AXI Chip2Chip v5.0

LogiCORE IP Product Guide

Vivado Design Suite

PG067 November 24, 2020

XILINX

Table of Contents

Introduction

The LogiCORE™ IP AXI Chip2Chip is a soft Xilinx IP core for use with the Vivado® Design Suite. The adaptable block provides bridging between AXI systems for multi-device System on-chip solutions. The core supports multiple device-to-device interfacing options and provides a low pin count, high performance AXI chip-to-chip bridging solution.

Features

- Supports AXI4 Memory Map interface data width of 32-bit, 64-bit and 128-bit.
- Supports optional AXI4-Lite data width of 32-bit
- Two interface choices:
  - Single Ended or Differential SelectIO™ interface
  - Aurora interface that provides AXI4-Stream interface to seamlessly integrate into the Aurora IP core
- Independent Master or Slave mode selection for AXI4 and AXI4-Lite interfaces
- Supports Common Clock or Independent Clock operations
- Supports multiple Width Conversion options for reduced I/O utilization
- Supports Link Detect FSM with deskew operation for the SelectIO™ interface
- Supports Link Detect FSM and implements Hamming SECDED error correction code (ECC) for Aurora interfaces
- Allows all five AXI4 channels to operate independently
- Supports an additional high-priority cut through channel for communicating interrupts
- Supports completion of the pending AXI transactions in case the link fails between chip2chip master and chip2chip slave
- Provides a dedicated high-priority internal channel for link status monitoring and reporting.

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family[1]	Versal™, UltraScale+™ Families, UltraScale™ Architecture, Zynq®-7000, Xilinx 7 series
Supported User Interfaces	AXI4, AXI4-Lite
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Verilog
Example Design	Verilog
Test Bench	Verilog
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver	N/A
Tested Design Flows[2]	
Design Entry	Vivado® Design Suite
Simulation	For support simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 54806
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado® IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The LogiCORE™ IP AXI Chip2Chip core functions like a bridge to seamlessly connect two devices over an AXI interface. The core transparently bridges transactions in compliance with AXI protocol specifications. The bridging function allows all AXI channels to operate independently by forwarding per-channel data and control information in compliance with AXI per-channel Valid-Ready handshake.

The optional AXI4-Lite functions in the core are independent of AXI4 Interface functions. The AXI4-Lite interface can be used for low-bandwidth access such as configuration/status registers of a peripheral Master/Slave.

Two instances of the AXI Chip2Chip core are required for performing the bridging function. Figure 1-1 provides block diagrams for both Master and Slave AXI Chip2Chip IP cores.

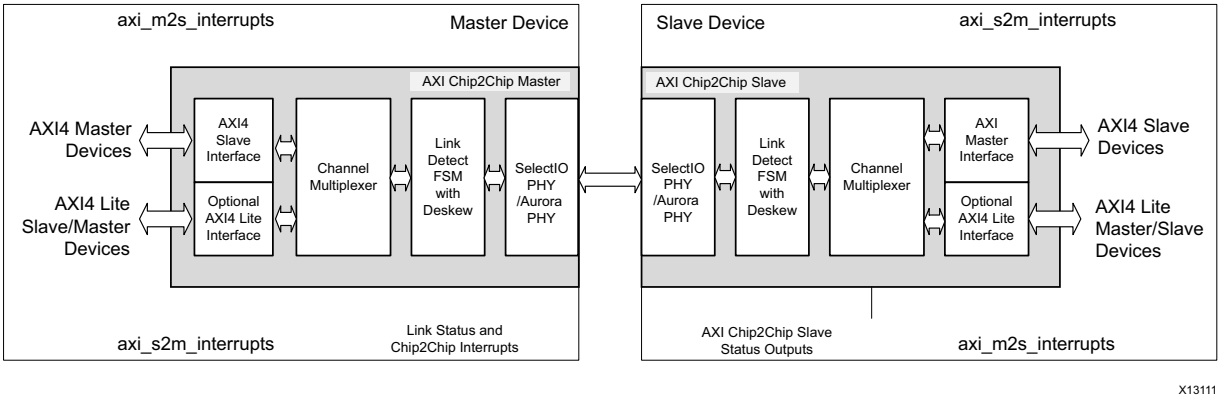

Figure 1-1: **AXI Chip2Chip Block Diagram**

The AXI Chip2Chip Master instance provides an AXI4 slave interface that can be directly connected to AXI Master or AXI interconnect devices. The AXI Chip2Chip Slave instance provides an AXI4 Master interface that can be connected to AXI Slave or AXI interconnect devices. The bridging functions in AXI Chip2Chip cores convert the wide on-chip AXI signaling to a compact device-to-device interfacing by utilizing a minimum set of device I/Os. The AXI Chip2Chip bridging also implements functions that provide error-free communication over the device I/Os.

The AXI4-Lite configuration option allows master or slave mode selection. For example, when the processor is connected to an AXI Chip2Chip Master instance, then an AXI4-Lite instance can be set to master mode; this setup will provide an AXI4-Lite slave interface. When peripheral Masters are connected to an AXI Chip2Chip Master instance, then an

AXI4-Lite instance can be set to slave mode and it will provide an AXI4-lite master interface. For more details on AXI4-Lite configuration options, see User Tab in Chapter 4.

AXI Chip2Chip operations can be categorized into five modules: AXI4 Interface, AXI4-Lite Interface, Channel Multiplexer, Link Detect FSM, and PHY interface.

Navigating Content by Design Processes

Hardware, IP and Platform Development

Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:

- Port Descriptions
- Clocking
- Resets
- Customizing and Generating the Core
- Example Design

Board System Design (PCB Design and System Bring-up)

Design a physical PCB board through a schematic and board layout. Power, thermal and signal integrity considerations, and debugging and testing.

- Clocking
- Resets
- Calibration and Link Error Detection
- Example Design

Feature Summary

This section summarizes the functionality of the core modules.

AXI4 Interface

The AXI Chip2Chip core provides an AXI4 interface to map to AXI Memory Mapped devices in the device general interconnect. AXI Memory Mapped devices can be AXI Master, AXI interconnect, or AXI Slave functions. The AXI Interface can operate in either Common Clock or in Independent clock modes. For more details on clocking and latencies, see General Design Guidelines in Chapter 3.

TIP: *The AXI4 interface of the Chip2Chip core provides WUSER signals to maintain compatibility with AXI3 interface specifications. Any AXI3 master that supports write interleaving can use the AXI4 WUSER[3:0] signals to map the WID[3:0] signals in AXI3 write data channel.*

AXI4-Lite Interface

The AXI4-Lite functions in the core are implemented with a shared address and data bus approach. This allows AXI4-Lite Master to accept a new write transaction only on completion of previous write transaction. This means it only accepts new writes on receiving a write response from the AXI4-Lite Slave. Similarly, the AXI4-Lite Master accepts a new read transaction only on completion of previous read transaction. This means it only accepts reads after receiving a read response and data from the AXI4-Lite Slave. For more details on the AXI4 Lite clocking and reset, see General Design Guidelines in Chapter 3.

Channel Multiplexer

The Channel Multiplexer multiplexes AXI Address and Data channels over FPGA I/Os. In addition, the AXI Chip2Chip core internally determines a 2:1 or 4:1 width conversion based on the Chip2Chip PHY Width option selected for the cores. Width conversion is used for reduced I/O utilization between the two devices. For more details on width conversion, see User Tab in Chapter 4.

The Channel Multiplexer also multiplexes AXI, AXI4-Lite and interrupt interfaces over the same set of FPGA I/Os. The priority round-robin multiplexing in the Chip2Chip core assigns the highest priority to interrupt signals, second highest priority to the low-bandwidth AXI4-Lite interface, and last priority to the AXI interface. The priority round-robin multiplexing is in effect when more than one of these interfaces are active simultaneously.

SelectIO Link Detect FSM with Deskew

The SelectIO™ Link Detect FSM with deskew operation ensures that the AXI Chip2Chip Master core initiates transactions only when both Master and Slave cores are out of reset and deskew patterns are exchanged without any bit errors. Deskew operations align data until an optimized sampling point is determined for the data. The nibble level deskew operation also enhances the maximum frequency of operation for the SelectIO™ interface. For more details on Link Detect FSM and deskew operations, see Calibration and Link Error Detection in Chapter 3.

Aurora Link Detect FSM

The Link Detect FSM ensures that the AXI Chip2Chip Master core initiates transactions only after the Aurora channel initialization is complete. For more details on Aurora Link Detect FSM, see Auto-Negotiation in Chapter 3.

SelectIO PHY Interface

The AXI Chip2Chip core provides the SelectIO FPGA interface as an interfacing option between the devices. The SelectIO provides minimum latency between the devices and provides SDR or DDR operations. When the SelectIO interface is used, the I/O type and I/O location must be specified in the Xilinx Design Constraints file (XDC).

Aurora PHY Interface

The AXI Chip2Chip core provides an optional AXI4-Stream interface to connect to the Aurora IP core (64B/66B or 8B/10B). AXI Chip2Chip core integration with Aurora IP can also be done through the Vivado IP integrator. The Aurora IP core supports the Xilinx® proprietary Aurora protocol layer and implements Xilinx Multi Gigabit Transceivers (MGTs) for high speed serial communications. To mitigate bit errors associated with a high speed serial interface, the AXI Chip2Chip core for this configuration implements a per-lane Hamming ECC code. The Hamming ECC module implements single-bit error correction and multiple (double) bit error detection (SECDED) functions. To connect the AXI Chip2Chip core with the Aurora IP core, the Aurora IP core should be configured in AXI4-Stream mode without CRC check.

The ECC module performs bit error correction for single-bit errors detected in the received data. When multiple bits are detected with errors, the Multibit Error interrupt signal is asserted.For more details on the clocking, reset and other signal connectivity with the Aurora core, see General Design Guidelines in Chapter 3.

Interrupt Signals

The AXI Chip2Chip core allows level interrupts to be communicated through a high-priority internal channel. Interrupts can be independently communicated between AXI Masters and AXI Slaves. On detecting a value change in the interrupt inputs, the AXI Chip2Chip Master core initiates a high-priority transfer to update the interrupt outputs of the AXI Chip2Chip Slave core. Similarly, on detecting a value change in the interrupt inputs, the AXI Chip2Chip Slave core initiates a high-priority transfer to update the interrupt outputs of the AXI Chip2Chip Master core. At system level, you should ensure that the interrupt level is not changed until it reaches the other end. For example, an interrupt that is set on Slave instance may be unset only when it has reached the master instance. Interrupts may not be transferred properly if they toggle at a faster rate.

The AXI Chip2Chip Master core also generates interrupts for link error conditions. Interrupt signals are asserted by the AXI Chip2Chip Master core. For this, the error conditions

detected in the AXI Chip2Chip Slave core are communicated to the Master device through a high-priority internal channel.

The following interrupt signals are supported in the AXI Chip2Chip Master core:

- **Link Error Interrupt**: Asserted when the AXI Chip2Chip Slave core is reset during normal operation. For more details on Link Error Interrupt, see Resets in Chapter 3.
- **Multibit Error Interrupt**: When asserted, a Multibit Error interrupt indicates multiple bits are received in error in the Master or Slave AXI Chip2Chip core. For the SelectIO™ interface, a multibit error is determined during deskew operations and indicates deskew operation failure. Multibit Error for the Aurora interface is determined by the ECC function.
- **Configuration Error Interrupt**: The Configuration Error Interrupt is set when a mismatch is detected between the Master and Slave AXI Chip2Chip core configurations.

  After being asserted, interrupt flags can be cleared only with a reset.

- **Graceful Exit in case of Link Failure**: The Chip2Chip core supports completion of pending AXI Transactions, when there is a link failure between Chip2Chip master and Chip2Chip slave (when the core is generated by enabling the **Enable Link Handler** option).

  Note: Select this feature only when there is a need to complete the pending AXI Transactions between master and slave (for example: Cable unplug between master / slave).

- **Link Handler Operation for Chip2Chhip Master Core:** Link handler keeps track of the AXI Transactions and performs the following actions when the link goes down:

  a. "*_awready" will be gracefully pulled to low, that is, no new requests will be accepted from the AXI Master.

  b. It accepts the exact number of pending data transactions and later the "*_wready" will be gracefully pulled to low.

  c. Link Handler sends out the write response as "Slave Error" for the pending transactions to the AXI Master.

  d. "*_arready" will be gracefully pulled to low, that is, no new requests will be accepted from the AXI Master.

  e. It the generates the remaining number of pending read transactions to the AXI Master. Once there are no-more pending read transactions the " *_rvalid" will be gracefully pulled low. Here, the Read Response will be the Error Response and the Read Data will be all zeros

- **Link Handler Operation for Chip2Chhip Slave Core:** Link handler keeps track of the AXI Transactions and performs the following actions when the link goes down:

  a. "*_awvalid" will be gracefully pulled to low, that is, no new write requests will be issued to the AXI Slave.

b. It generates the remaining number of pending write data transactions to the AXI Slave. Once there are no-more pending write transactions the "*_wvalid" will be gracefully pulled to low. The Write Data and the Strobe will be all zeros.

c. "*_arvalid" will be gracefully pulled to low, that is, no new read requests will be issued to the AXI Slave.

d. It accepts the exact number of pending data transactions and later the "*_rready" will be gracefully pulled to low, when there are no more pending requests.

For more information, see Link Handler Sequence.

Applications

Figure 1-2 shows an example of the AXI Chip2Chip use case with SelectIO™ PHY.

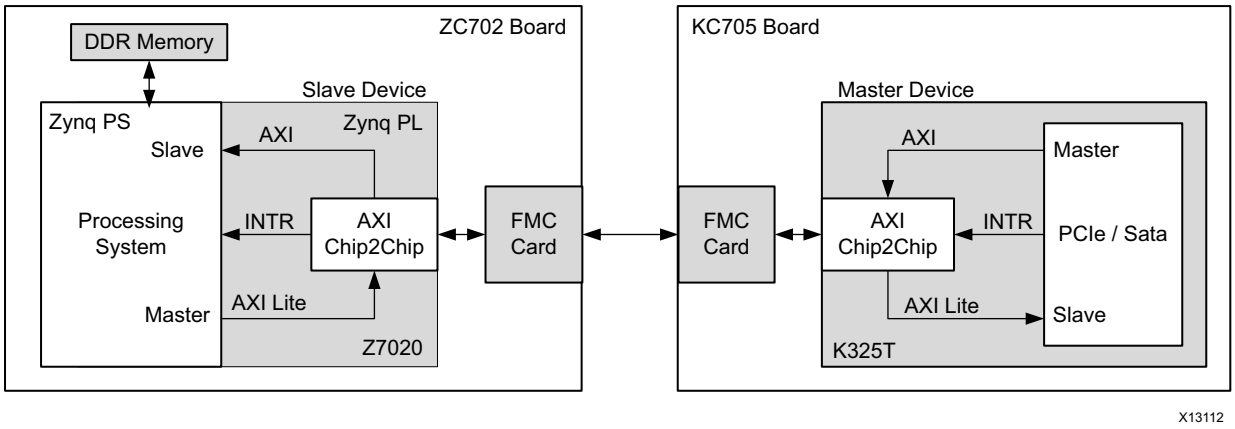

Figure 1-2: **AXI Chip2Chip Core Application Diagram**

In this use case, a Kintex®-7 device implementing a PCIe Peripheral Master is connected to a Zynq®-7020 device over an AXI Memory Mapped interface. Because it implements the Peripheral Master on the Chip2Chip AXI interface, the Kintex®-7 device is the Master device. Because it implements an AXI DDR Memory slave, the Zynq-7020 device is the Slave device. In this use case, the processing subsystem in the Zynq-7020 device uses the AXI4-Lite interface of the Chip2Chip core to access the control and status registers of the Peripheral Master in the Kintex®-7 device. The PCIe® Peripheral Master uses the AXI interface of the Chip2Chip core for writing and reading data from the DDR memory connected to the Zynq-7020 device. The PCIe Master in this case uses the Chip2Chip core interrupt signaling to trigger any PCIe interrupt service routines in the host processor.

Licensing and Ordering

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE™ IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards Compliance

This core has bus interfaces that comply with the *Arm® AMBA® AXI4 Protocol Specification Version 1.0*.

Performance Maximum Frequencies

The AXI Chip2Chip core is characterized based on the benchmarking methodology described in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5]. Table 2-1 shows the results of the characterization runs for 7 series devices.

IMPORTANT: *Maximum frequencies for UltraScale™ and Zynq®-7000 devices are expected to be similar to Kintex®-7 and Artix®-7 device numbers.*

Table 2-1: **Maximum Frequencies for 7 Series Devices**

Family	Speed Grade	F_{MAX} (MHz)		
		AXI4	AXI4-Stream	AXI4-Lite
Virtex7	–1	200	200	180
Kintex7		200	200	180
Artix7		150	150	120
Virtex7	–2	240	240	200
Kintex7		240	240	200
Artix7		180	180	140
Virtex7	–3	280	280	220
Kintex7		280	280	220
Artix7		200	200	160

Throughput and Latency

The AXI Chip2Chip core throughput is governed by the configured AXI Data Width, operating frequency of AXI interface, PHY interface and PHY Width mode (also called

compact ratio or muxing ratio). The following empirical formula provides guidance on the maximum theoretical throughput that the core can provide for the AXI Read/Write channel with a known overhead.

Throughput = ((1 – overhead factor) * AXIDataWidth * PHYFrequency) / MuxingRatio

The overhead factor is dependent on the PHY interface characteristics and PHY-specific Chip2Chip core overhead. The following documents contain more information about the PHY interface characteristics:

- *LogiCORE IP Aurora 8B/10B Product Guide* (PG046) [Ref 12]
- *LogiCORE IP Aurora 64B/66B Product Guide* (PG074) [Ref 13]
- *LogiCORE IP High Speed SelectIO Wizard Product Guide* (PG188) [Ref 14]
- *LogiCORE IP SelectIO Interface Wizard Product Guide* (PG070) [Ref 15]

The following example assumes the overhead factor is 0.25, the AXIDataWidth is 32, the PhyFrequency is 100, and the MuxingRatio is 1.

Throughput = ((1 – 0.25) * 32 * 100)/1 = 2400 Mbps

The above calculation assumes the AXI clock frequency and PHY clock frequency ratio is within 0.75 to 1.25 range for the Physical interface to be effectively utilized.

The MuxingRatio depends on the PHY width selection:

- MuxingRatio is 1 when PHY Width is selected as Compact 1–1
- MuxingRatio is 2 when PHY Width is selected as Compact 2–1
- MuxingRatio is 4 when PHY Width is selected as Compact 4–1

IMPORTANT: *The AXI Chip2Chip core should be configured so the theoretical throughput is higher than the average traffic sent over the link.*

Table 2-2 lists the latencies and throughput measurements on the AXI4 interface of the Chip2Chip Master core with SelectIO™ interface. The measurements were taken with simultaneous read and write operations. The measurement setup issued up to four AXI4 outstanding transactions. The AXI (system) clock frequency was set to 100 MHz, and ALEN was set to 16 beats. The measured latency can have up to 5-10% variation and does not account for system latencies outside of the AXI Chip2Chip core.

Table 2-2: **Throughput and Latency for AXI4 Interface of the AXI Chip2Chip Master Core**

Features			Latencies (AXI Clocks)		Throughput (Mb/s)	
AXI Data Width	PHY Width (Number of I/Os)	PHY Clock / PHY Type[(1)]	AW_Valid to B_Valid	AR_Valid to R_Valid	Write Data Channel	Read Data Channel
32-bit	Compact 4-1 (38)	200 MHz / SDR	77	44	1190	1280
	Compact 2-1 (32)	150 MHz / DDR	69	47	1770	1920
	Compact 1-1 (58[(2)])	100 MHz / DDR	57	42	2350	2550
64-bit	Compact 4-1 (28)	150 MHz / DDR	95	59	1780	1920
	Compact 2-1 (46)	100 MHz / DDR	77	51	2370	2560

Notes:

1. The number of I/Os is determined by the PHY Type and PHY Width configurations. See Table 4-2, page 33 for more details.
2. Common Clock mode of operation was selected for configurations having the same PHY clock and AXI clock frequencies (100 MHz).

Resource Utilization

For details about Resource Utilization, visit Performance and Resources Utilization web page

Port Descriptions

This section provides port descriptions for the AXI Chip2Chip core.

Global Signals

Table 2-3 describes the global signals for the AXI Chip2Chip core.

Table 2-3: **Global Interface Signals**

Name	Direction	Description
s_aclk	Input	Global Slave Interface Clock. For Independent Clock mode, all signals on the AXI Slave interface of an AXI Chip2Chip Master core are synchronous to s_aclk. For Common Clock mode, all AXI Chip2Chip Master core operations are synchronous to s_aclk.
axi_c2c_phy_clk	Input	Physical Interface Clock. The axi_c2c_phy_clk signal is applicable only when Independent Mode operation is selected for the core. AXI Chip2Chip Master core operations excluding the AXI Slave Interface are synchronous to axi_c2c_phy_clk.
idelay_ref_clk	Input	SelectIO™ Interface I/O Reference Clock. This signal is applicable only when the SelectIO™ interface is selected as the FPGA interfacing option. The applicable frequency for idelay_ref_clk is 200 MHz or 300 MHz (±10 MHz).
s_aresetn	Input	Global Reset. This signal is active-Low and asserted asynchronously. The de-assertion of this signal is synchronized with the clock domain. All applicable clock inputs to the AXI Chip2Chip Master core must be stable when s_aresetn input is deasserted.
m_aclk	Input	Global Master Interface Clock (Independent Clock). The m_aclk signal is an input when the Independent Clock mode of operation is selected for the core. Note: In Common Clock mode, the m_aclk input has to be connected to m_aclk_out driven by the AXI Chip2Chip Slave core.
m_aclk_out	Output	Global Master Interface Clock (Common Clock). The m_aclk_out signal is output when the Common Clock Mode of operation is selected for the core.
m_aresetn	Input	Global Reset. This signal is active-Low and asserted asynchronously. The de-assertion of this signal is synchronized with the clock domain. All applicable clock inputs to the AXI Chip2Chip Slave core must be stable when m_aresetn input is deasserted.
Optional AXI4-Lite Signals		
m_axi_lite_aclk	Input	Master Interface AXI4-Lite Clock. Applicable only when Slave Mode of AXI4-Lite is selected. All signals are sampled on the rising edge of this clock.
s_axi_lite_aclk	Input	Slave Interface AXI4-Lite Clock. Applicable only when Master Mode of AXI-Lite is selected. All signals are sampled on the rising edge of this clock.

AXI Interface Signals

Table 2-4 describes the AXI Interface signals for the AXI Chip2Chip core.

Table 2-4: **AXI Interface Signals**

Name	Direction	Description
s_axi_*	NA	AXI4 Slave Interface Signals. These signals are available only in master mode. See the *Vivado AXI Reference Guide* (UG1037)for AXI4 Slave signals [Ref 2].
m_axi_*	NA	AXI4 Master Interface Signals. These signals are available only in slave mode. See the *Vivado AXI Reference Guide* (UG1037)for AXI Master signals [Ref 2].
s_axi_lite_*	NA	AXI4-Lite Slave Interface Signals. These signals are available only when AXI4-Lite mode is set as Master. See the *Vivado AXI Reference Guide* (UG1037) for AXI4 Slave signals [Ref 2].
m_axi_lite_*	NA	AXI4-Lite Master Interface Signals. These signals are available only when AXI4-Lite mode is set as Slave. See the *Vivado AXI Reference Guide* (UG1037) for AXI Master signals [Ref 2].

Device Interface Signals

Table 2-5 describes the Master Device Interface signals for the AXI Chip2Chip core.

Table 2-5: **Device Interface Signals**

Name	Direction	Description
Single Ended SelectIO™ Interface		
axi_c2c_selio_tx_clk_out	Output	SelectIO™ FPGA interface clock from Master device to Slave device.
axi_c2c_selio_tx_data_out[m–1:0]	Output	SelectIO™ FPGA Interface Data from Master device to Slave device. 'm' is the number of Output I/Os required for Master-to-Slave device interfacing. For details, see User Tab in Chapter 4.
axi_c2c_selio_rx_clk_in	Input	SelectIO™ FPGA interface clock from Slave device to Master device.
axi_c2c_selio_rx_data_in[m–1:0]	Input	SelectIO™ FPGA interface signals from Slave device to Master device. 'm' is number of Input I/Os required for Slave to Master device interfacing. For details, see User Tab in Chapter 4.
Differential SelectIO™ Interface		
axi_c2c_selio_tx_diff_clk_out_p axi_c2c_selio_tx_diff_clk_out_n	Output	Select IO™ differential clock from Master to Slave device. Differential clocking is valid when C_USE_DIFF_CLK is set to 1.

Table 2-5: **Device Interface Signals *(Cont'd)***

Name	Direction	Description
axi_c2c_selio_tx_diff_data_out_p[m–1:0] axi_c2c_selio_tx_diff_data_out_n[m–1:0]	Output	SelectIO™ differential Data from Master to Slave device. m is the number of Output I/Os required for Master-to-Slave device interfacing. For details, see Chip2Chip PHY Width in Chapter 4. Differential data is valid when C_USE_DIFF_IO is set to 1.
axi_c2c_selio_rx_diff_clk_in_p axi_c2c_selio_rx_diff_clk_in_n	Input	Select IO™ differential clock from Slave to Master device. Differential clocking is valid when C_USE_DIFF_CLK is set to 1.
axi_c2c_selio_rx_diff_data_in_p[m–1:0] axi_c2c_selio_rx_diff_data_in_n[m–1:0]	Input	SelectIO™ differential data signals from Slave to Master device. m is number of Input I/Os required for Slave to Master device interfacing. For details, see User Tab in Chapter 4. Differential data is valid when C_USE_DIFF_IO is set to 1.
aurora_do_cc	Output	Clock compensation pattern generator signal used by aurora core. Asserted for every 10 µs intervals. This port is removed in aurora6466b_v10_0.This can be left open.
aurora_pma_init_in	Input	PMA initialization signal. Input signal that initializes the serial transceiver cells.
aurora_pma_init_out	Output	PMA initialization signal to Aurora IP. This signal is generated from the aurora_pma_init_in signal.
aurora_init_clk(1)	Input	Single-ended init_clk_out signal from the Aurora IP.
aurora_mmcm_not_locked	Input	MMCM locked indication from the Aurora IP. This signal indicates that the user clock from the Aurora IP is not stable.
aurora_reset_pb	Output	Reset signal for the Aurora block. This signal should be connected to the reset_pb signal of the Aurora IP.

Notes:

1. For UltraScale and UltraScale+ devices, this clock should be the same as init clock input for Aurora. For more details, see *LogiCORE IP Aurora 8B/10B Product Guide* (PG046) [Ref 12] and *LogiCORE IP Aurora 64B/66B Product Guide* (PG074) [Ref 13].

Interrupt and Status Signals

Table 2-6 describes the interrupt and status signals for the AXI Chip2Chip core.

Table 2-6: **Interrupt and Status Signals**

Name	Direction	Description
axi_c2c_link_status_out	Output	Link Status: Asserted when Link Detect FSM is in the SYNC state. Deasserted when either the Master or Slave AXI Chip2Chip core is under reset or when the Link Detect FSM is not in the SYNC state.
axi_c2c_link_error_out	Output	Link Error Interrupt: Asserted when the AXI Chip2Chip Slave core is reset during normal operations. This signal is valid only in Master mode.
axi_c2c_multi_bit_error_out	Output	Multi-bit Error Interrupt: When asserted, this interrupt indicates multiple bits are received with errors in the Master or Slave AXI Chip2Chip core. For the SelectIO™ interface, a multi-bit error is determined during deskew operations and indicates failure of those operations.
axi_c2c_m2s_intr_in	Input	Level Interrupt signaling from AXI Master to AXI Slave. The interrupt input should not toggle too frequently. It should stay high or low for a sufficient amount of time, for reliable transfer.
axi_c2c_s2m_intr_out	Output	Level Interrupt signaling from AXI Slave to AXI Master. The output values change only when a change in interrupt data is observed. If the interrupts on Slave instance changes too frequently, then Master instance may not be able to keep a track of it.
axi_c2c_m2s_intr_out	Output	Level Interrupt signaling from AXI Master to AXI Slave. The output values change only when a change in interrupt data is observed. If the interrupts on Master instance changes too frequently, then Slave instance may not be able to keep a track of it.
axi_c2c_s2m_intr_in	Input	Level Interrupt signaling from AXI Slave to AXI Master. The interrupt input should not toggle too frequently. It should stay high or low for a sufficient amount of time for reliable transfer.

Note: All the output ports belonging to the interrupt and status signals are synchronous to the *_aclk clock domain.

Designing with the Core

This chapter includes guidelines and additional information to make designing with the core easier.

General Design Guidelines

The customizable AXI Chip2Chip core provides multiple clocking and I/O interface options. You can determine the frequency at which the interface needs to be operated. Based on the interface frequency, you can select the I/O type by providing the appropriate constraints in the Xilinx constraints file (XDC). Selecting the SelectIO™ interface DDR option doubles the I/O speed without impacting the latency or performance. Based on the selection in the User Tab, additional internal width conversion stages can be enabled. Each 2:1 stage of width conversion can increase bridging latencies and can also impact performance.

In addition, you can select the common clock and independent clock operations. The common clock mode of operation reduces clock domain crossing latencies, and the independent clock mode provides additional clock conversion functionality. Both AXI Chip2Chip Master and AXI Chip2Chip Slave cores can be independently selected for either Common Clock or Independent clock operation with the exception of an AXI Chip2Chip Master core with Aurora interface. Operating the AXI Chip2Chip core at frequencies greater than AXI interface frequencies (Independent clock operation) reduces the bridging latencies and can improve overall performance of the AXI Chip2Chip bridging function.

Clocking

Figure 3-1 provides the clocking requirement for the SelectIO™ interface. In addition to AXI clocks, the deskew function, when enabled, requires an additional 200 MHz or 300 MHz (± 10 MHz) reference clock. Both AXI Chip2Chip Master and AXI Chip2Chip Slave cores can be independently selected for either Common Clock or Independent Clock operations. When the AXI Chip2Chip Slave core is selected for Common Clock operation, the core provides clock and reset (Link Status) to the interfacing slave AXI system.

When the AXI4-Lite interface is enabled, it always operates on an independent `axi_lite` clock input. The AXI4-Lite Master core operations are synchronous to `s_axi_lite_aclk`, and the AXI Lite Slave core operations are synchronous to `m_axi_lite_aclk`.

IMPORTANT: *All input clocks to the Master or Slave Chip2Chip cores must be stable when Reset input to the core is deasserted.*

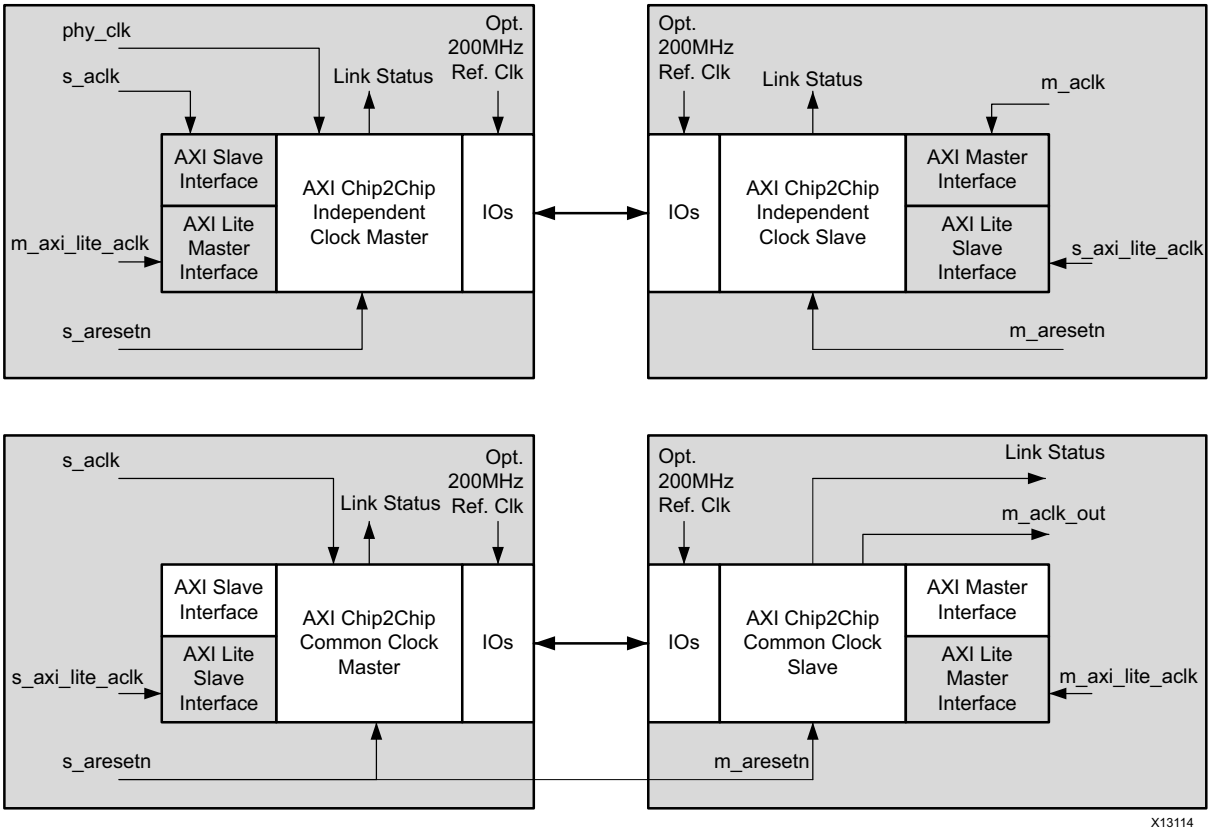

Figure 3-1: **Clocking with the SelectIO Interface**

Figure 3-2 shows the clocking, reset and interface connectivity with the Aurora IP core. In this example, the Aurora core requires differential GT reference clock (`gt_refclk`) inputs and differential free running clock (`init_clk`) inputs for core operations. The Aurora core provides the single-ended PHY clock (`user_clk`) and single ended `aurora_init_clk` to the AXI Chip2Chip core.

The stability of `user_clk` is validated by `mmcm_not_locked` output from the Aurora core. AXI Chip2Chip link-up operations are initiated only when Aurora output `mmcm_not_locked` is deasserted and `channel_up` is asserted. On link-up, the AXI Chip2Chip generates a pulse (stay alive pulse) on `do_cc` output once in every 10,000 clock cycles of free running `aurora_init_clk`.

The AXI Chip2Chip Master core with Aurora interface supports only independent clock mode. The AXI Chip2Chip Slave core supports both Common Clock and Independent clock modes. The `s_aclk` is always required as AXI input clocks for AXI Chip2Chip Master core. The `m_aclk` is always required as AXI input clocks for the AXI Chip2Chip Slave core. The `m_aclk_out` is provided as an additional clock output in Common Clock mode. The `m_aclk_out` output can be used as an AXI System Clock. In addition, it should be connected to `m_aclk` input of the AXI Chip2Chip Slave core.

In common clock operations, the AXI Chip2Chip Slave core connects the Aurora `user_clk` to the `m_aclk_out` output of the core. In this case, the `pma_init` reset must be handled externally, and asserting `pma_init` resets the MMCM generating the `user_clk`. It is also recommended to connect `pma_init` to the `m_aresetn` input of the AXI Chip2Chip Slave core in Common Clock mode.

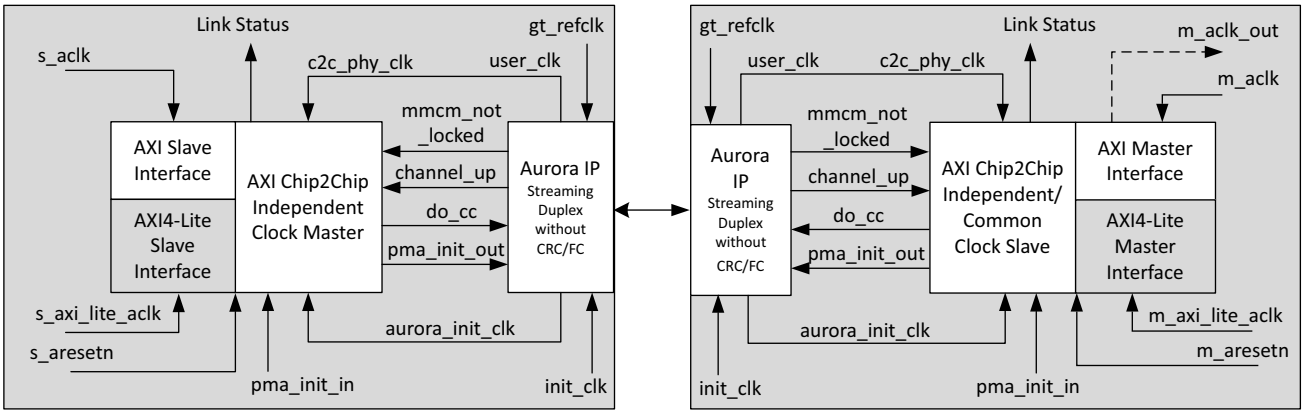

Figure 3-2: **AXI Chip2Chip Connectivity and Clocking with the Aurora IP Core**

Resets

The AXI Chip2Chip core allows both Master and Slave cores to have independent reset mapping. The link detect FSM ensures the transactions from the Master device (AXI4 and AXI4-Lite) get initiated only when both Master and Slave AXI Chip2Chip cores are out of reset and ready to accept transactions. Reset can also be propagated from Master device to Slave device. In this case, you need to map the reset from Master device to Slave device.

There is no separate reset for the AXI4-Lite interface. The AXI4-Lite interface is brought out of reset when the link detects FSM is in LINKUP state (when the link status output of the core is asserted). All input clocks to the core, including AXI4-Lite clock, must be stable when the core is brought out of reset (when `aresetn` core input is deasserted).

It is not recommended to reset either Master or Slave AXI Chip2Chip core during normal operation or when Link Status is asserted. For both SelectIO™ and Aurora interfaces, resetting the Master AXI Chip2Chip results in a link loss condition. If there is a link loss, both AXI Chip2Chip cores perform LINKUP operations to reestablish the link when the Master AXI Chip2Chip core is brought out of reset. However for the SelectIO™ interface, this reset sequence requires `s_aresetn` to be propagated from the Master Device to the Slave Device. When the Slave device is reset during normal operations, the link status is deasserted and a link error interrupt is asserted in the Master AXI Chip2Chip core.

An asserted `pma_init` input performs both a general interconnect and transceiver reset in the Aurora core, and an asserted `reset_pb` performs only the general interconnect reset in the Aurora core. The `pma_init` input is required to be connected to the `pma_init_in`

input of AXI Chip2Chip core. The `pma_init_out` from the AXI Chip2Chip core is required to be connected to the `pma_init` input of the Aurora core. To comply with the Aurora core recommendations for hot plug operations, an asserted `pma_init` (for a minimum of single pulse of `init_clk` width) causes the AXI Chip2Chip to assert the general interconnect reset to the Aurora core. A 26-bit hot plug counter overflow asserts the `pma_init` to the Aurora core. The general interconnect reset (`reset_pb`) to the Aurora core also gets asserted when an AXI reset is applied to the AXI Chip2Chip core.

Use the following steps for reset removal sequence for Chip2Chip cores:

1. Remove the master Chip2Chip core out of the reset.
2. Remove the slave Chip2Chip core out of the reset.

This ensures successful link bring-up for both Select IO and Aurora Configuration.

Note: The Chip2Chip core implicitly generates a reset sequence for the Aurora configuration, at power on.

Calibration and Link Error Detection

Table 3-1 shows a normal operation with link up in the AXI Chip2Chip core for SelectIO PHY interface.

Table 3-1: **Normal Operation with Link Up**

Master	Slave
When initialization starts, the Master sends ACK pattern for fixed number of times followed by CALIBRATION pattern and waits for Slaves response.	
	Slave continues to send INIT pattern. When it sees ACK pattern from Master, it confirms number of times ACK pattern is repeated and goes into self-calibration process. If self-calibrated, Slave sends ACK pattern to confirm.
After receiving ACK pattern from Slave, Master sends NACK pattern.	
	After receiving NACK pattern from Master, Slave sends CALIBRATION pattern to Master.
Master continues to send NACK pattern and goes into self-calibration process. If self-calibrated, Master sends ACK pattern to confirm.	
	Slave responds with ACK pattern and moves to data transfer state and asserts link_status.
After receiving ACK pattern from Slave, Master moves to data transfer state and asserts link_status.	

Table 3-2 shows a calibration failure in the Slave device in the AXI Chip2Chip core for SelectIO PHY interface.

Table 3-2: **Calibration Failure in Slave Device**

Master	Slave
When initialization starts, Master sends ACK Pattern for fixed number of times followed by CALIBRATION pattern and waits for Slave's response.	
	Slave continues to send INIT pattern. When it sees ACK pattern from Master, it confirms number of times ACK pattern is repeated and goes into self-calibration process. If unable to self-calibrate, Slave sends NACK pattern, goes into Error state, and asserts multi-bit error interrupt to indicate calibration failure.
After receiving NACK pattern from Slave, Master goes into Error state, asserts multi-bit error interrupt to indicate calibration failure.	

Table 3-3 shows a calibration failure in the Master device in the AXI Chip2Chip core for SelectIO™ PHY interface.

Table 3-3: **Calibration Failure in Master Device**

Master	Slave
When initialization starts, Master sends ACK Pattern for fixed number of times followed by CALIBRATION pattern and waits for Slave's response.	
	Slave continues to send INIT pattern. When it sees ACK pattern, it confirms number of times ACK pattern is repeated and goes into self-calibration process. If self-calibrated, Slave sends ACK pattern to confirm.
After receiving ACK pattern from Slave, Master sends NACK pattern.	
	After receiving NACK pattern from Master, Slave sends CALIBRATION pattern to Master.
Master continues to send NACK pattern and goes into self-calibration process. If unable to self-calibrate, Master goes into Error state, sends INIT pattern, and asserts multi-bit error interrupt to indicate calibration failure.	
	After receiving INIT pattern from Master, Slave moves to Error state and asserts multi-bit error interrupt to indicate calibration failure.

Note: Link error recovery is possible only after going through reset cycle.

IMPORTANT: *A Master or Slave device can fail in self-calibration if there is a physical or transient fault on the link or if there is physical connection error.*

The AXI Chip2Chip core implements Link Detect FSM for device detection and calibration functions for the SelectIO PHY interface. The calibration process is initiated when either the Master or Slave AXI Chip2Chip is brought out of reset. During the calibration process, a fixed set of patterns are exchanged between the Master and Slave devices. The receiving device responds with an appropriate pattern if received patterns do not match the expected fixed patterns. Deskew operations align the data until an optimized sampling point is determined. After the patterns are determined to match for the greatest number of the sampling points, the receiving device responds with an ACK. This operation is performed at nibble level for 32 sampling points. The best sampling point is determined for each nibble in the data. Link status is asserted after both Master and Slave devices respond with an ACK. The Link Failure (`axi_c2c_multi_bit_error_out`) signal is asserted when a multi-bit error is determined during deskew operations and indicates the failure of those operations. In this case, either the interface rate can be reduced or I/O Type can be appropriately selected to achieve the required interface rate.

When the Link Status signal is asserted, the AXI Chip2Chip core transparently bridges transactions in compliance with AXI protocol specifications. It is not recommended to reset or disconnect either the Master or Slave AXI Chip2Chip core during normal operation or when the Link Status signal is asserted. When the Slave device is reset or if the cable is disconnected during normal operations, the Link Status signal is deasserted and a link error interrupt is asserted in the Master device. After being asserted, a link error interrupt can be cleared only with a reset. The AXI Chip2Chip core operations are re-initiated when the Master and Slave AXI Chip2Chip devices are brought out of reset.

Auto-Negotiation

The AXI Chip2Chip core implements Link Detect FSM for auto-negotiation operations when using an Aurora PHY interface. During auto-negotiation, the capabilities of the AXI Chip2Chip Master and Slave cores are matched before the link layer is enabled. After the Aurora channel is up, the Master and Slave cores exchange patterns (based on the respective configurations) with each other. The Link status signal is asserted when the configurations of the Master and Slave cores match. Otherwise, the Configuration Error status signal is asserted.

Table 3-4 shows a normal operation with link up in the AXI CHip2Chip core, for the Aurora Mode.

Table 3-4: **Normal Operation in Aurora Mode**

Master	Slave
When the Master receives channel up input as High, it starts the initialization process. It sends a PAT0 pattern for a fixed number of times and expects the Slave to respond with the same pattern within a specified interval.	
	When the Slave receives channel up input as High, it starts the initialization process. It sends a PAT0 pattern repeatedly until it sees a matching pattern from Master for a fixed number of times.
The Master sends a PAT1 pattern for a fixed number of times and expects the Slave to respond with a matching pattern for the same number of times within a specified interval.	
	The Slave sends a PAT1 pattern repeatedly until it sees a matching pattern from the Master for a fixed number of times.
After receiving the PAT1 pattern sequence, the Master indicates it is ready for data transfer. The Master checks channel up status in this state. If channel up is deasserted, the Master deasserts link status and goes to initial state. If channel up is High, the Master asserts link status and starts the data transfer.	
	After receiving the PAT1 pattern sequence, the Salve indicates it is ready for data transfer. The Slave checks channel up status in this state. If channel up is deasserted, the Slave goes to the initial state. If channel up is High, it asserts link status and starts the data transfer.

Table 3-5 shows a configuration error in the AXI Chip2Chip core, for the Master Aurora Mode.

Table 3-5: **Configuration Error in Master Aurora Mode**

Master	Slave
When the Master receives a channel up input as High, it starts the initialization process. It sends a PAT0 pattern for a fixed number of times and expects the Slave to respond with the same pattern within a specified interval.	
	When the Slave receives channel up input as High, it starts the initialization process. It sends a PAT1 pattern repeatedly until it sees matching pattern from the Master for fixed number of times.
If the Master does not receive the expected pattern within a specified interval, it asserts the configuration error status signal. If the Master does receive the expected pattern, it sends a PAT1 pattern for a fixed number of times and expects the Slave to respond with a matching pattern for the same number of times within a specified interval.	
	The Slave sends a PAT1 pattern repeatedly until it sees a matching pattern from the Master for fixed number of times.
If the Master does not receive the expected pattern within a specified interval, it asserts the configuration error status signal. Otherwise, it continues the normal link up operation.	

Note: A Master device in Aurora mode can assert the configuration error status signal. An asserted configuration error status signal indicates the Link Detect FSM failed due to a configuration mismatch of Master and Slave AXI Chip2Chip cores. Recovery from a configuration error is only possible after going through a reset cycle.

Link Handler Sequence

When the Chip2Chip Core is enabled with the Link Handler it handles the AXI transactions gracefully when the link goes down.

Use the following steps when the Link Handler is enabled:

1. Monitor the output port "axi_c2c_lnk_hndlr_in_progress". When the link goes down, "axi_c2c_lnk_hndlr_in_progress" goes high, indicating that the Link Handler is operating and it is handling the AXI transactions.
2. When the Link Handler is in operation, the responses are sent as error responses and the data integrity is not maintained.
3. Once the Link Handler completes all the pending transactions, the "axi_c2c_lnk_hndlr_in_progress" goes low.
4. On detecting the "axi_c2c_lnk_hndlr_in_progress" going low, reset the Chip2Chip Cores as per the Chip2Chip reset sequence.
5. After the reset is removed, the Chip2Chip Core will function normally.

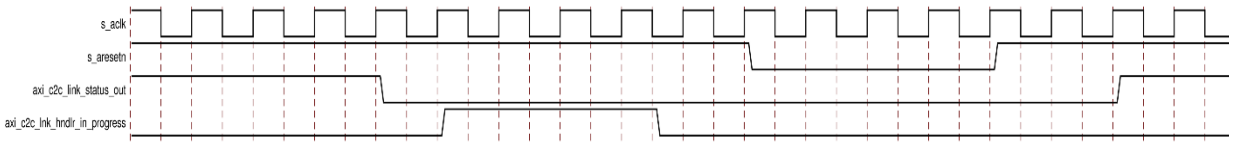

Figure 3-3: **Link Handler Sequence**

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 9]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 8]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 3]

Customizing and Generating the Core

This section contains information and instructions for using the Vivado Design Suite to customize the LogiCORE™ IP AXI Chip2Chip core.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 8].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

Figure 4-1 shows the Vivado IDE for the AXI Chip2Chip core. The options are described following the figure.

User Tab

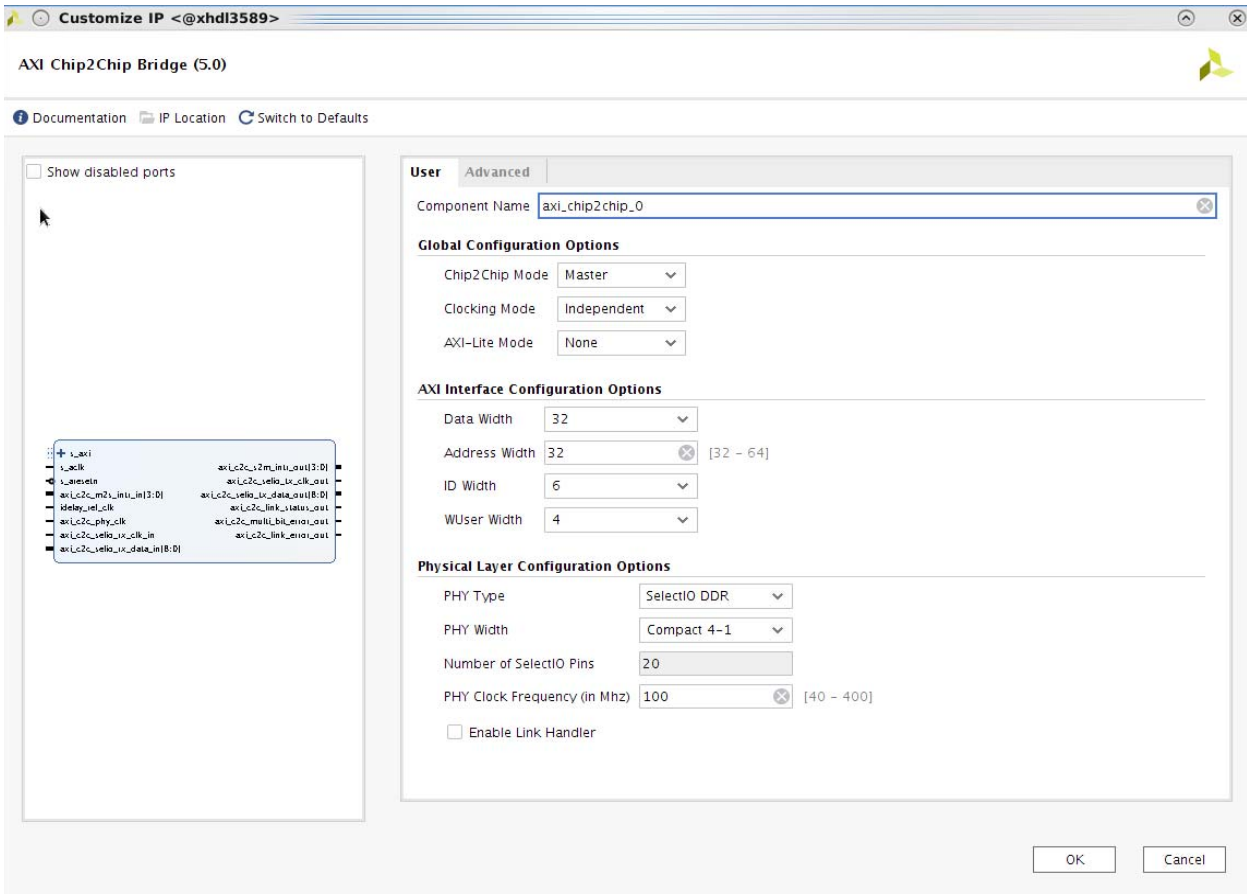

Figure 4-1: **Customization Vivado IDE for the AXI Chip2Chip Core**

- **Chip2Chip AXI Mode**: The Chip2Chip AXI Mode configuration option determines AXI Chip2Chip Master or Slave mode of operation.
- **AXI Clocking Mode**: The AXI Chip2Chip core can be configured with either Independent or Common Clock domains.

  The Independent Clock configuration allows you to implement unique clock domains on the AXI interface and FPGA I/Os. The AXI Chip2Chip core handles the synchronization between clock domains. Both the AXI interface and FPGA I/Os can also be maintained in a single clock domain. The AXI Chip2Chip core can be used to generate a core optimized for a single clock by selecting the Common Clock option.

- **Chip2Chip AXI4-Lite Mode**: The Chip2Chip AXI4-Lite Mode configuration option determines AXI4-Lite Master or Slave mode of operation, as shown in Table 4-1. When AXI4-Lite interfacing is not required, this configuration option should be set to "None."

Table 4-1: **AXI4-Lite Configuration Options**

Chip2Chip Mode		Chip2Chip AXI4-Lite Options	
Mode	**AXI4 Interface**	**Mode**	**AXI4-Lite Interface**
Master FPGA	Slave	Master	Slave
		Slave	Master
		None	None
Slave FPGA	Master	Master	Slave
		Slave	Master
		None	None

- **AXI Data Width**: The AXI Data Width user option allows the width of AXI data to be configured. Valid settings for the AXI Data Width are 32, 64 and 128. This setting must be maintained the same in both Master and Slave AXI Chip2Chip cores.
- **AXI ID Width**: The AXI ID provides an identification tag for the group of signals in the channel. AXI ID is supported for all write and read channels. ID width can be configured from 0 to 12 bits. This setting must be maintained the same in both Master and Slave AXI Chip2Chip cores.
- **AXI WUSER Width**: AXI WUSER defines sideband information that can be transmitted with the write data channel. The valid range for WUSER width is from 0 to 4 bits. This setting must be maintained the same in both Master and Slave AXI Chip2Chip cores.

IMPORTANT: *Because the AXI Chip2Chip core supports a maximum ID width of 12, ensure that the propagated ID width to the AXI Chip2Chip core is less than or equal to 12. This commonly happens in Zynq®-7000 device systems because the ID width of GP ports is 12. To avoid this scenario, the ID widths of the GP ports can be compressed by modifying the Static Remap option available in the processing system.*

TIP: *The AXI ID Width of the AXI Chip2Chip Slave core should match the AXI ID Width of the AXI Chip2Chip Master core.*

IMPORTANT: *In IP integrator, the AXI ID and WUSER Width of the interconnect are automatically propagated to the AXI Chip2Chip Master core. However for the AXI Chip2Chip Slave core, you have to override the AXI ID Width and WUSER Width so that it matches the parameters of the Master AXI Chip2Chip core.*

- **Chip2Chip PHY Type**: The Chip2Chip PHY type can be set to either “SelectIO™ SDR”, “SelectIO™ DDR”, "Aurora64B66B" or “Aurora 8B/10B”. This setting must be maintained the same in both Master and Slave AXI Chip2Chip cores.

  The AXI Chip2Chip IP does not instantiate an Aurora core, but it does provide an interface to connect to it. Be sure to select the right device when simulating,

synthesizing, and implementing the example design of AXI Chip2Chip with the PHY Type set as Aurora.

- **Chip2Chip PHY Width**: The Chip2Chip PHY Width configuration determines I/Os used for device-to-device SelectIO™ interfacing. This setting must be maintained the same in both Master and Slave AXI Chip2Chip cores. Table 4-2 provides the mapping between Chip2Chip PHY width and the number of input and output I/Os utilized with the selected option.

Table 4-2: **FPGA SelectIO Utilization**

AXI Data Width	Chip2Chip PHY Type(1)	Chip2Chip PHY Width	Number of Output I/Os	Number of Input I/Os
32	SelectIO™ SDR	Compact 4:1(2)	19	19
		Compact 2:1	31	31
	SelectIO™ DDR	Compact 4:1(2)	10	10
		Compact 2:1	16	16
		Compact 1:1	29	29
64	SelectIO™ SDR	Compact 4:1(2)	26	26
		Compact 2:1	45	45
	SelectIO™ DDR	Compact 4:1(2)	14	14
		Compact 2:1	23	23
		Compact 1:1	42	42

Notes:

1. SelectIO PHY interface routes the clock with the data pins.
2. Compact 4:1 is not supported when the AXI4-Lite Interface is enabled for the core.

- **Chip2Chip PHY Frequency**: When using the SelectIO™ FPGA interface, the Chip2Chip PHY implements the mixed-mode clock manager (MMCM) on the PHY input clocks. MMCMs are used for clock phase alignment, clock slew reduction, and for compensating clock buffer delays. For common clock AXI Chip2Chip Slave operations, the `m_aclk_out` output is generated from the MMCM. The Chip2Chip PHY Frequency provides the clock frequency parameter to the MMCM.

  For Common clock, C_SELECTIO_PHY_CLK must be set to the `s_aclk` frequency. For Independent clock, C_SELECTIO_PHY_CLK must be to set to the `axi_c2c_phy_clk` frequency. This setting must be maintained the same in both Master and Slave AXI Chip2Chip cores.

IMPORTANT: *In IP integrator, the PHY Frequency parameter is automatically computed based on the clock frequency of the port connected to `axi_c2c_phy_clk` (Master Independent clocking configuration) or `axi_c2c_selio_rx_*_clk_in*` port(s) (Slave configuration). In Master Common clocking configuration, the frequency of the connected AXI clock is propagated to the PHY Frequency parameter.*

- **No. of Lanes**: Number of Lanes to be selected in Aurora IP when configuring the C2C Core with Aurora Mode.
- **Enable Link Handler**: By enabling this option the core will handle the graceful exit of the Pending AXI Transactions. When it is selected there will be an additional port 'axi_c2c_lnk_hndlr_in_progress'.
- **Number of Aurora Lanes**: When the interface is Aurora 64B/66B, the number of Aurora lanes parameter shows the This parameter is calculated with the AXI data width and Chip2Chip PHY width.

Advanced Tab

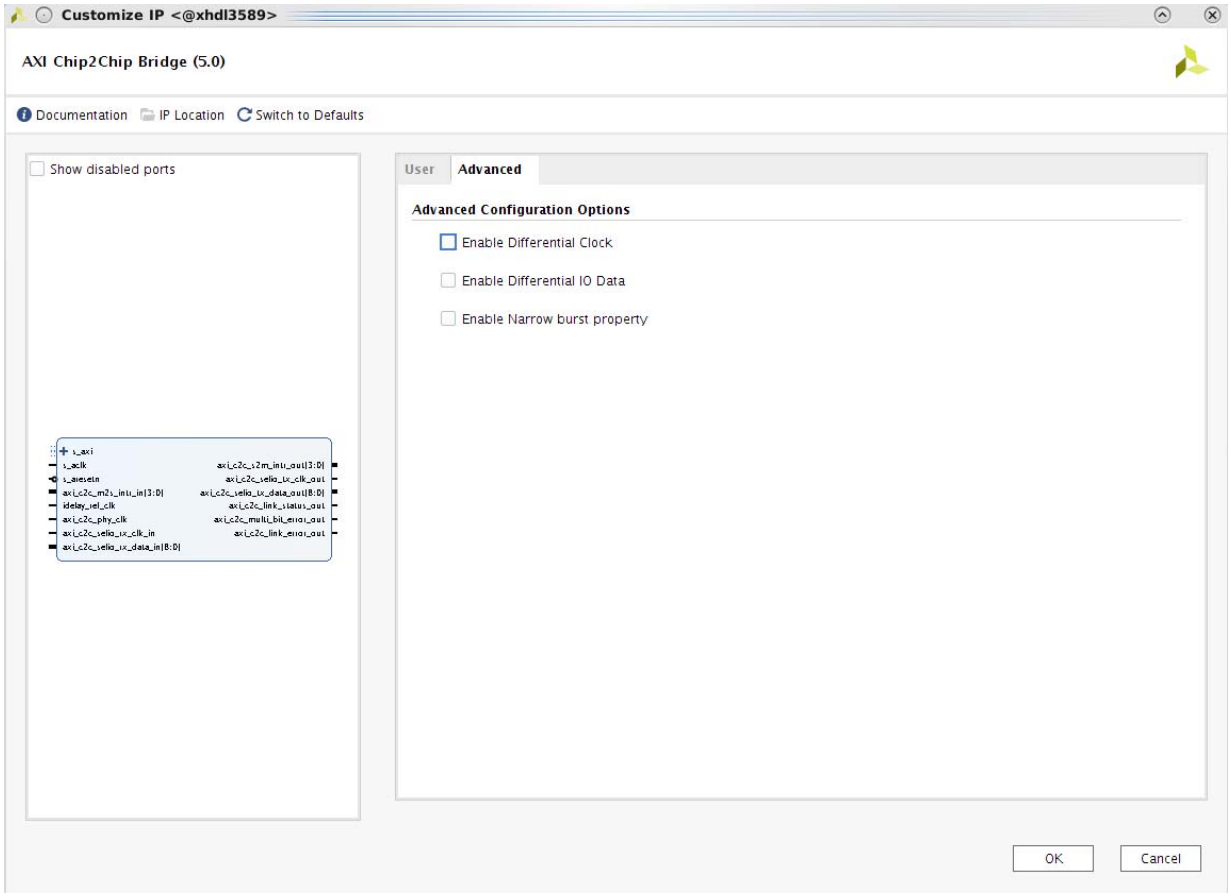

Figure 4-2: **Vivado IDE for Advanced AXI Chip2Chip Core Parameters**

Figure 4-2 shows the Vivado IDE for advanced AXI Chip2Chip core parameters. This tab includes the following options for the SelectIO™ FPGA interface:

- **Enable Differential Clock**: When set to 1, implements differential I/O buffer on the two clocks I/Os used for device interfacing. This setting must be maintained the same in both Master and Slave AXI Chip2Chip cores.

- **Enable Differential IO Data**: When set to 1, implements differential I/O buffer on the data I/Os used for device interfacing. This setting must be maintained the same in both Master and Slave AXI Chip2Chip cores.
- **Enable Narrow Burst Property**: Setting simply sets the Narrow Bus attribute on the AXI4 bus interface. This does not affect the IP operation.

User Parameters

Table 4-3 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-3: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value[(1)]	User Parameter/Value[(1)]	Default Value
AXI Mode	C_MASTER_FPGA - 1 - 0	1
Clocking mode -Independent -Common	C_COMMON_CLK - 0 - 1	0
AXI-Lite Mode - None - Master AXI-Lite - Slave AXI-Lite	C_INTERFACE_TYPE - 0 - 1 - 2	0
Data Width - 32 - 64 - 128	C_INTERFACE_TYPE - 32 - 64 - 128	32
Address Width Range (32 to 64)	C_AXI_ADDR_WIDTH Range (32 to 64)	32
ID width Range (0 to 24)	C_AXI_ID_WIDTH *Applicable for master mode C_M_AXI_ID_WIDTH *Applicable for slave mode Range (0 to 24)	6
WUSER Width Range (0 to 4)	C_AXI_WUSER_WIDTH *Applicable for master mode C_M_AXI_WUSER_WIDTH *Applicable for slave mode Range (0 to 4)	4
PHY Type - SelectIO™ SDR - SelectIO™ DDR - AURORA6466B - AURORA8B10B	C_INTERFACE_TYPE - 0 - 1 - 2 - 3	1

Table 4-3: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
PHY Clock Frequency (in Mhz) Range (40 to 400)	C_SELECTIO_PHY_CLK Range (40 to 400)	100
Enable Differential Clock - false - true	C_USE_DIFF_CLK - 0 - 1	0
Enable Differential IO Data - false - true	C_USE_DIFF_IO - 0 - 1	0
Enable Link Handler - false - true	C_EN_AXI_LINK_HNDLR - 0 - 1	0
For faster example design simulation - false - true	C_SIMULATION	0

Notes:

1. The Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Simulation Speed Up

The `C_SIMULATION` parameter is introduced to speed up the example design simulations with Aurora 8B10B and Aurora 64B66B configuration.

- During the IP core generation, include the following tcl command to the dict as part of the core generation.
  ```
  c_simulation true
  ```

Note: This mode of IP core generation is only for simulation purpose. Do not add this command as part of the IP core generation if you intend to test it on the board.

Output Generation

For specifics about files created when the core is generated, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Constraining the Core

This section contains details about constraining the core.

Required Constraints

The physical layer is a set of SelectIO™ interface pins that carry source synchronous clock with the data pins.

These I/O pins need I/O Location and I/O Standard constraints. These constraints are board specific and needs to be specified accordingly in the top-level XDC.

Clock Frequencies

The recommended frequency for the AXI interface is up to 200 MHz. For the maximum frequency numbers achieved on the SelectIO PHY interface, see Table A-1 in Appendix A, Verification, Compliance, and Interoperability. The clocking mode for the AXI Chip2Chip core needs to be set based on the AXI Interface Frequency and the required SelectIO interface PHY frequency. The required clocking constraints for the AXI Chip2Chip core are listed below:

- **s_aclk**: The AXI interface of the AXI Chip2Chip Master core operates in the `s_aclk` clock domain.
- **axi_c2c_phy_clk**: `axi_c2c_phy_clk` is the SelectIO interface PHY clock and is applicable when the AXI Chip2Chip Master core is configured in Independent Clock mode. For Common Clock mode, this clock constraint is not required because the PHY clock is the same as `s_aclk`. When using the Aurora interface, the `axi_c2c_phy_clk` of the AXI Chip2Chip Master and Slave cores should be constrained with the frequency of the `USER_CLK` output of the Aurora core.
- **m_aclk**: The AXI interface of the AXI Chip2Chip Slave core operates in the `m_aclk` clock domain.
- **s_axi_lite_aclk**: AXI4-Lite Master Mode operates in the `s_axi_lite_aclk` clock domain.
- **m_axi_lite_aclk**: AXI4-Lite Slave Mode operates in the `m_axi_lite_aclk` clock domain.
- **idelay_ref_clk**: Both the master and slave AXI Chip2Chip cores utilize the IDELAY_CTRL block for SelectIO PHY calibration. The `idelay_ref_clk` input is the reference clock to the IDELAY_CTRL block. This clock is 200 MHz or 300 MHz (± 10MHz) based on the selected device.
- **axi_c2c_selio_rx_clk_in**: `axi_c2c_selio_rx_clk_in` is the source synchronous clock of the SelectIO physical layer. This clock pin must be constrained with the PHY clock frequency. When Common Clocking mode is used, this clock runs at the same frequency as `s_aclk`.

Clock Management

The AXI Chip2Chip core utilizes the MMCM module to recover the SelectIO PHY clock. The frequency of the PHY clock is specified by setting the C_SELECTIO_PHY_CLK parameter.

Clock Placement

The clock input pins on the physical layer must be placed on clock-capable I/Os only.

Banking

Device-specific banking rules for placement of PHY I/O pins need to be considered when specifying the top-level XDC.

I/O Standard and Placement

The I/O pins of the AXI Chip2Chip core need I/O Location and I/O Standard constraints. These constraints need to be specified in the top-level XDC.

Simulation

This section contains information about simulating IP in the Vivado Design Suite. For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the Vivado Design Suite User Guide: Logic Simulation (UG900) [Ref 3].

IMPORTANT: *For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

Overview

Figure 5-1 shows the configuration of the example design with a SelectIO™ interface. For an Aurora interface, connect the streaming interface of the AXI Chip2Chip core to the Aurora core, as shown in Figure 5-2.

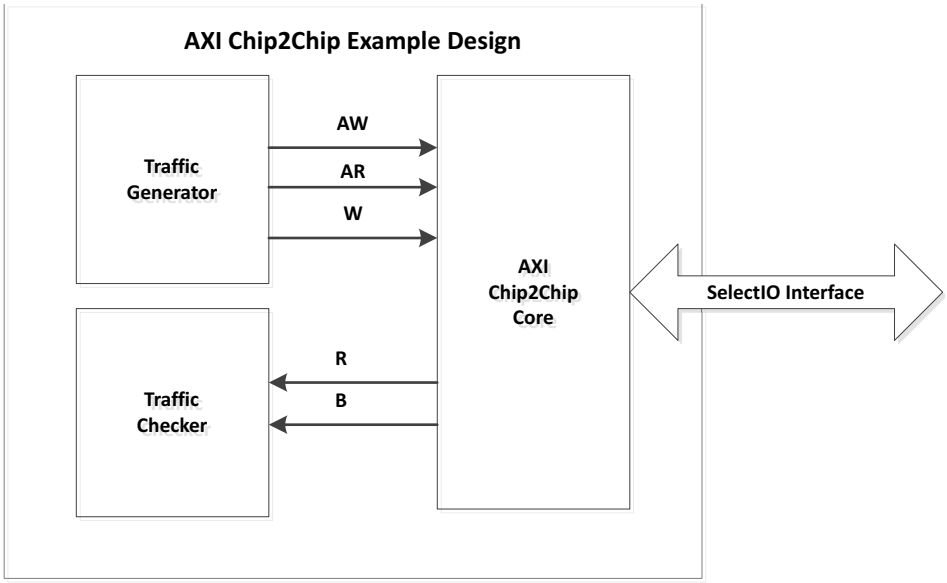

Figure 5-1: **Example Design Block Diagram**

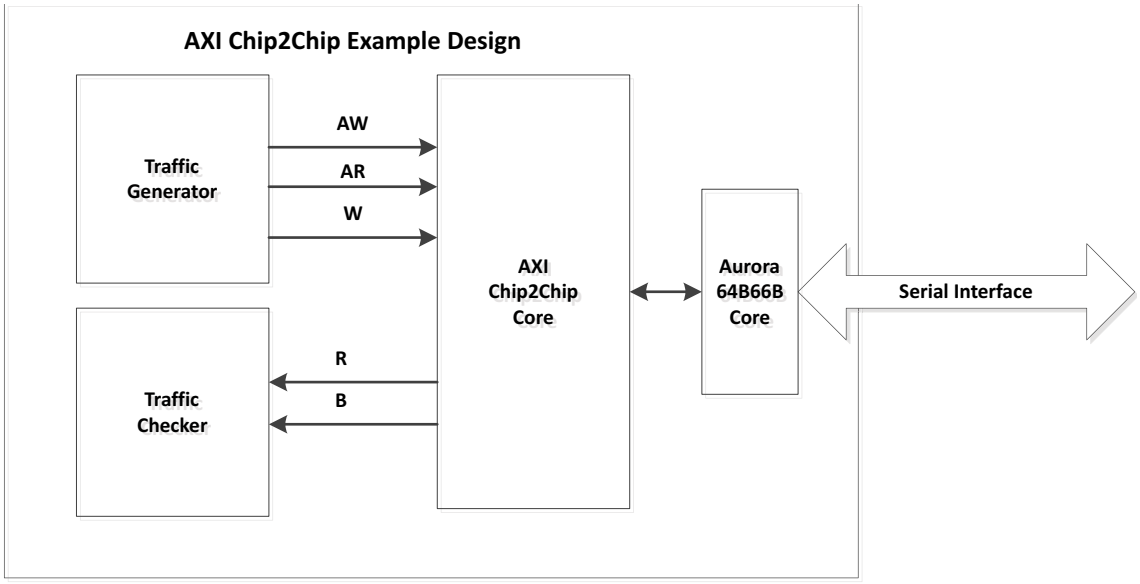

Figure 5-2: **Example Design Block Diagram with Aurora Interface**

The example design contains the following:

- An instance of the AXI Chip2Chip core
- Clocking wizard to generate clock signals for the example design
- Traffic generator for AXI4 and AXI4-Lite interfaces
- Traffic checker for AXI4 and AXI4-Lite interfaces
- an instance of the Aurora 64B66B core in duplex configuration.

IMPORTANT: *Be sure to select the right device when simulating, synthesizing, and implementing the example design of the AXI Chip2Chip with PHY type set as Aurora.*

Implementing the Example Design

Depending on the board selected, provide XDC constraints for the system clock pins and SelectIO pins of AXI Chip2Chip core. The status signals (Link Status, Multi-Bit Error, and Link Error) can be mapped to LEDs to show the status of the AXI Chip2Chip cores.

See the *AXI Chip2Chip Reference Design for Real-Time Video Application* (XAPP1160) [Ref 1] to set the SelectIO pin constraints for AXI Chip2Chip core on KC705 board.

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite.

Figure 6-1 and Figure 6-2 show the demonstration test bench with a SelectIO™ interface and an Aurora interface, respectively.

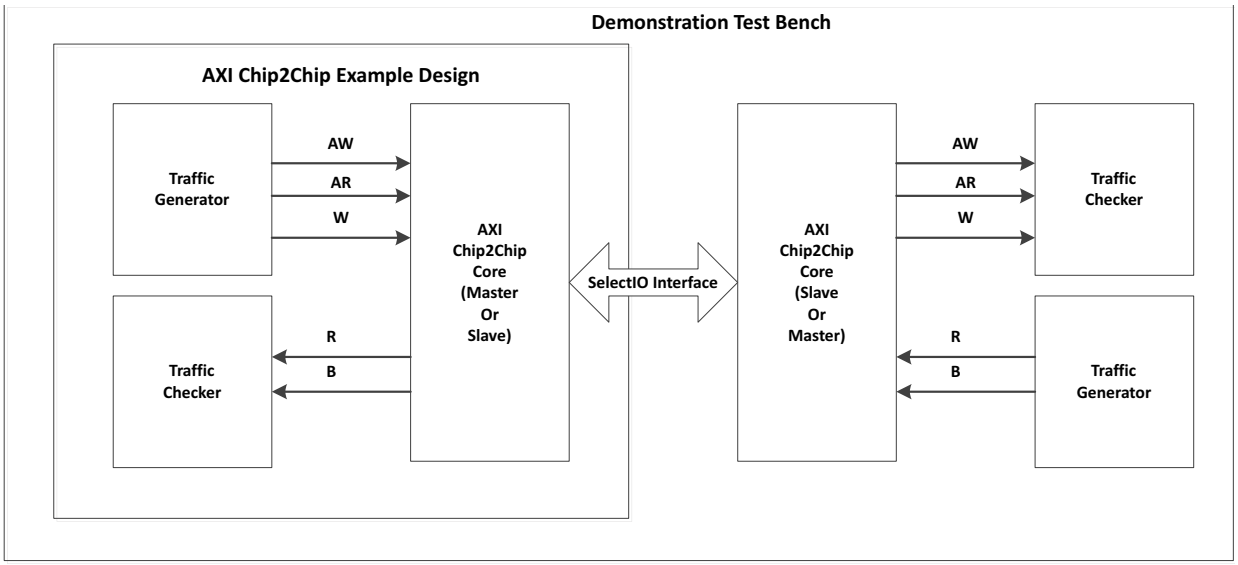

Figure 6-1: **Demonstration Test Bench Block Diagram**

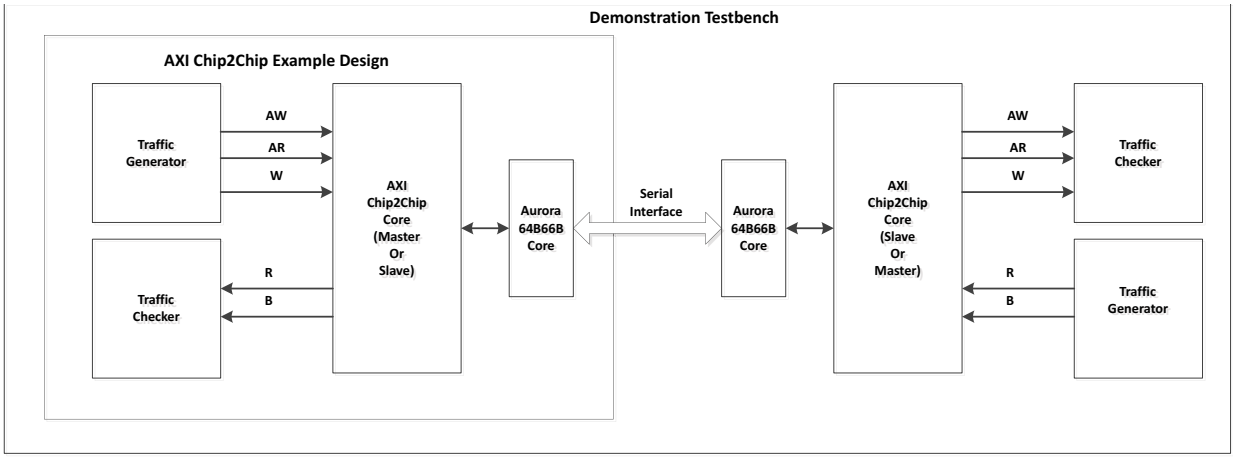

Figure 6-2: **Demonstration Test Bench with Aurora Interface**

To demonstrate the AXI Chip2Chip core, an instance of AXI Chip2Chip core in complementary mode is connected to the AXI Chip2Chip core in the example design.

The demonstration test bench performs the following tasks:

- Generates input clock signals.
- Applies a reset to the example design.
- Waits for one of the interrupt signals (Link Status and Multi-Bit Error) to be asserted. If Link status is asserted, a stable link is established between the Master and Slave AXI Chip2Chip cores. If Configuration Error or Multi-Bit Error is asserted, the test bench fails with `Error: Link Not Detected`.
- If a link is successfully established, `Link detected` is displayed in the console.
- The traffic generator starts generating fixed traffic patterns at the inputs of the AXI Chip2Chip cores.
- The traffic checker checks the output signals of the AXI Chip2Chip cores against expected patterns. If the received data has an error, then error messages are issued at the console with the name, expected value and actual value of the signal in error condition.
- The transactions are shown for a time interval of 10,000 ns and the test bench finishes with the `Test Completed Successfully` in the console.

Appendix A

Verification, Compliance, and Interoperability

This appendix provides details about how this IP core was tested for compliance.

Simulation

AXI Chip2Chip cores have been tested with Xilinx® Vivado® Design Suite and the Mentor Graphics Questa SIM simulator. For the supported versions of these tools, see the Xilinx Design Tools: Release Notes Guide.

For more details about simulating your design, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 3].

The IP is tested using Xilinx proprietary standard AXI Memory Mapped OVM Verification Components (OVCs).

Hardware Testing

Figure A-1 shows the hardware testing setup for the AXI Chip2Chip core.

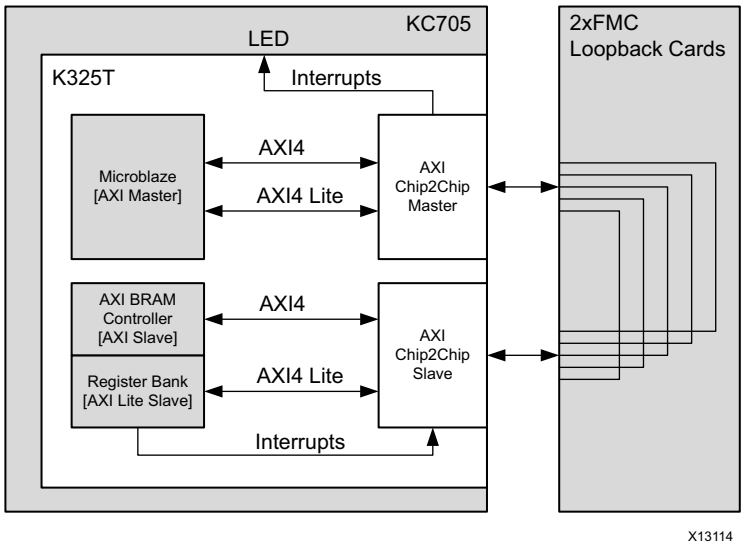

Figure A-1: **AXI Chip2Chip Hardware Testing Setup**

The AXI Chip2Chip core with a SelectIO™ FPGA interface has been hardware validated on a KC705 board using a Kintex®–7 FPGA with –1 speed grade (325T). The setup uses two additional FMC loopback cards. Table A-1 provides configuration details for the AXI Chip2Chip core and the frequency achieved by utilizing this setup with the SelectIO™ interface.

Table A-1: **Hardware Testing Configuration with a SelectIO FPGA interface**

Features			I/Os Utilized	PHY Clock (MHz)
AXI Data Width	**Chip2Chip PHY Type**	**Chip2Chip PHY Width**	**Single Ended [HR I/O Banks]**	**LVCMOS_25 I/O [Unterminated]**
32-bit	SelectIO™ SDR	Compact 4:1	38	200
	SelectIO™ DDR	Compact 1:1	58	100
	SelectIO™ DDR	Compact 2:1	32	150
	SelectIO™ DDR	Compact 4:1	20	150
64-bit	SelectIO™ DDR	Compact 2:1	46	100
	SelectIO™ DDR	Compact 4:1	28	150

Notes:

1. The AXI (system) clock frequency was set to 100 MHz, and the Common Clock mode of operation was selected for configurations having the same PHY clock and AXI clock frequencies (100 MHz).

In addition, XAPP1160 provides a setup demonstrating real-time video traffic across Kintex®-7 FPGA boards (KC705) and Zynq®-7000 devices [Ref 1]. This setup uses the AXI Chip2Chip core for connectivity across the FPGA using LPC/HPC connector cables.

The *AXI Chip2Chip Aurora Reference Design for Real-Time Video Applications* (XAPP1216) demonstrates real-time video traffic between two Kintex®–7 FPGA KC705 evaluation boards or one KC705 board and one Zynq®-7000 ZC706 evaluation board [Ref 10]. The AXI

Chip2Chip core provides connectivity between the two boards using SMA data connector cables.

Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado® Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information about migrating to the Vivado® Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 6].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado® Design Suite.

Parameter Changes

There were no parameter changes for this release.

Port Changes

For designs using SelectIO™, there were no port changes.

For designs using Aurora as the PHY type, the following ports were added:

- aurora_do_cc
- aurora_pma_init_in
- aurora_pma_init_out
- aurora_init_clk
- aurora_mmcm_not_locked

- aurora_reset_pb

Note: The support for inter operability between different IP versions is not maintained.

Debugging

This appendix provides information for using the resources available on the Xilinx® Support website, debug tools, and other step-by-step processes for debugging designs that use the AXI Chip2Chip core.

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI Chip2Chip core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the AXI Chip2Chip core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that you have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the AXI Chip2Chip

Answer Record: 54806

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

Xilinx provides premier technical support for customers encountering issues that require additional assistance. To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

There are many tools available to address AXI Chip2Chip core design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used to interact with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 7].

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB or connector cable issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `locked` port.

Core-Specific Checks

The following checks can further the debugging process:

- Check that the `axi_c2c_multi_bit_error_out` signals of both the Master and Slave cores are not asserted. The `axi_c2c_link_status_out` signal should be asserted High after the cores are calibrated.
- If the Slave is reset during normal operation (`axi_c2c_link_error_out`), reset the entire Master-Slave system.
- After downloading the software in any of the boards, reset the entire system. If there is no reset propagation, the Slave needs to be reset first, followed by the Master.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output `s_axi_arready` asserts when the read address is valid, and output `s_axi_rvalid` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The interface is enabled, and `s_axi_lite_aclk`/`m_axi_lite_aclk` are stable when the core is brought out of reset.

- The interface is not being held in reset, and the link status output of the core is asserted.
- The other interface inputs and outputs are connected and toggling.
- The main core clocks are toggling and the link error or multi-bit error interrupt outputs of the core are not asserted.
- If the simulation has been run, verify in simulation that the waveform is correct for accessing the AXI4-Lite interface.

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 provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (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 the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

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

References

This section provides supplemental material useful with this product guide:

1. *AXI Chip2Chip Reference Design for Real-Time Video Application* (XAPP1160)
2. *Xilinx Vivado AXI Reference Guide* (UG1037)
3. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
4. *Vivado Design Suite User Guide: Implementation* (UG904)
5. *Vivado Design Suite User Guide: Designing with IP* (UG896)
6. *ISE to Vivado Design Suite Migration Methodology Guide* (UG911)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
8. *Vivado Design Suite User Guide: Getting Started* (UG910)
9. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
10. *AXI Chip2Chip Aurora Reference Design for Real-Time Video Applications* (XAPP1216)
11. AMBA® AXI4 specification

    http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0022d/index.html
12. *LogiCORE IP Aurora 8B/10B Product Guide* (PG046)
13. *LogiCORE IP Aurora 64B/66B Product Guide* (PG074)
14. *LogiCORE IP High Speed SelectIO Wizard Product Guide* (PG188)
15. *LogiCORE IP SelectIO Interface Wizard Product Guide* (PG070)
16. *7 Series FPGAs SelectIO Resources User Guide* (UG471)
17. *7 Series FPGAs PCB Design Guide User Guide* (UG483)
18. *UltraScale Architecture SelectIO Resources User Guide* (UG571)
19. *UltraScale Architecture PCB Design User Guide* (UG583)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/24/2020	5.0	Added the Simulation Speed Up section in the Design Flow Steps chapter.
04/04/2018	5.0	Added description for C_SIMULATION.

Date	Version	Revision
10/04/2017	5.0	• Added a new feature for handling pending AXI Transactions in-case of Link Failure. • Removed the Disable De-Skew and Disable Clock Shift option from the GUI. • Updated the core to 5.0, as the IP is modified to calculate the correct lanes/ IO count for certain combinations of Data Width, ID Width and Address Width.
04/05/2017	4.3	Added separate parameters to set the ID widths in the slave configuration.
11/18/2015	4.2	Added support for UltraScale+ architecture-based devices.
09/30/2015	4.2	Enabled support for UltraScale™ Architecture based devices that are using Aurora physical layer interface.
04/01/2015	4.2	• Enhanced support for 128 data width (in aurora interface mode) and 64 address width. • Added User Parameters Table.
11/19/2014	4.2	• Updated "Throughput and Latency" in the Product Specification chapter. • Added more details to "Auto-Negotiation" in the Designing with the Core chapter.
10/01/2014	4.2	• Added support for Aurora 8B/10B IP core. • Clarified maximum ID width restrictions as it relates to Zynq®-7000 devices.
04/02/2014	4.2	• Updated core to v4.2. • Added details about integrating the core with the Xilinx® Aurora IP core using the Vivado® IP integrator. Included new ports details in "Migrating and Upgrading" appendix. • Added clocking, reset and interface connectivity details for connecting to the Aurora IP core.
12/18/2013	4.1	Added support for Aurora interface.
10/02/2013	4.1	• Updated core to v4.1. • Added Example Design and Test Bench chapters. • Added the Migrating and Updating appendix. • Added support for IP integrator. • Changed all signals and ports to lowercase.
03/30/2013	4.0	• Updated core to v4.0. • Added support for Vivado® Design Suite. • Removed support for ISE® Embedded Development Kit (EDK).
12/18/2012	3.0	• Updated core to v3.00a and ISE Embedded Development Kit (EDK) to v14.4. • Added support for AXI4-Lite. • Added Appendix C, Debugging.
10/25/2012	2.1	Corrected typo in Figure 1-1.
10/16/2012	2.0	Xilinx® initial release. Updated core to v2.00a and ISE Embedded Development Kit (EDK) to v14.3.
07/25/2012	1.0	Xilinx® Beta release.

Please Read: Important Legal Notices