

## Hardware Demonstration Design

A hardware demonstration design, targeting the Kintex Ultrascale KCU105, Virtex Ultrascale VCU108, Zynq Ultrascale+ ZCU102 or Virtex Ultrascale+ VCU118 evaluation platforms, can be generated using the script provided in the download.

A Master and Slave demonstration design is also available on the Virtex Ultrascale VCU108 evaluation board as shown in Figure 2.

This design can be implemented using the Xilinx Vivado Design Suite using the provided source code and scripts. The design can be downloaded to the targeted FPGA board, controlled and monitored using a Xilinx Vivado Hardware Manager GUI.

The CPRI Hardware Demo folder contains the following files and folders:

- cpri\_hwdemo.tcl. This script is to be used to generate the Hardware Demo.
- hwdemo\_files. This folder contains all the required VHDL files for the Hardware demo project.

Figure 1 illustrates the Hardware demonstration design, which consists of the following:

- CPRI core generated including the shared logic level. A single core is instantiated and intended to be connected in loopback by SMA or optical cabling.
- The CPRI core has been generated with the following parameters
  - Master core
  - Line rate support and reference clock
    - Kintex Ultrascale:
      - 10.1376 Gbps with 307.2 MHz reference clock
    - Virtex Ultrascale:
      - 24.33024 Gbps with 245.76 MHz reference clock
    - Zynq Ultrascale+:
      - 12.16512 Gbps with 245.76 MHz reference clock
    - Virtex Ultrascale+:
      - 24.33024 Gbps with 245.76 MHz reference clock
  - R21 timers and AXI-4 Lite Management Interface have been included
  - Ethernet Logic and GMII Interface have been included
  - Additional transceiver and status ports
- CPRI example design modules to stimulate and monitor the CPRI core interfaces.
- Hardware demonstration top-level VHDL wrapper source code, instantiating the CPRI core, example design modules, VIO and ILA modules, and additional logic to setup, control and monitor the CPRI core.

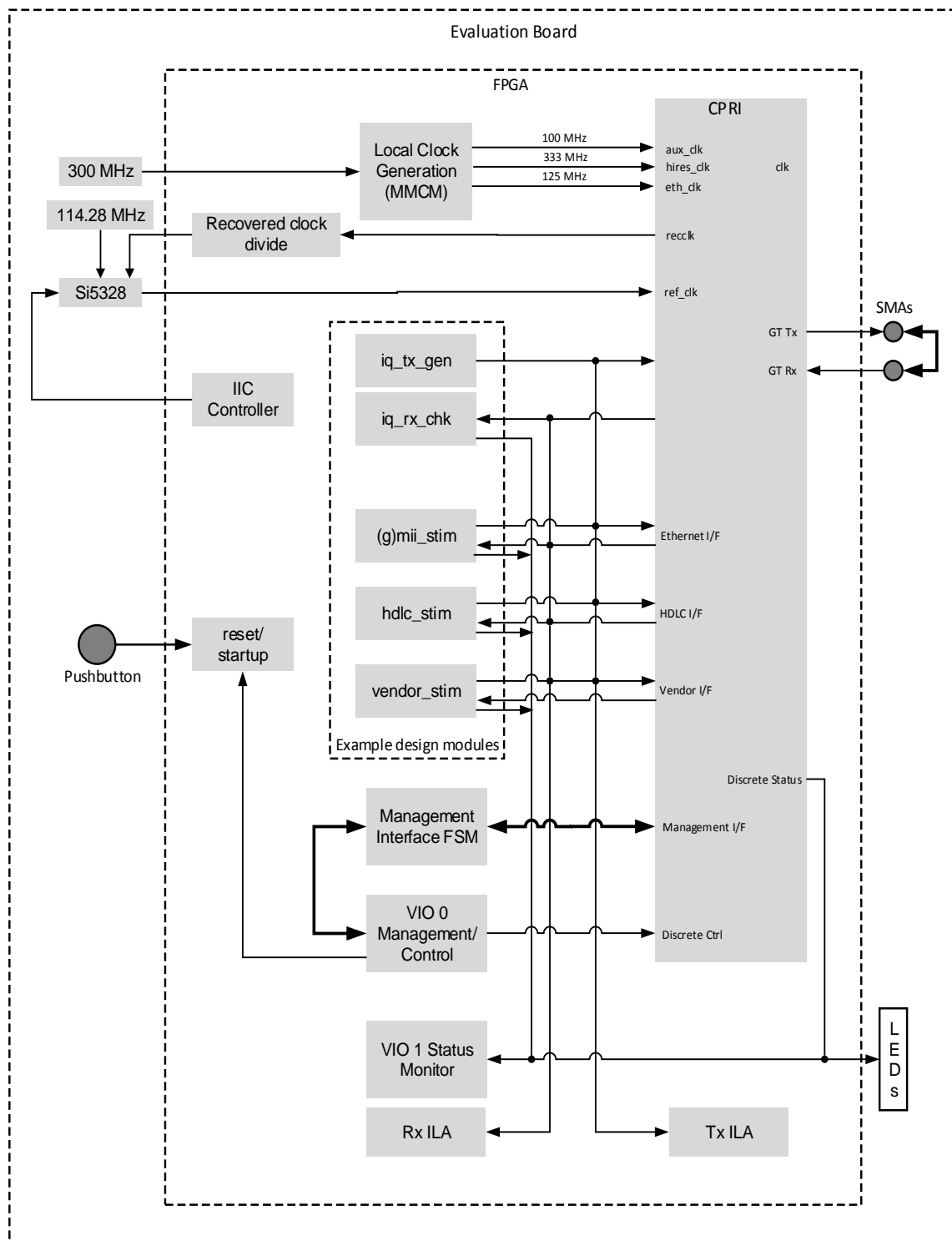


Figure 1 Hardware Demonstration Design

Figure 2 illustrates the Master-Slave Hardware demonstration design, which consists of the following:

- A Master CPRI core and Slave CPRI core is generated using shared logic. The master and slave cores are intended to be connected in loopback via the bullseye connector J87. The Master-Slave design is generated on the Virtex Ultrascale VCU108 evaluation board.
- The CPRI cores have been generated with the following parameters
  - Line rate support: 12.16512 Gbps
  - Reference clock: 245.76 MHz
  - R21 timers and AXI-4 Lite Management Interface have been included
  - Ethernet Logic and GMII Interface have been included
  - Additional transceiver and status ports
- CPRI example design modules to stimulate and monitor the CPRI core interfaces.
- Hardware demonstration top-level VHDL wrapper source code, instantiating the CPRI cores, example design modules, VIO and ILA modules, and additional logic to setup, control and monitor the Master and Slave CPRI cores.

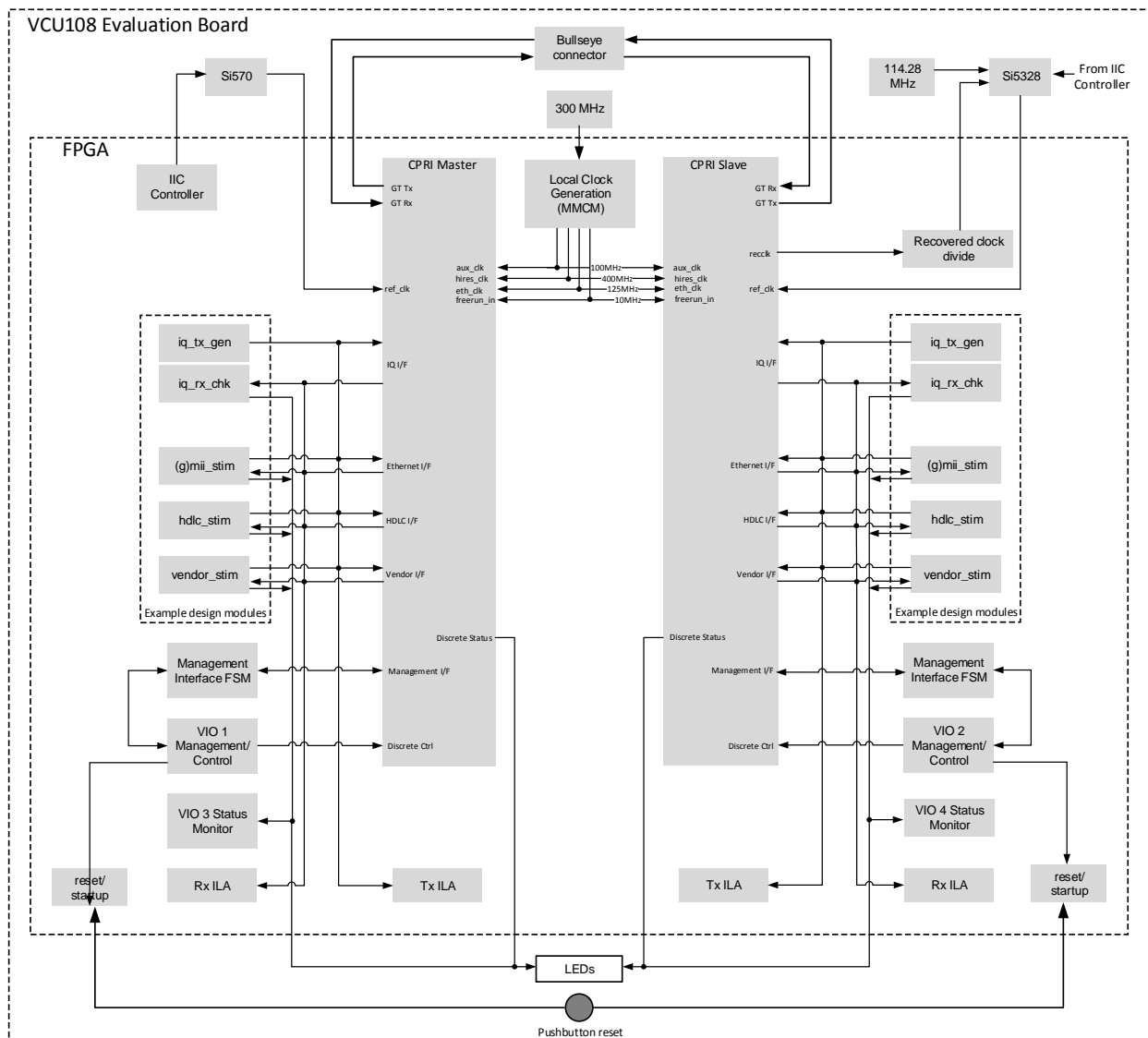


Figure 2 Hardware Demonstration Design

## Overview of the Hardware Demonstration Design

The hardware demonstration design is similar to the example design described in the CPRI Product Guide and reuses all of the example design stimulus and monitoring components. The design instantiates the support level of the CPRI core as is intended for a user design. It also instantiates the relevant stimulus/monitor modules for the main CPRI core interfaces from the example design. The demonstration design consists of the following:

### CPRI Serial Interface

The CPRI core serial TX and RX ports are connected to the serial transceiver SMA connectors on the demonstration board, and are intended to be connected in loopback TX -> RX.

### Reference Clock Generation

The reference clock for the serial transceiver is sourced from the Si5328 clock cleaner chip on the KCU105, VCU108, ZCU102 and VCU118 boards. The device is set up using the I<sup>2</sup>C bus interface. The I<sup>2</sup>C controller in the design initializes the device on start-up and reset. It is set up to provide the selected reference clock frequency from the 114.285 MHz external reference clock input to the Si5328. The I<sup>2</sup>C controller logic is provided as source code (iic\_controller.vhd), and uses a simple look-up table for the I<sup>2</sup>C commands to be transmitted.

### Local Clock Generation

The 300MHz system clock from the fixed oscillator on the demonstration board is routed to an MMCM which generates the following additional clocks

- aux\_clk (100 MHz)
- hires\_clk (400 MHz )
- eth\_clk (125 MHz for GMII)
- An additional 10 MHz clock is generated for use by the start-up sequencing logic and, in Ultrascale designs, for the transceiver reset logic.

### Reset/Startup

Logic is included to control an orderly start-up sequence on configuration or reset of the design. This includes ensuring that the local clock generation is properly locked, initiating the I<sup>2</sup>C programming of the reference clock generation, a time delay of around 1 second to ensure a stable reference clock input, before releasing the CPRI core itself from reset.

The design can be reset either by 'soft' reset using the Control VIO, or by a pushbutton on the demonstration board.

### Management Interface FSM

This small state machine interfaces to the Control VIO to perform AXI management interface Read/Write access cycles to the CPRI core when requested through the VIO console.

### VIO/ILA Modules

The design instantiates the VIO and ILA modules and connects them up to the remainder of the design.

## Status LEDs

Additional visual status indication is provided using the GPIO LEDs on the demonstration board. Information on the LEDs is given in Table 1, Table 2 and Table 3.

GPIO LED	Function	Description
0	State = Operational	ON = CPRI state = Operational
1	CPRI Speed	Indicates current operating line rate
2		
3		
4		
5	Alarm	ON = CPRI Alarm active
6	MMCM Lock	ON = Local Clock Generation MMCM locked
7	refclk alive	Flashing – divide of reference clock

Table 1 GPIO LED Assignments (KCU105/VCU108/ZCU102/VCU118 boards)

LED Code	Line Rate (Mb/s)	Description
0001	614	x1
0010	1228	x2
0011	2457	x4
0100	3072	x5
0101	4915	x8
0110	6144	x10
0111	9830	x16
1000	10137	x20
1001	8110	x16
1010	12165	x24
1011	24330	x48
1100	8110 FEC Enabled	x16
1101	10137 FEC Enabled	x20
1110	12165 FEC Enabled	x24
1111	24330 FEC Enabled	x48
0000	disabled	disabled

Table 2 Line Rate Indication LED Encoding (KCU105/VCU108/ZCU102/VCU118 boards)

GPIO LED	Function	Description
0	State = Operational	ON = Master operational AND Slave operational
1	CPRI Speed	Indicates Master core current operating line rate
2		
3		
4		
5	Alarm	ON = Master alarm active OR Slave alarm active
6	MMCM Lock	ON = Local Clock Generation MMCM locked
7	refclk alive	Flashing – divide of Slave recovered reference clock

Table 3 GPIO LED Assignments (Master-Slave design on VCU108 board)

## Generating the Hardware Demonstration Design

All the files required to create the hardware demonstration design are located in the file available for download. The following steps must be carried out to successfully generate the hardware demo project.

1. Unzip and open the hardware demo folder.
2. In this folder is located `cpri_hwdemo.tcl` file which is a Vivado tcl script. Source this file from the command prompt (`vivado -mode batch -source cpri_hwdemo.tcl`) and select the required demo design. This implement script will perform the following actions:
  - i. Create a new Vivado project for the board selected by the user
  - ii. Add all the required IP and source files to the project.
  - iii. Synthesize the design
  - iv. Implement the design
  - v. Generate a bitstream
3. Once the project has been created, copy the relevant `hw_1` folder into the following directory: `<board name>_hwdemo/<board name>_hwdemo.hw`. This will setup the VIO consoles in the Vivado Hardware Manager.
4. Open the Vivado project which is located in `<board name>_hwdemo/<board name>_hwdemo.xpr` using the Vivado GUI.

## Running the Hardware Design Using Vivado Lab Tools

### Prerequisites for Running the Demonstration

- KCU105, VCU108, ZCU102 or VCU118 Evaluation Platform
- Vivado toolset
- Micro-USB cable for USB JTAG connection
- For loopback operation 2x SMA to SMA cables are required for KCU105 boards. Fibre optic cable required for ZCU102 boards and QSFP loopback cable required for VCU108 and VCU118 boards<sup>1</sup>
- Bullseye connector cable is required for Master-Slave VCU108 demo design.

### Connecting the Demonstration Board

Connect USB port on host PC to USB-JTAG port on the demonstration board (micro-USB). Connect the following loopback cable connections:

- KCU105
  - J29 (serial transceiver Tx P) to J31 (serial transceiver Rx P)
  - J28 (serial transceiver Tx N) to J30 (serial transceiver Rx N)

---

<sup>1</sup> For Virtex Ultrascale boards if fibre optic or QSFP loopback cable not available, CPRI can be set to operate in PMA loopback by setting register 0x20 to 0x08.

- VCU108
  - Transceiver interfaces to QSFP cage J96
- ZCU102
  - Transceiver interfaces to 2x2 SFP cage P2
- VCU118
  - Transceiver interfaces to QSFP1 cage J96
- VCU108 Master-Slave demo design – J87 Bullseye connector
  - P15 (serial transceiver Tx0\_P – Master) to J11 (serial transceiver Rx1\_P – Slave)
  - P16 (serial transceiver Tx0\_N – Master) to J12 (serial transceiver Rx1\_N – Slave)
  - P17 (serial transceiver Rx0\_P – Master) to J13 (serial transceiver Tx1\_P – Slave)
  - P18 (serial transceiver Rx0\_N – Master) to J14 (serial transceiver Tx1\_N – Slave)

**Note:** The bullseye connector must be firmly tightened to ensure good connection.

## Configuring the FPGA

- Connect USB port on host PC to USB-JTAG port on the evaluation board (micro-USB)
- Power up the evaluation board
- Open the Vivado Hardware Manager from the Flow Navigator in the opened project
- Open a new hardware target
- Select the targeted FPGA, and click on next. Leave all other options on the default values.
- Right click on the device that should have appeared in the GUI and select *program device* from drop down menu.<sup>2</sup>
- Locate the bitfile (the location of the file should be selected by default) and click ok. This action will program the device.

Once the configuration is complete, the CPRI core should come up into the operational state.

## Setting Up the VIO GUI

Once the device has been configured the ILA and VIO probes will appear in the Vivado GUI under the device (hw\_ila\_1, hw\_ila\_2, hw\_vio\_1 and hw\_vio\_2). The signals in the control VIO should appear as in Figure 3 and the status VIO should appear as in Figure 4.

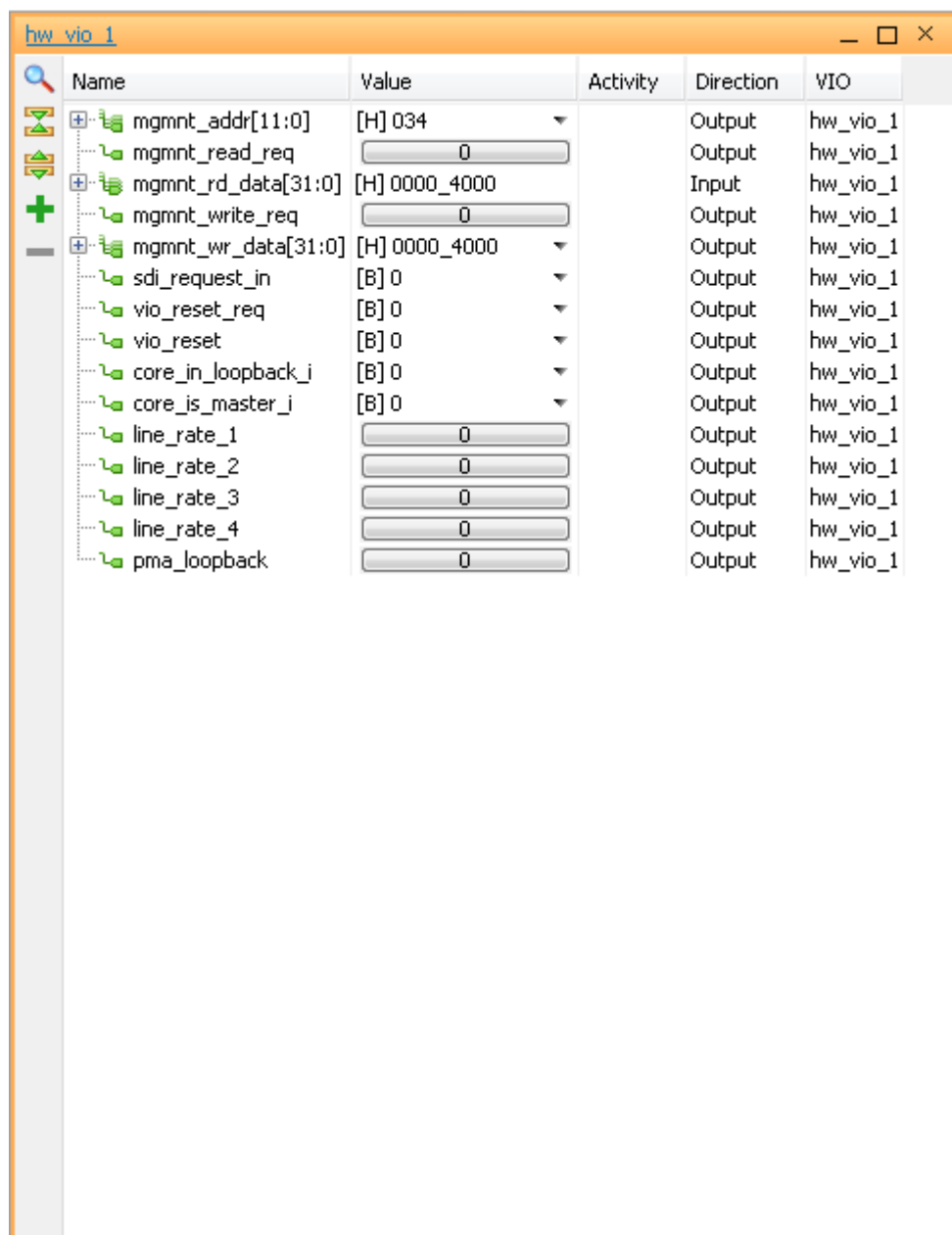
**Note:** After programming check the VIO status in the Hardware pane. If the status is 'Outputs out-of-sync' then right click and select Refresh input and output values from VIO core.

For the Master-Slave design hw\_vio\_1 and hw\_vio\_3 refer to the master CPRI core and hw\_vio\_2 and hw\_vio\_4 refer to the slave CPRI core. Similarly hw\_ila\_1 and hw\_ila\_3 refer to the master CPRI core and hw\_ila\_2 and hw\_ila\_4 refer to the slave CPRI core.

After programming, on the slave hw\_vio\_2 set core\_is\_master and core\_in\_loopback both to zero. The master and slave should synchronise with each other at the top line rate straight away.

---

<sup>2</sup> When programming the device for the first time Vivado may present an error [Labtools 27-1974] stating that there is a mismatch between the design programmed into the device and the current design. This is normal since we have not yet programmed our design onto the device. Select ok and continue.



Name	Value	Activity	Direction	VIO
mgmnt_addr[11:0]	[H] 034		Output	hw_vio_1
mgmnt_read_req	0		Output	hw_vio_1
mgmnt_rd_data[31:0]	[H] 0000_4000		Input	hw_vio_1
mgmnt_write_req	0		Output	hw_vio_1
mgmnt_wr_data[31:0]	[H] 0000_4000		Output	hw_vio_1
sdi_request_in	[B] 0		Output	hw_vio_1
vio_reset_req	[B] 0		Output	hw_vio_1
vio_reset	[B] 0		Output	hw_vio_1
core_in_loopback_i	[B] 0		Output	hw_vio_1
core_is_master_i	[B] 0		Output	hw_vio_1
line_rate_1	0		Output	hw_vio_1
line_rate_2	0		Output	hw_vio_1
line_rate_3	0		Output	hw_vio_1
line_rate_4	0		Output	hw_vio_1
pma_loopback	0		Output	hw_vio_1

Figure 3 HW VIO 1 (Control VIO)



Name	Value	Activity	Direction	VIO
speed5[4:0]	[U] 15		Input	hw_vio_2
clk_ok	●		Input	hw_vio_2
refclk_ok	●		Input	hw_vio_2
iic_done	●		Input	hw_vio_2
mmcm_locked	●		Input	hw_vio_2
local_lof	●		Input	hw_vio_2
local_los	●		Input	hw_vio_2
local_rai	●		Input	hw_vio_2
remote_lof	●		Input	hw_vio_2
remote_los	●		Input	hw_vio_2
remote_rai	●		Input	hw_vio_2
stat_alarm	●		Input	hw_vio_2
reset_acknowledge_out	[B] 0		Input	hw_vio_2
sdi_request_out	[B] 0		Input	hw_vio_2
barrel_shift_value[6:0]	[H] 1F		Input	hw_vio_2
fifo_transit_time[13:0]	[H] 20DB	↕	Input	hw_vio_2
coarse_timer_value[17:0]	[H] 0_0137		Input	hw_vio_2
rx_gb_latency_value[15:0]	[H] 0BF9	↕	Input	hw_vio_2
tx_gb_latency_value[15:0]	[H] 0514	↕	Input	hw_vio_2
r21_timer[35:0]	[S] 10	↕	Input	hw_vio_2
state_operate	●		Input	hw_vio_2
state_vs	●		Input	hw_vio_2
state_cm	●		Input	hw_vio_2
state_1sync	●		Input	hw_vio_2
state_reset	●		Input	hw_vio_2
state_proto	●		Input	hw_vio_2
state_passive	●		Input	hw_vio_2
frame_count[31:16]	[U] 27017	↕	Input	hw_vio_2
vs_word_count[31:16]	[U] 1688	↕	Input	hw_vio_2
eth_rx_frame_count[31:16]	[U] 3202	↕	Input	hw_vio_2
hdlc_bit_count[31:16]	[U] 3377	↕	Input	hw_vio_2
iq_error_count[15:0]	[U] 0		Input	hw_vio_2
vs_error_count[15:0]	[U] 0		Input	hw_vio_2
eth_error_count[15:0]	[U] 0		Input	hw_vio_2
hdlc_error_count[15:0]	[U] 0		Input	hw_vio_2

Figure 4 HW VIO 2 (Status VIO)

## Status VIO Signal Description

1. The counters, frame\_count, vs\_word\_count, eth\_rx\_frame\_count and hdlc\_bit\_count display the number of data packets correctly received by the example design interface modules.
2. The counters, iq\_error\_count, vs\_error\_count, eth\_error\_count and hdlc\_error\_count display the number of data packets incorrectly received by the example design interface modules. The example design modules will continue to transmit and receive data to the CPRI core on all the above interfaces while the core is in the operate state.

## Operating the Hardware Demonstration

### Read/Write Access to CPRI Core Management Registers Using VIO Console

The VIO Console allows read/write access to the CPRI core Management Interface registers.

#### To Perform a Register Read Access

1. Enter the required register address value (in hexadecimal) in the '*mgmnt\_addr*' section in the *hw\_vio\_1* window.
2. Pulse the '*mgmnt\_read\_req*' button high from the VIO window.
3. The data read from the addressed core is reflected in the '*mgmnt\_read\_data*' probe in the VIO window.

#### To Perform a Register Write Access

1. Enter the required register address value (in hexadecimal) in the '*mgmnt\_addr*' section in the VIO window.
2. Enter the required register write value (in hexadecimal) in the '*mgmnt\_write\_data*' section in the VIO window.
3. Pulse the '*mgmnt\_write\_req*' button high from the VIO window.
4. The requested write operation is carried out.

### Line rate shortcut buttons

On Ultrascale boards four line rate shortcut buttons are provided for convenience. *line\_rate\_1*, *line\_rate\_2*, *line\_rate\_3* and *line\_rate\_4* are mapped to the highest four line rates in each demo design as shown in Table 4.

Button	KCU105	VCU108 Master-Slave	VCU108 (64 bit)	ZCU102	VCU118
<i>line_rate_1</i>	10.1376 Gb/s	12.16512 Gb/s	24.33024 Gb/s FEC	12.16512 Gb/s	24.33024 Gb/s FEC
<i>line_rate_2</i>	9.8304 Gb/s	10.1376 Gb/s	12.16512 Gb/s FEC	10.1376 Gb/s	24.33024 Gb/s
<i>line_rate_3</i>	6.1440 Gb/s	9.8304 Gb/s	10.1376 Gb/s FEC	9.8304 Gb/s	
<i>line_rate_4</i>	4.9152 Gb/s	8.11008 Gb/s	8.11008 Gb/s FEC	8.11008 Gb/s	

Table 4 line rate shortcut button definitions

### PMA Loopback shortcut button

A PMA Loopback shortcut button is provided. When pressed this puts the core into PMA loopback. To take the core out of PMA loopback write 0x08 to register 0x20.

### Soft Reset shortcut button

A Soft Reset shortcut button is provided. When pressed this resets the core. This is the equivalent of writing logic 1 to management register 0x38 bit 31.

## Discrete Control/Alarm

In addition to accessing the management registers, some discrete control/alarm inputs to the core (Reset Req/Ack IN, SDI Req IN) can be directly controlled by the soft push buttons *vio\_reset\_req* and *sdi\_request\_in*, and the core response monitored in the Status VIO.

A full reset of the demo can be performed by the *vio\_reset* pushbutton in the *hw\_vio\_1* console which performs a full reset and start-up sequence of the overall design.

## Reset Pushbutton

One of the push buttons (SW7 - Centre push button) on the KCU105 board performs a reset of the CPRI core in the demo. On the VCU108 - SW5 CPU Reset push button is used, on the ZCU102 - SW18 North/Up push button is used and on the VCU118 – SW7 Centre push button is used.

## Slave Operation

When the CPRI core is configured as a master, it can be set to run as a slave by setting *core\_is\_master\_i* low in the *hw\_vio\_1* console. A reset is required for this change to have effect.

## Non Loopback Operation

It is possible to link together a master and a slave core implemented on different demonstration boards. To do this the slave should set *core\_in\_loopback\_i* low in the *hw\_vio\_1* console.

In this mode the reference clock to the CPRI slave core will initially be derived by the Si5328 from its 114.285 MHz reference input. When the recovered clock is stable, the Si5328 switches to use the recovered clock output from the FPGA. This synchronizes the slave clocking to the master.

**Note:** The Si5328 clock cleaner chip fitted to the KCU105, VCU108, ZCU102 and VCU118 boards is a general purpose chip and not optimized for a CPRI slave. Therefore the clock lock times when operating as a slave using the recovered clock may be as much as 5-10 seconds.

## CPRI Demonstration Options

The following section lists several CPRI demonstration options.

### Changing line rate

By writing a one hot pattern to the Line Speed Capability Register the CPRI core can be forced onto a specific line rate, i.e. only one rate is available.

Write 0000\_0001 to register 034 and observe *speed[4:0]* on the VIO status window has changed to 0.614Gbps line rate (1).

Read back the Current Line Speed Register (030) and observe it returns the line rate just written. This can be repeated for all line rates. The interface counters should all continue counting.

### Placing the CPRI core into PMA loopback

This is best demonstrated with the Tx -> Rx loopback cables disconnected.

A write of 0000\_0008 to register 020 (Transceiver Loopback and Ethernet Reset Request Register) will put the CPRI core into PMA loopback. At this point the CPRI core will achieve sync and link up. The VIO status window will show the core is in the operate state, frame counters will start counting and speed[4:0] will be the highest available line rate according to the capability register.

A write of 0000\_0000 to register 020 will take the core out of PMA loopback mode. If the Tx and Rx serial cables are not attached then the core will go to the l1\_sync state. Also frame counters will stop counting and speed[4:0] will continually cycle between all capable line rates as the core tries to link up at each line rate in turn.

If the core is placed back into PMA loopback it will link up at the next line rate in the cycle. The core will go back into the operate state, frame counters will restart and speed[4:0] will stop cycling.

### Cable pull

Connect the Tx -> Rx serial loopback SMA cables to the board as described in the section on Connecting the Demonstration Board. It is not necessary to put the CPRI core into PMA loopback as described above. In the VIO status window the core should be in the operate state, speed[4:0] will be fixed on the negotiated line rate and the frame counters should be counting up.

Disconnect one of the SMA connectors and observe the CPRI core has changed state to l1\_sync, speed[4:0] should be cycling between all available line rates and the frame counters should have stopped counting.

Re-connect the SMA connector to the board and observe the core has gone back into the operate state, speed[4:0] has fixed at the next available rate and the frame counters have started counting again.