Zynq UltraScale+ RFSoC ZCU216 RF Data Converter Evaluation Tool

User Guide

UG1433 (v1.0) March 23, 2020

XILINX®

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
03/23/2020 Version 1.0	
Initial release.	N/A

Table of Contents

XILINX

Chapter 1

Introduction

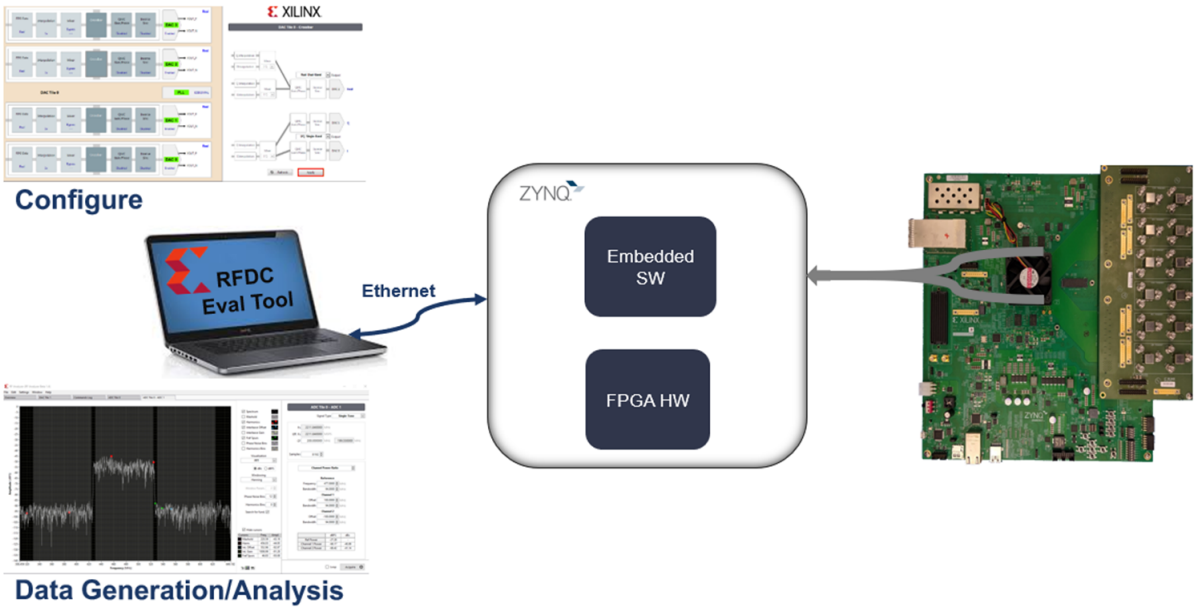
The Zynq® UltraScale+™ RFSoC RF Data Converter (DC) Evaluation Tool and ZCU216 Evaluation Kit is the ideal combination of evaluation software and test platform to facilitate cutting edge application development. The evaluation tool can be used to jump start RF-class analog designs and to demonstrate the capabilities and excellent performance of the Gen 3 RF data converters. Advantages of using this tool include acceleration of the product design cycle, reduction of product go-to-market expense, and quicker revenue realization.

Overview

The evaluation tool provides the means to control the ZCU216 RF DC IP (see *Zynq UltraScale+ RFSoC RF Data Converter LogiCORE IP Product Guide* (PG269)) and associated design from a host computer. The tool allows the exploration of RF configurations, generation and capture of RF data, and observation of key RF metrics.

The following figure shows the evaluation tool and test platform overview.

Figure 1: **Evaluation Tool and Test Platform Overview**



The tool is built from these components:

- *ZCU216 Evaluation Board User Guide* (UG1390)
- FPGA hardware design (see Chapter 3: Hardware Design)
- FPGA embedded software design (see Chapter 4: Software Design and Build)
- GUI (see *RF Data Converter Interface User Guide* (UG1309))
- Protocol used to communicate between the host computer (GUI) and the software design (see Chapter 5: Protocol Specifications)

Installation

See the RF DC Evaluation Tool for ZCU216 Board Quick Start website for installation and folder structure information.

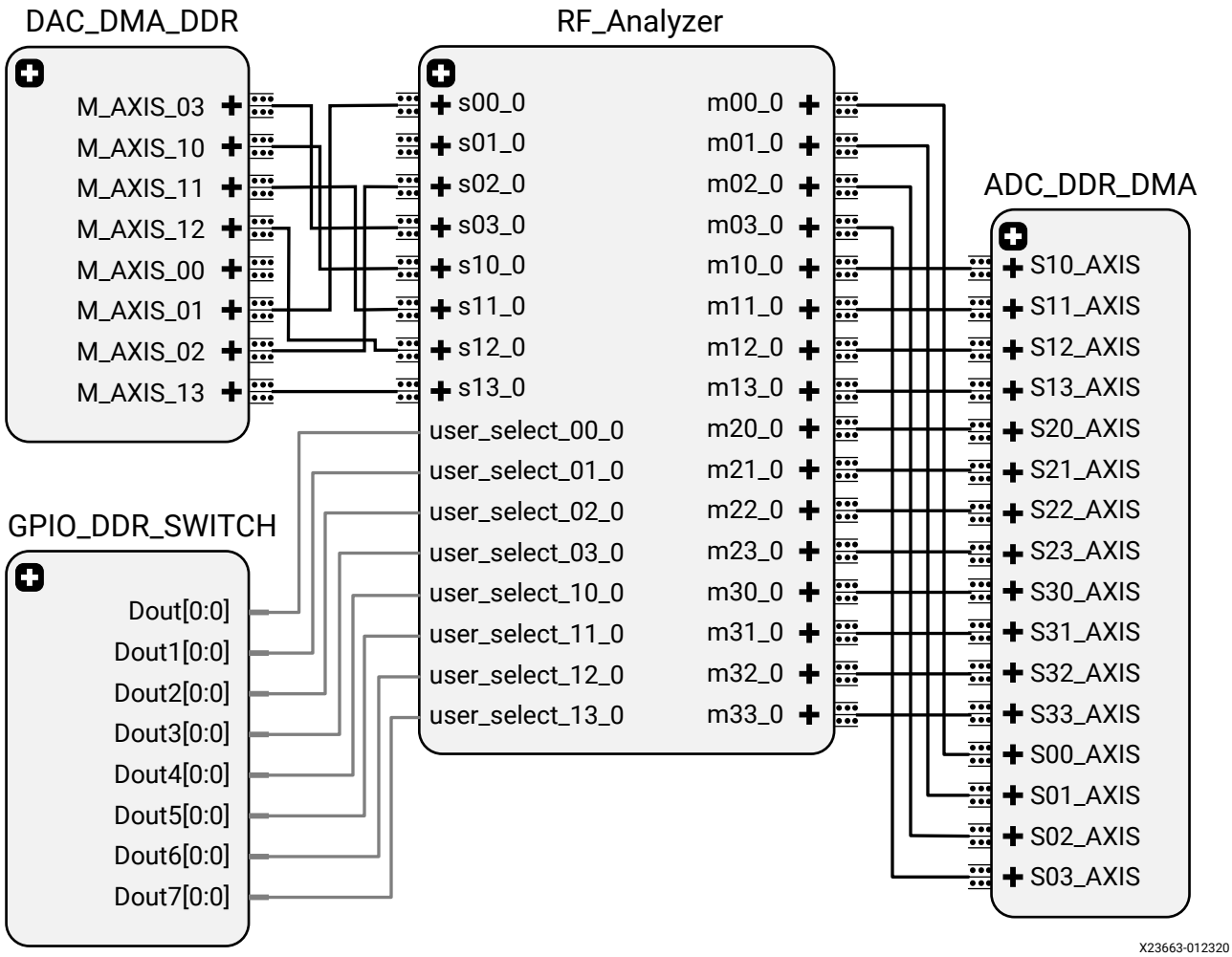
Hardware Design

The Vivado® design suite project used to build this programmable logic (PL) design is located in the install directory in the `pl` folder. The hardware design architecture is based on the RF analyzer architecture (see *Zynq UltraScale+ RFSoC RF Data Converter LogiCORE IP Product Guide* (PG269)). Notable additions to this architecture include:

- MicroBlaze™ is replaced by the Cortex™-A53
- PL DDR4 support, including DMA, AXI4-Stream broadcaster and switch, and GPIO control
- Clocking scheme to support DDR4 and multi-tile synchronization
- I2C connection to control external clocks (CLK104, see *ZCU216 Evaluation Board User Guide* (UG1390))

The following figure shows the high-level hardware architecture.

Figure 2: **High-level Hardware Architecture**



Block RAM Generation and Capture

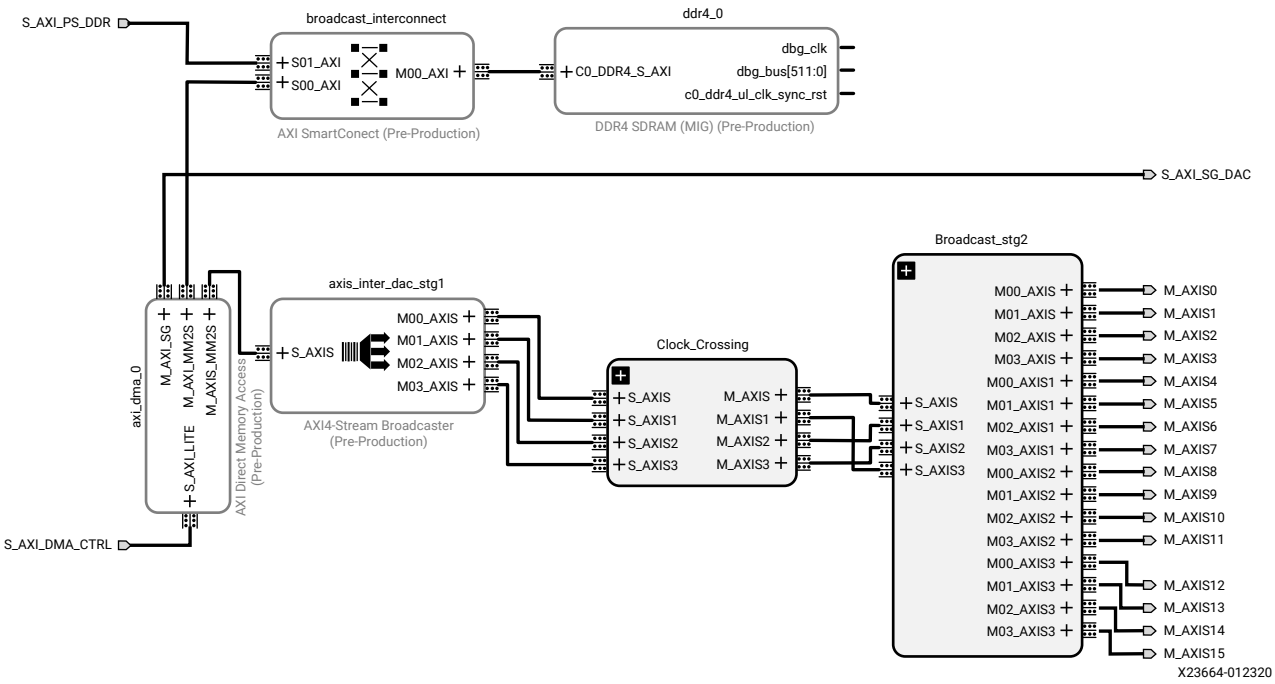
The block RAM generation and capture are described in the "RF Analyzer" section in the *Zynq UltraScale+ RFSoC RF Data Converter LogiCORE IP Product Guide* (PG269).

DAC DDR

A DDR4 memory controller (see *UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide* (PG150)) is instantiated to control the external DDR4 memory. It connects to an AXI DMA controller (see *AXI DMA LogiCORE IP Product Guide* (PG021)). The waveform loaded into the DDR4 memory is then broadcast to the DAC tiles selected by the AXI GPIO (see *AXI GPIO LogiCORE IP Product Guide* (PG144)). The GPIO connects to the user_select input of the block RAM generation and effectively bypasses this block when user_select is High.

For crossing clock domains, the broadcasting is done in two steps. The first step is part of the DDR clock domain and the second step is part of the DAC tile clock domain. The DAC DDR block architecture is illustrated in the following figure.

Figure 3: **DAC DDR Block Architecture**



ADC DDR

A DDR4 memory controller (see *UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide* (PG150)) is instantiated to control the external DDR4 memory. The ADC output is duplicated on the output of the RF analyzer block RAM capture block. An AXI4-Stream interconnect is then used as a switch to select which ADC waveform is fed into the DMA and captured into the DDR memory. To allow the software to control the DMA accurately, a TLAST

block is created. This block generates a TLAST pulse so that the DMA generates an interrupt after the correct number of samples. Since the dual ADC tiles have two AXI4-Stream outputs in I/Q mode, an AXI4-Stream combiner is used to write both streams in the DDR memory. Consequently, the real mode cannot be used to capture the dual ADC tile in the DDR. The ADC DDR block architecture is shown in the following figures.

Figure 4: **ADC DDR Block Architecture for Quad ADC Tiles**

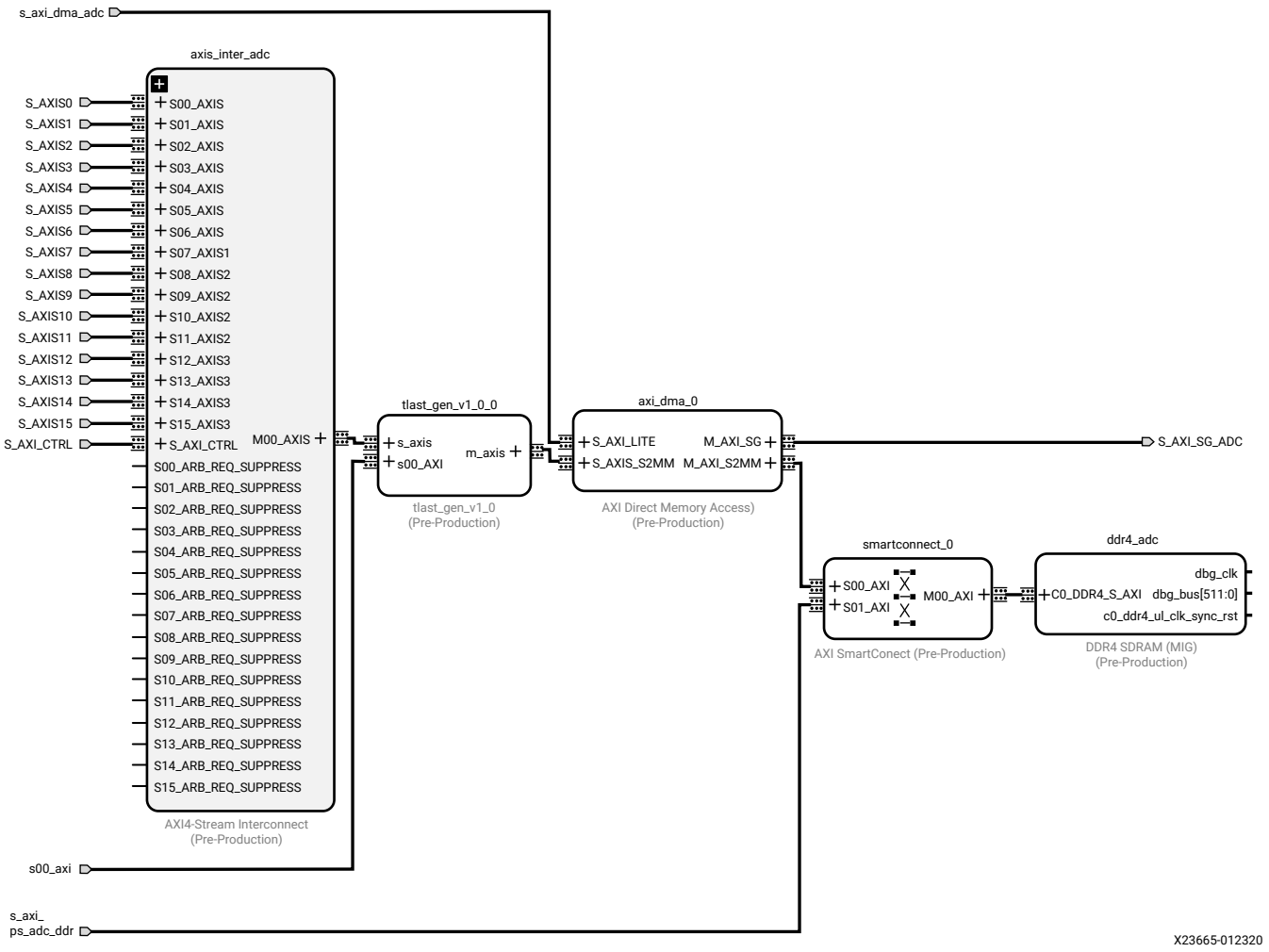
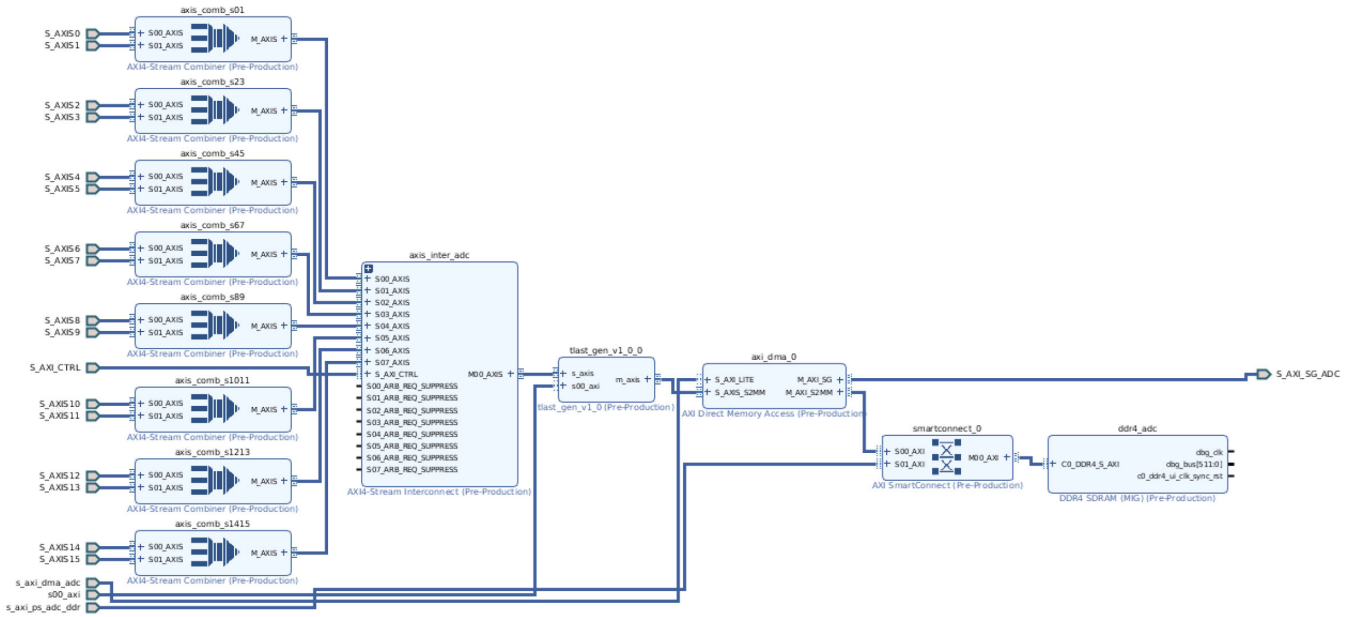


Figure 5: **ADC DDR Block Architecture for Dual ADC Tiles**



Clocking Scheme

The clocking scheme is built to support both individual tile clocks and independent ADC or DAC multi-tile synchronization (MTS). Consequently, two of the typical clocking schemes from the *Zynq UltraScale+ RFSoC RF Data Converter LogiCORE IP Product Guide* (PG269) are merged. The first clocking scheme uses the output of each tile to connect to a mixed-mode clock manager (MMCM) or phase-locked loop (PLL), which gives an independent clock per tile. The second clocking scheme supports MTS, takes its input from a fabric pin, and generates an output for the tiles selected by the MTS functions. BUFGMUX and MMCM dual input allows merging these two clocking schemes. For the ADC, a mixture of MMCM and PLL are used. The ADC clocking scheme figure shows a representation for two tiles. The DAC clocking scheme figure shows a representation for two tiles. For the DAC, only the MMCM is used. The path used in the MTS case is shown in red.

Figure 6: **ADC Clocking Scheme**

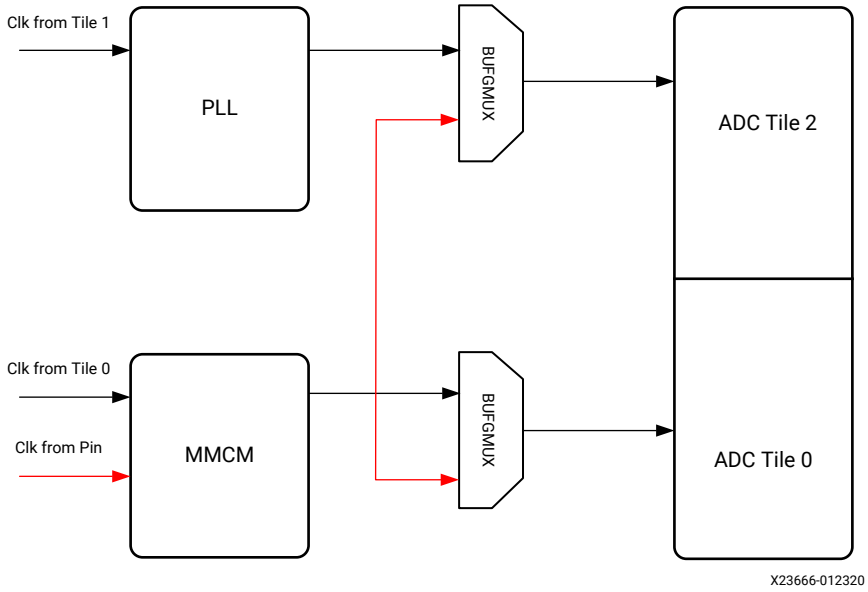
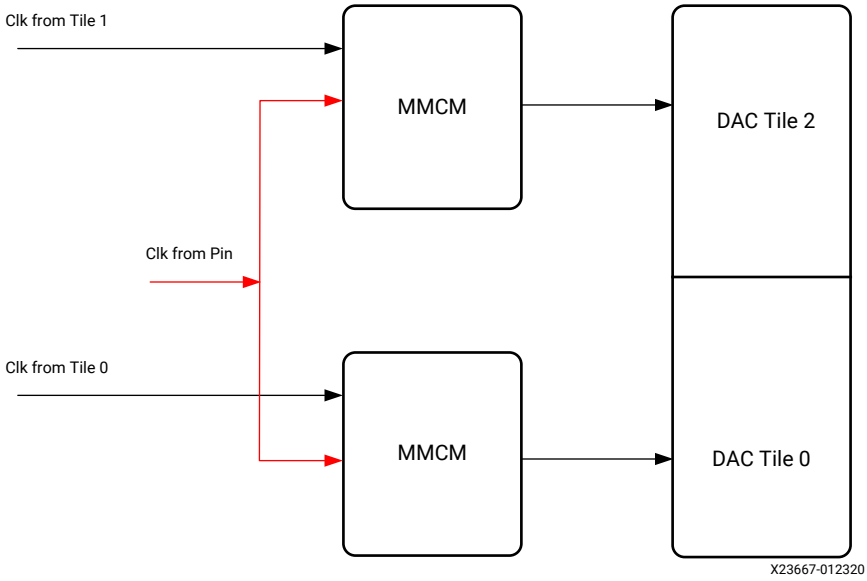


Figure 7: **DAC Clocking Scheme**



Chapter 4

Software Design and Build

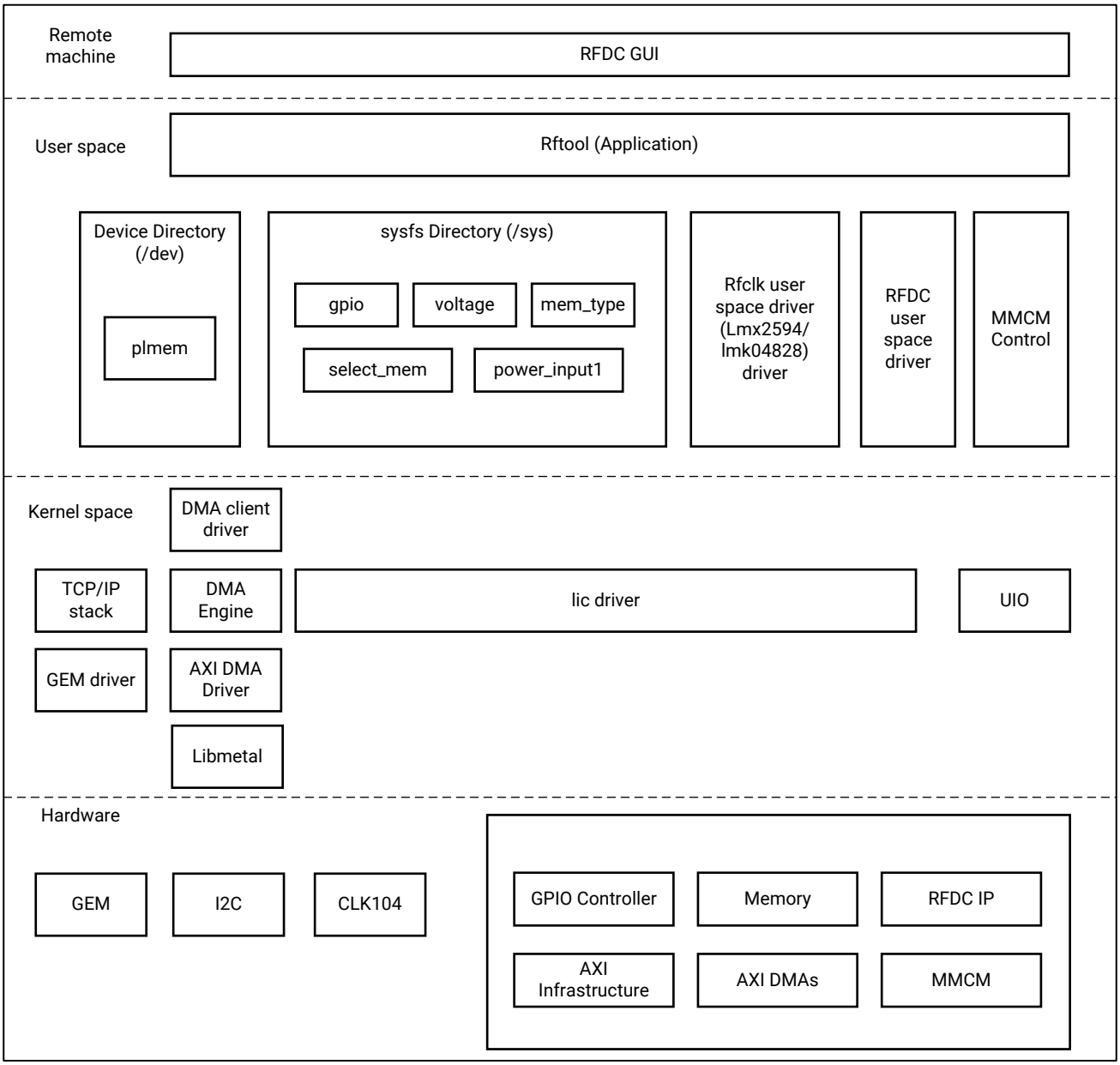
This chapter describes the software platform running on the application processing unit (APU), which is logically further subdivided into user and kernel space components (see Figure 8: APU Linux Software Platform). The Linux application rftool in the user space receives the command over the Ethernet and performs the appropriate action. This application is the main interface to the GUI and uses a string-based communication protocol described in Chapter 5: Protocol Specifications. Device drivers expose a systematic interface to control hardware. These interfaces are used by the user space application to control hardware.

All configurations of the ADCs and DACs are done by the Linux application rftool running on the processing system (PS). The application supports all the configuration options supported by the GUI. The GUI sends configuration details via the Ethernet interface to the Linux application, which are implemented by their respective software APIs.

Software Architecture

The following figure shows the APU Linux software platform, which includes two logical software flows: the control path and the datapath. The control path and datapath are implemented using two different TCP sockets. The components in the software flows are implemented in the user space and the kernel space.

Figure 8: **APU Linux Software Platform**



User Space Components

The Linux user space components used are:

- The rftool is the Linux application that receives commands over the Ethernet from the PC GUI and performs appropriate actions.

- RFDC user space drivers provide APIs for communication with the RFDC hardware.
- RFCLK user space drivers provide APIs for communications with external CLK104 daughter boards.
- DMA client driver interface /dev/pl_mem is used to allocate a buffer from the PL DDR, and is also used to trigger a DMA transaction from the user space.

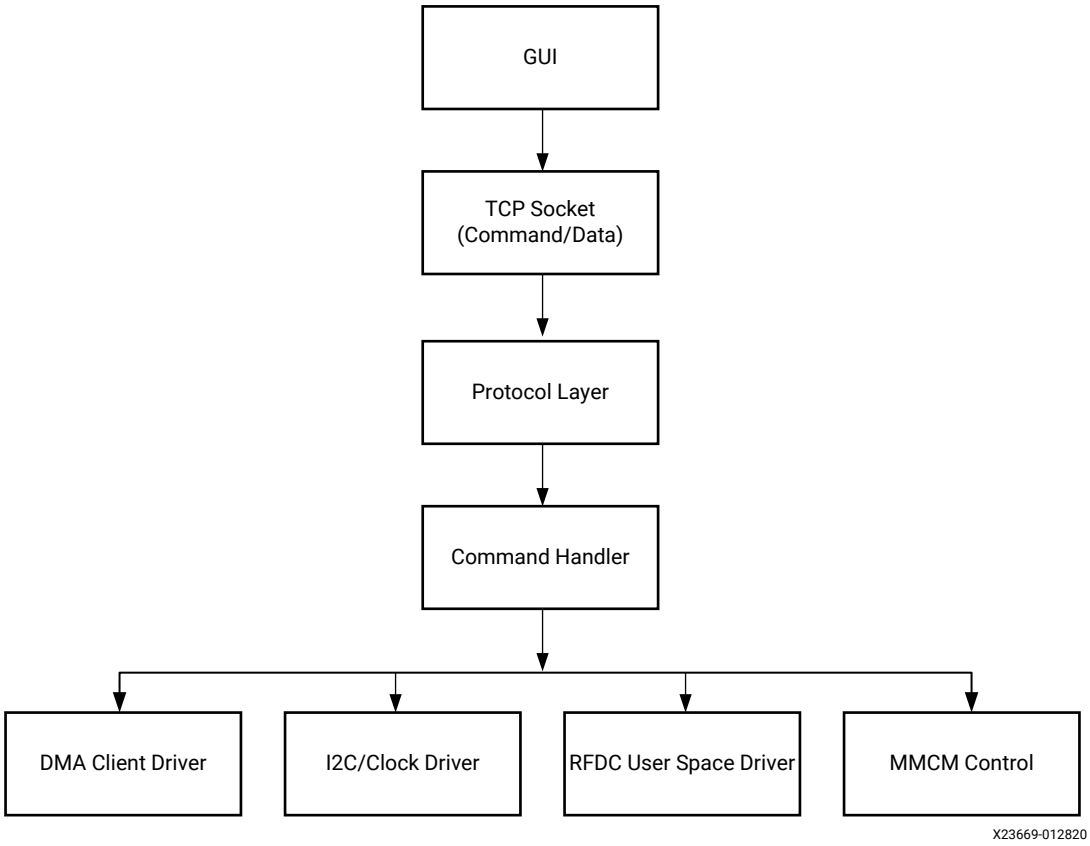
Kernel Space Components

The Linux kernel space components used are:

- The GEM driver provides the interface to send and receive packets over the GEM Ethernet controller.
- The TCP/IP stack provides the interface to create a transmission control protocol (TCP) socket. It also provides the interface to send and receive data over the TCP socket. The stack uses the GEM driver to send and receive packets from a network interface card (NIC).
- The DMA client driver provides the interface for a user application to trigger DMA transactions.
- The AXI DMA driver and DMA engine provide a driver interface to the AXI DMA. The DMA client driver uses APIs for this DMA engine. The DMA engine uses the AXI DMA driver to control hardware.
- The libmetal driver is a kernel space driver that allows the execution of user space drivers.
- Userspace I/O (UIO) is a Linux framework used to export a hardware space to a user space. It is used by the RFDC user space driver libmetal to configure RFDC hardware.
- MMCM control is reconfiguring the MMCM/PLL depending on tile clock requirement.

The following figure shows the application execution flow.

Figure 9: **Application Execution Flow**



Software Build

To build the software, the board support package (BSP) and build tools are needed. For the tools, see the 2019.2 version of the *PetaLinux Tools Documentation: PetaLinux Command Line Reference* (UG1157)

1. Ensure the PetaLinux tools are available in the PATH environment variable.
2. To create the project, enter:

```
petalinux-create -t project -s <bsp> -n <project_name>
```

3. To build the project, enter:

```
cd <project_name>

petalinux-build -v
```

4. To make the images for the SD card, enter:

```
petalinux-package --force --boot --fsbl ./images/linux/
zynqmp_fsbl.elf --fpga ./images/linux/system.bit --pmufw ./
images/linux/pmufw.elf --u-boot ./images/linux/u-boot.elf
```

5. Copy these files to the SD card:

   `<project_name>/images/linux/BOOT.BIN`

   `<project_name>/images/linux/image.ub`

   `autostart.sh` (copy from the `Externals\image\216` folder)

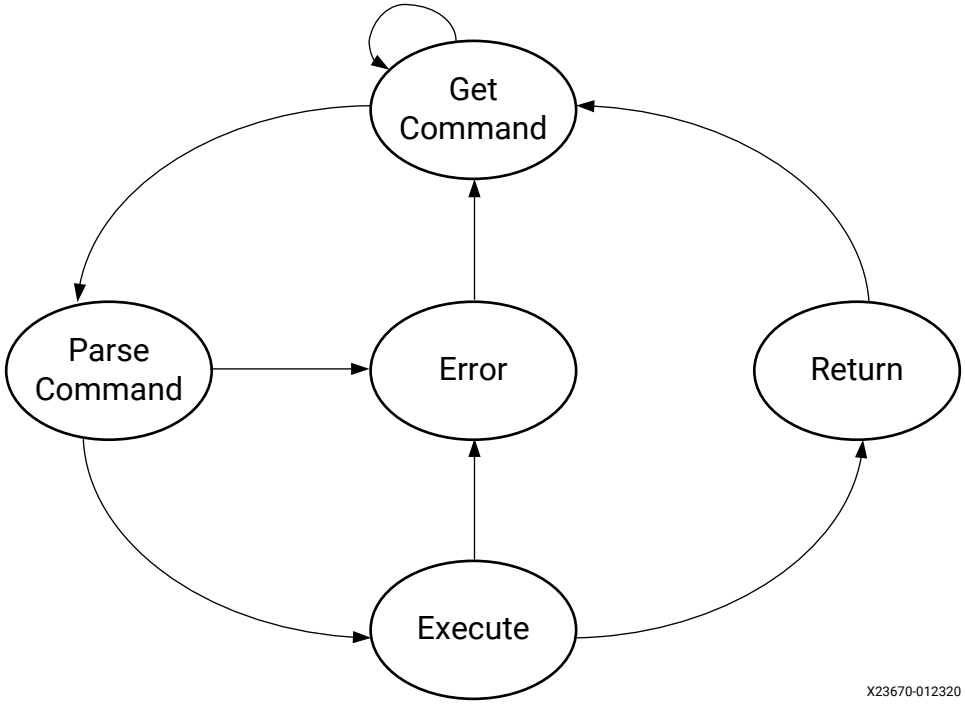
Chapter 5

Protocol Specifications

Protocol Overview

To communicate between the host and the design, a simple but robust protocol is used. This protocol is a string-based, space-separated command and response protocol. The state machine is represented in the following figure.

Figure 10: **Finite State Machine Representing the Protocol**



Each command is sent in text (ASCII) format, parsed to check that the command is known and the number of arguments is correct, and then executed. The design returns the command name plus some parameters or the error type without any parameter if an error condition occurs. The host can get information on the error by using the `getlog` command, which reads the metal log buffer (see *Zynq UltraScale+ RFSoC RF Data Converter LogiCORE IP Product Guide* (PG269)).

The command can be split into these categories.

- RFDC commands are usually a direct translation of the RFDC driver with an addition of error checking when necessary
- PL design commands control the PL side of the design (MMCM, GPIO, capture and generation memories, etc.)
- Board control commands control some of the board components such as external clocks

Protocol Rules

The full list of commands is provided in Appendix A: Command List.

The command syntax for input is:

- `CMD PARAM1 PARAM2 PARAM3` (with any number of parameters)
- Commands and parameters are separated by an ASCII space
- Commands and parameters are human readable, i.e., sent in ASCII format
- Receive end of line is `\n`

The output return format is:

- If an input command is not recognized, the parser errors out and sends `ERROR: CMD: Invalid Command\n`
- If an input command is recognized, the parser checks the number of arguments, and sends `ERROR: CMD: Invalid Number of Arguments\n`
- If execution succeeds, the result is returned `CMD param1 param2 ... paramX value1 value2 ... valueY\n`
- Commands that are not expected to return values return their name if execution succeeds: `[CMD]`
- CMD and values are separated by a space
- Receive end of line is `\n`
- Any log or messages from metal-log can be returned via the `getlog` command; the `\r\n` characters are replaced from any log messages with "|" and a single `\n` appended at the end

Socket Interface

The TCP/IP socket protocol is used for the datapath and control path. The GUI running on the host machine has TCP clients, and the TCP servers are running under the Linux application on the Zynq UltraScale+ RFSoC PS. The control path works on TCP port 8081 and the datapath works on TCP port 8082. The TCP socket uses a Linux TCP/IP stack, which uses the GEM Ethernet controller and driver for sending packets. Because the TCP socket is used, it is possible to run the GUI on any networked machines.

XILINX

Chapter 6

Software Design Notes — Multimaster Access of CLK104 on TCA9548 Multiplexer

The TCA9548 multiplexer is shared between the I2C masters MSP430 and APU on Gen 3 RFSoC. The operation to select the CLK104 device on multiplexer and read/write operation is completed in three steps in the I2C driver. When one master is accessing the device, the other master should not access it.

Command List

The commands supported by the evaluation tool are classified as basic commands, RFDC IP commands, board control commands, and PL data and control commands. The following table lists the supported commands.

Table 1: **Command List**

Command	Input Parameters	Output Parameters	Description
Basic Commands			
GUI_Title	None	Rftool Version	Rftool version, for example, “(RFEvalTool v1.8)”.
RfdcVersion	None	Version Number(s)	RFdc API version.
Version	None	Rftool Version	Rftool version.
TermMode	Mode	None	Switch between UI mode (mode=0) and Interactive mode (mode=1). Interactive mode prints more status information to the terminal and should not be used with the evaluation tool GUI.
GetLog	None	Logged Strings	Uses metal_log to return any error, warning, or informational messages returned from the API or command interface.
JtagIdcode	None	Idcode	Return the device JTAG IDCODE.
Help	None	None	Print help on command interface.
Disconnect	None	None	
RFDC IP Commands			
GetBlockStatus	Type, Tile, Block	Type, Tile, Block, BlockStatus.SamplingFreq, BlockStatus.AnalogDataPathStatus, BlockStatus.DigitalDataPathStatus, BlockStatus.DataPathClocksStatus, BlockStatus.IsFIFOFlagsAsserted, BlockStatus.IsFIFOFlagsEnabled	Cf. PG269
GetCalFreeze	Tile, Block	Tile, Block, CalFreezePtr.CalFrozen, CalFreezePtr.FreezeCalibration, CalFreezePtr.DisableFreezePin	Cf. PG269
GetCalibrationMode	Tile, Block	Tile, Block, CalibrationMode	Cf. PG269

Table 1: **Command List** *(cont'd)*

Command	Input Parameters	Output Parameters	Description
GetClkDistribution	Type, Tile	Type, Tile, SettingsPtr.PLLEnable, SettingsPtr.DivisionFactor, SettingsPtr.DistributedClock, DistStatusPtr.Enabled, DistStatusPtr.DistributionSource, DistStatusPtr.UpperBound, DistStatusPtr.LowerBound, DistStatusPtr.MaxDelay, DistStatusPtr.MinDelay, DistStatusPtr.IsDelayBalanced	Cf. PG269
GetClockSource	Type, Tile	Type, Tile, ClockSource	Cf. PG269
GetCoarseDelaySettings	Type, Tile, Block	Type, Tile, Block, Settings.CoarseDelay, Settings.EventSource	Cf. PG269
GetConnectedData	Type, Tile, Block	Type, Tile, Block, ConnectedIData, ConnectedQData	Cf. PG269
GetDACCompMode	Tile, Block	Tile, Block, EnablePtr	Cf. PG269
GetDataPathMode	Tile, Block	Tile, Block, Mode	Cf. PG269
GetDecimationFactor	Tile, Block	Tile, Block, DecimationFactor	Cf. PG269
GetDecoderMode	Tile, Block	Tile, Block, DecoderMode	Cf. PG269
GetDither	Tile, Block	Tile, Block, Mode	Cf. PG269
GetDSA	Tile, Block	Tile, Block, SettingsPtr.DisableRTS, SettingsPtr.Attenuation	Cf. PG269
GetFabClkOutDiv	Type, Tile	Type, Tile, FabClkDiv	Cf. PG269
GetFabRdVldWords	Type, Tile, Block	Type, Tile, Block, FabricDataRate	Cf. PG269
GetFabWrVldWords	Type, Tile, Block	Type, Tile, Block, FabricDataRate	Cf. PG269
GetFIFOStatus	Type, Tile	Type, Tile, enable	Cf. PG269
GetIMRPassMode	Tile, Block	Tile, Block, Mode	Cf. PG269
GetInterpolationFactor	Tile, Block	Tile, Block, InterpolationFactor	Cf. PG269
GetIntrStatus	Type, Tile, Block	Type, Tile, Block, IntrStatus	Cf. PG269
GetInvSincFIR	Tile, Block	Tile, Block, enable	Cf. PG269
GetIPStatus	None	IpStatus.DACTileStatus[Tile].IsEnabled, IpStatus.DACTileStatus[Tile].BlockStatusMask, IpStatus.DACTileStatus[Tile].TileState, IpStatus.DACTileStatus[Tile].PowerUpState, IpStatus.DACTileStatus[Tile].PLLState, IpStatus.ADCTileStatus[Tile].IsEnabled, IpStatus.ADCTileStatus[Tile].BlockStatusMask, IpStatus.ADCTileStatus[Tile].TileState, IpStatus.ADCTileStatus[Tile].PowerUpState, IpStatus.ADCTileStatus[Tile].PLLState, IpStatus.State	Cf. PG269
GetLinkCoupling	Tile, Block	Tile, Block, Mode	Cf. PG269
GetMixerSettings	Type, Tile, Block	Type, Tile, Block, Mixer_Settings.Freq, Mixer_Settings.PhaseOffset, Mixer_Settings.EventSource, Mixer_Settings.MixerType, Mixer_Settings.CoarseMixFreq, Mixer_Settings.MixerMode, Mixer_Settings.FineMixerScale	Cf. PG269
GetMTSEnable	Type, Tile	Type, Tile, enable	Cf. PG269
GetNyquistZone	Type, Tile, Block	Type, Tile, Block, NyquistZone	Cf. PG269
GetOutputCurr	Tile, Block	Tile, Block, OutputCurr	Cf. PG269

Table 1: **Command List** *(cont'd)*

Command	Input Parameters	Output Parameters	Description
GetPLLConfig	Type, Tile	Type, Tile, PLLSettings.Enabled, PLLSettings.RefClkFreq, PLLSettings.SampleRate, PLLSettings.RefClkDivider, PLLSettings.FeedbackDivider, PLLSettings.OutputDivider	Cf. PG269
GetPLLLockStatus	Type, Tile	Type, Tile, LockStatus	Cf. PG269
GetQMCSettings	Type, Tile, Block	Type, Tile, Block, QMC_Settings.GainCorrectionFactor, QMC_Settings.PhaseCorrectionFactor, QMC_Settings.EnablePhase, QMC_Settings.EnableGain, QMC_Settings.OffsetCorrectionFactor, QMC_Settings.EventSource	Cf. PG269
GetThresholdSettings	Tile, Block	Tile, Block, ThresholdSettings.UpdateThreshold, ThresholdSettings.ThresholdMode[0], ThresholdSettings.ThresholdMode[1], ThresholdSettings.ThresholdAvgVal[0], ThresholdSettings.ThresholdAvgVal[1], ThresholdSettings.ThresholdUnderVal[0], ThresholdSettings.ThresholdUnderVal[1], ThresholdSettings.ThresholdOverVal[0], ThresholdSettings.ThresholdOverVal[1]	Cf. PG269
GetDACPower	Board_id, Tile	Board_id, Tile, value_1, value_2, value_3, value_4, value_5	Cf. PG269
IntrClr	Type, Tile, Block, IntrMask	None	Cf. PG269
IntrDisable	Type, Tile, Block, IntrMask	None	Cf. PG269
IntrEnable	Type, Tile, Block, IntrMask	None	Cf. PG269
MTS_Sysref_Config	Enable, dac_tiles, adc_tiles	DAC_Sync_Config.Target_Latency, DAC_Sync_Config.Offset[Tile], DAC_Sync_Config.Latency[Tile], DAC_Sync_Config.Marker_Delay, DAC_Sync_Config.SysRef_Enable, ADC_Sync_Config.Target_Latency, ADC_Sync_Config.Offset[Tile], ADC_Sync_Config.Latency[Tile], ADC_Sync_Config.Marker_Delay, ADC_Sync_Config.SysRef_Enable	Cf. PG269
MultiConverter_Init	Type	None	Cf. PG269
MultiConverter_Sync	Type, ADC_Sync_Config.Target_Latency/ DAC_Sync_Config.Target_Latency, Tile	Sync_config.Target_Latency, Sync_config.Offset[Tile], Sync_config.Latency[Tile], Sync_config.Marker_Delay, Sync_config.SysRef_Enable	Cf. PG269
Reset	Type, Tile	None	Cf. PG269
ResetNCOPhase	Type, Tile, Block	None	Cf. PG269
RF_ReadReg16	Offset	Value	Cf. PG269
RF_ReadReg32	Offset	Value	Cf. PG269
RF_WriteReg16	Offset, Value	None	Cf. PG269
RF_WriteReg32	Offset, Value	None	Cf. PG269
SetCalFreeze	Tile, Block, CalFreezePtr.FreezeCalibration, CalFreezePtr.DisableFreezePin	None	Cf. PG269
SetCalibrationMode	Tile, Block, Calibration Mode	None	Cf. PG269

Table 1: **Command List** *(cont'd)*

Command	Input Parameters	Output Parameters	Description
SetClkDistribution	Distribution_Settings.DAC[Tile].SourceTile Distribution_Settings.DAC[Tile].PLLEnable Distribution_Settings.DAC[Tile].PLLSettings.RefClkFreq Distribution_Settings.DAC[Tile].PLLSettings.SampleRate Distribution_Settings.DAC[Tile].DivisionFactor Distribution_Settings.DAC[Tile].DistributedClock Distribution_Settings.ADC[Tile].SourceTile Distribution_Settings.ADC[Tile].PLLEnable Distribution_Settings.ADC[Tile].PLLSettings.RefClkFreq Distribution_Settings.ADC[Tile].PLLSettings.SampleRate Distribution_Settings.ADC[Tile].DivisionFactor Distribution_Settings.ADC[Tile].DistributedClock	None	Cf. PG269
SetCoarseDelaySettings	Type, Tile, Block, Settings.CoarseDelay, Settings.EventSource	None	Cf. PG269
SetDACPowerMode	Board_id, Tile, Block, op_current	Board_id, Tile, Block, op_current	Cf. PG269
SetDACVOP	Tile, Block, uACurrent	None	Cf. PG269
SetDataPathMode	Tile, Block, Mode	Tile, Block, Mode	Cf. PG269
SetDecimationFactor	Tile, Block, DecimationFactor	None	Cf. PG269
SetDecoderMode	Tile, Block, DecoderMode	None	Cf. PG269
SetDither	Tile, Block, Mode	Tile, Block, Mode	Cf. PG269
SetDSA	Tile, Block, DisableRTS, Attenuation	Tile, Block, Attenuation	Cf. PG269
SetFabClkOutDiv	Type, Tile, FabClkDiv	None	Cf. PG269
SetFabRdVldWords	Tile, Block, FabricDataRate	None	Cf. PG269
SetFabWrVldWords	Tile, Block, FabricDataRate	None	Cf. PG269
SetIMRPassMode	Tile, Block, Mode	Tile, Block, Mode	Cf. PG269
SetInterpolationFactor	Tile, Block, InterpolationFactor	None	Cf. PG269
SetInvSincFIR	Tile, Block, enable	None	Cf. PG269
SetMixerSettings	Type, Tile, Block, Mixer_Settings.Freq, Mixer_Settings.PhaseOffset, Mixer_Settings.EventSource, Mixer_Settings.MixerType, Mixer_Settings.CoarseMixFreq, Mixer_Settings.MixerMode, Mixer_Settings.FineMixerScale	None	Cf. PG269
SetMMCMReg	Type, Tile, Mult, Mult_frac, Div, clkout0_div, clkout0_frac	None	Cf. PG269
SetNyquistZone	Type, Tile, Block, NyquistZone	None	Cf. PG269
SetQMCSettings	Type, Tile, Block, QMC_Settings.EnablePhase, QMC_Settings.EnableGain, QMC_Settings.GainCorrectionFactor, QMC_Settings.PhaseCorrectionFactor, QMC_Settings.OffsetCorrectionFactor, QMC_Settings.EventSource	None	Cf. PG269

Table 1: **Command List** *(cont'd)*

Command	Input Parameters	Output Parameters	Description
SetThresholdSettings	Tile, Block, ThresholdSettings.UpdateThreshold, ThresholdSettings.ThresholdMode[0], ThresholdSettings.ThresholdMode[1], ThresholdSettings.ThresholdAvgVal[0], ThresholdSettings.ThresholdAvgVal[1], ThresholdSettings.ThresholdUnderVal[0], ThresholdSettings.ThresholdUnderVal[1], ThresholdSettings.ThresholdOverVal[0], ThresholdSettings.ThresholdOverVal[1]	None	Cf. PG269
SetupFIFO	Type, Tile, Enable	None	Cf. PG269
Shutdown	Type, Tile	None	Cf. PG269
StartUp	Type, Tile	None	Cf. PG269
UpdateEvent	Type, Tile, Block, Event	None	Cf. PG269
DynamicPLLConfig	Type, Tile, Source, RefClkFreq, SamplingRate	RefClkDivider, FeedbackDivider, OutputDivider	Cf. PG269
MultiBand	Type, Tile, DigitalDataPathMask, DataType, DataConverterMask	Type, Tile, DigitalDataPathMask, DataType, DataConverterMask	Cf. PG269
SetSignalDetector	Tile, Block, Signal_Detector_Settings.Mode, Signal_Detector_Settings.TimeConstant, Signal_Detector_Settings.Flush, Signal_Detector_Settings.EnableIntegrator, Signal_Detector_Settings.HighThreshold, Signal_Detector_Settings.LowThreshold, Signal_Detector_Settings.HysteresisEnable	Tile, Block	Cf. PG269
GetSignalDetector	Tile, Block	Tile, Block, Signal_Detector_Settings.Mode, Signal_Detector_Settings.TimeConstant, Signal_Detector_Settings.Flush, Signal_Detector_Settings.EnableIntegrator, Signal_Detector_Settings.HighThreshold, Signal_Detector_Settings.LowThreshold, Signal_Detector_Settings.HysteresisEnable	Cf. PG269
SetCalCoefficients	Tile, Block, CalBlock, Coeffs.Coeff0, Coeffs.Coeff1, Coeffs.Coeff2, Coeffs.Coeff3, Coeffs.Coeff4, Coeffs.Coeff5, Coeffs.Coeff6, Coeffs.Coeff7	None	Cf. PG269
GetCalCoefficients	Tile, Block, CalBlock	Tile, Block, CalBlock, Coeffs.Coeff0, Coeffs.Coeff1, Coeffs.Coeff2, Coeffs.Coeff3, Coeffs.Coeff4, Coeffs.Coeff5, Coeffs.Coeff6, Coeffs.Coeff7	Cf. PG269
DisableCoefficientsOverride	Tile, Block, CalBlock	None	Cf. PG269
SetDACCompMode	Tile, Block, Enable	Tile, Block, Enable	Cf. PG269
GetEnabledInterrupts	Type, Tile, Block	Type, Tile, Block, IntrMask	Cf. PG269
Board Control Commands			
GetExtPllConfig	Board_id, Pll_src	Freq	Return the current frequency of the LMX.
GetExtPllFreqList	Board_id, Pll_src	ADC_FREQ_LIST[LMX_ADC_NUM] or DAC_FREQ_LIST[LMX_DAC_NUM] depending on Pll_Src	Return a list of allowed values for the LMX.
RfclkReadReg	Board_id, chip_id, reg_addr	Board_id, chip_id, reg_addr, data	Read a register from the LMK/LMX.
RfclkWriteReg	Board_id, chip_id, reg_addr, data	Board_id, chip_id, reg_addr, data	Write a register from the LMK/LMX.
SetExtParentclk	Board_id, freq	Board_id, freq	Configure LMK clock with requested frequency.

Table 1: **Command List** *(cont'd)*

Command	Input Parameters	Output Parameters	Description
SetExtPllClkRate	Board_id, Pll_src, freq	Board_id, Pll_Src, freq	Configure PLL (LMX) with requested frequency.
PL Data and Control Commands			
GetMemtype	Type, Tile	Memtype	Get selected memory type (block RAM or DDR).
SetLocalMemSample	Type, Tile, Block, numsamples	None	Set the number of samples to be generated or captured.
SetMemtype	Type, Tile, mem_type	Error code on failure	Set memory type to DDR (0) or block RAM (1).
SetMMCM	Type, Tile	MMCM_Lock, Mult, Div, clkout0_div, Clk0DivFrac	Reconfigure the MMCM according to the tile settings, block 0 of this tile and Fs.
MTS_Setup	Type, Enable	None	Setup the clocking scheme for multi-tile synchronization feature.
GetMMCMReg	Type, Tile	Lock, Mult, MultFrac, Div, Clk0Div, Clk0DivFrac, Clk1Div	Get the MMCM settings.
MMCM_Rst	Type, Tile	None	Reset the MMCM.
LocalMemInfo	Type	Type, memBaseAddr, numTiles, numMem, memSize, numWords, mem_enable, mem_clksel	Get information on the memory.
LocalMemTrigger	Type, clksel, numsamples, rfdc_ch	None	Trigger the memory channel according to the mask rfdc_ch.
LocalMemAddr	Type, Tile, Block	Type, Tile, Block, Addr_I, Addr_Q	Get the memory addresses associated with the tile and block.
WriteDataToMemory	Tile, Block, number of bytes, interleaved pair	None	In this case, the buffer address is allocated by Linux (CMA pool) from PS DDR and firmware writes the data to the buffer by reading data from a socket.
ReadDataFromMemory	Tile, Block, number of bytes, interleaved pair	None	In this case, the buffer address is allocated by Linux (CMA pool) from PL DDR and firmware reads the data from the buffer and sends it to a socket. In this command, the number of bytes should be aligned to 32.

XILINX

Appendix B

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this guide:

1. *Zynq UltraScale+ RFSoC RF Data Converter LogiCORE IP Product Guide* (PG269)
2. *ZCU216 Evaluation Board User Guide* (UG1390)
3. *RF Data Converter Interface User Guide* (UG1309)
4. *UltraScale Architecture-Based FPGAs Memory IP LogiCORE IP Product Guide* (PG150)
5. *AXI DMA LogiCORE IP Product Guide* (PG021)
6. *AXI GPIO LogiCORE IP Product Guide* (PG144)

Please Read: Important Legal Notices

Copyright