEV0 hardware performance...
  [INFO] Measured RX PCI performance: 3.218 Gbytes/sec
  [INFO] Measured TX PCI performance: 1.129 Gbytes/sec
  [INFO] Measured RX PCI latency: 7.264 usec
  [INFO] Measured TX PCI latency: 10.516 usec

Starting device DEV0 channels...
  [INFO] Channel #1 enabled.
  [INFO] Channel #2 enabled.
  [INFO] Channel #3 enabled.
  [INFO] Channel #4 enabled.

Waiting for all channels to stop processing (interrupts disabled)...
  [WARNING] CTRL-C FROM THE KEYBOARD MAY CRASH THE COMPUTER!

Retrieving device DEV0 temperature status...
  [INFO] FPGA die temperature is 49.06 degrees Celsius.
  [INFO] ADC proximity temperature is 36.00 degrees Celsius.
  [INFO] QDR proximity temperature is 40.00 degrees Celsius.
  [INFO] Shielded proximity temperature is 44.00 degrees Celsius.
  [INFO] Card center proximity temperature is 40.00 degrees Celsius.

Retrieving device DEV0 power status...
  [INFO] Total power dissipation is 12.84 Watts.
  [INFO] Supply #1 summary: 3.36 Volts, 0.34 Amps, 1.13 Watts
  [INFO] Supply #2 summary: 12.20 Volts, 0.25 Amps, 3.03 Watts
  [INFO] Supply #3 summary: 12.20 Volts, 0.71 Amps, 8.69 Watts

Reviewing device DEV0 channel status...

Saving captured data on all device DEV0 receiver channels...
  [INFO] Writing channel #1 DMA buffer contents to file dmabuf0-1.txt.
  [INFO] Writing channel #1 data buffer contents to file rxdata0-1.txt.
  [INFO] Writing channel #2 DMA buffer contents to file dmabuf0-2.txt.
  [INFO] Writing channel #2 data buffer contents to file rxdata0-2.txt.

Freeing resources allocated to device DEV0...

...all operations completed.
```

7.2 Example Linux Runtime Report

```
******************************** SigFPGA ********************************

Scanning for Red Rapids hardware...

Initializing device DEV0...
  [INFO] Firmware revision (MMDDYYYY): 01102018
  [INFO] Identified Red Rapids Model 372.
  [INFO] Sample clock locked to internal reference, converter clocks locked.

Assigning DMA buffers to device DEV0...

Loading device DEV0 configuration settings...

Checking device DEV0 hardware performance...
  [INFO] Measured RX PCI performance: 3.071 Gbytes/sec
  [INFO] Measured TX PCI performance: 1.022 Gbytes/sec
  [INFO] Measured RX PCI latency: 7.112 usec
  [INFO] Measured TX PCI latency: 10.120 usec

Starting device DEV0 channels...
  [INFO] Channel #1 enabled.
  [INFO] Channel #2 enabled.
  [INFO] Channel #3 enabled.
  [INFO] Channel #4 enabled.

Waiting for all channels to stop processing (interrupts disabled)...
  [WARNING] CTRL-C FROM THE KEYBOARD MAY CRASH THE COMPUTER!

Retrieving device DEV0 temperature status...
  [INFO] FPGA die temperature is 47.08 degrees Celsius.
  [INFO] ADC proximity temperature is 36.00 degrees Celsius.
  [INFO] QDR proximity temperature is 34.00 degrees Celsius.
  [INFO] Shielded proximity temperature is 36.00 degrees Celsius.
  [INFO] Card center proximity temperature is 34.00 degrees Celsius.

Retrieving device DEV0 power status...
  [INFO] Total power dissipation is 12.74 Watts.
  [INFO] Supply #1 summary: 3.36 Volts, 0.34 Amps, 1.13 Watts
  [INFO] Supply #2 summary: 12.19 Volts, 0.25 Amps, 3.02 Watts
  [INFO] Supply #3 summary: 12.20 Volts, 0.70 Amps, 8.59 Watts

Reviewing device DEV0 channel status...

Saving captured data on all device DEV0 receiver channels...
  [INFO] Writing channel #1 DMA buffer contents to file dmabuf0-1.txt.
  [INFO] Writing channel #1 data buffer contents to file rxdata0-1.txt.
  [INFO] Writing channel #2 DMA buffer contents to file dmabuf0-2.txt.
  [INFO] Writing channel #2 data buffer contents to file rxdata0-2.txt.

Freeing resources allocated to device DEV0...

...all operations completed.
```