LTPI v3.0 IP Product Guide

PG448 (v3.0) November 20, 2025

Table of Contents

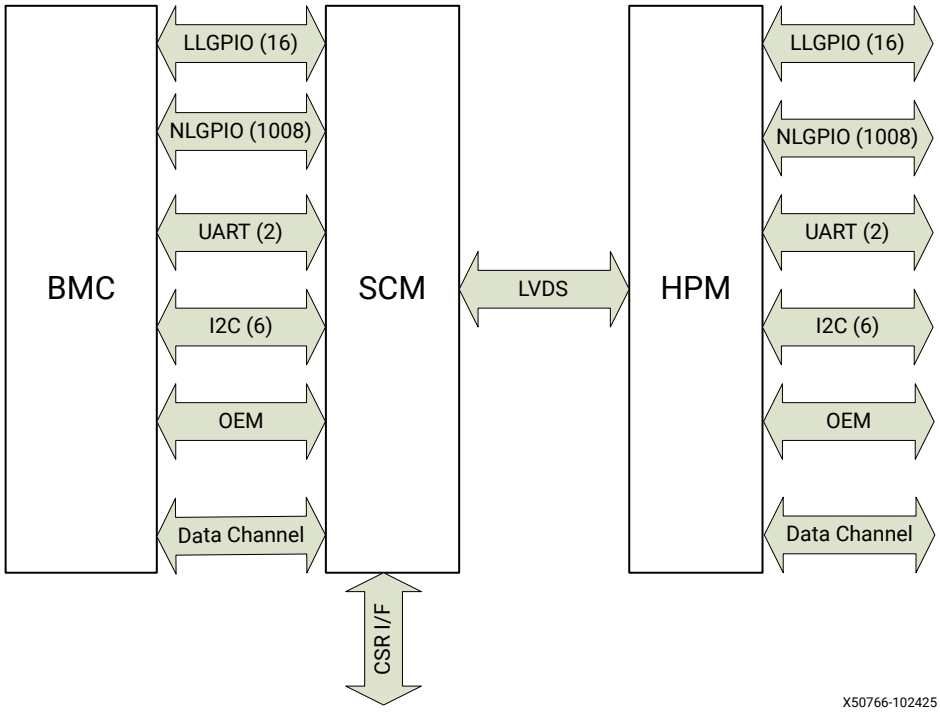
Chapter 1

Introduction

The LVDS Tunneling Protocol & Interface (LTPI) IP v3.0 was developed to work in accordance with *LTPI Specification r 1.2, v 1.0RC3* for use with the *Datacenter – Secure Control Module (DC-SCM) r 2.2, v 1.0RC2* systems. In a data center server context, the primary use case for LTPI is for the transport of these low-speed I/O channels between the Secure Control Module (SCM) which is responsible for card management and the Host Processor Module (HPM). LTPI enables the pins used by these low-speed channels on the DC-SCI connector to be reallocated for other purposes. This in turn enable new interfaces/features to be supported between the SCM and HPM.

The LTPI interface can be implemented with a pair of FPGAs for configurations with typical server Board Management Controller (BMC) devices:

Figure 1: **LTPI Signal Topology**



- **Host Processor Module (HPM) FPGA:** Provides a bridging of local HPM interfaces to LTPI.
- **Secure Controller Module (SCM) FPGA:** Provides a bridging of local SCM interfaces to LTPI.

Features

- Compliant with LTPI Specification r 1.2, v 1.0RC3
- AMD Artix™ 7, AMD Kintex™ 7, AMD Artix™ UltraScale+™, AMD Kintex™ UltraScale+™, AMD Spartan™ 7 and AMD Spartan™ UltraScale+™ families
- All standard LTPI rates starting from 25 Mb/s (25 MHz SDR):
  - Up to 800 Mb/s (400 MHz DDR) for AMD 7 series devices
  - Up to 1.2 Gb/s (600 MHz DDR) for AMD UltraScale+™ series devices
- General purpose I/O (GPIOs):
  - 16 LL GPIOs
  - Up to 1008 NL GPIOs
- Up to two universal asynchronous receiver-transmitter (UART) channels
- Up to six I2C/SMBus channels:
  - Each I2C/SMBus interface can be configured as either a controller or a target.
- Optional OEM channel:
  - Up to 32 bits of OEM data
- Optional data channel
  - When enabled, it is interfaced through the AXI4-Lite slave on SCM and AXI4-Lite master on HPM.

IP Facts

AMD LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family	AMD Artix™ 7, AMD Artix™ UltraScale+™, AMD Kintex™ 7, AMD Kintex™ UltraScale+™, AMD Spartan™ 7 and AMD Spartan™ UltraScale+™ FPGAs
Supported User Interfaces	AXI4-Lite Interface, GPIO, UART, SMBus/I2C, OEM, Data Channel
Provided with Core	
Design Files	Encrypted Verilog RTL
Example Design	System Verilog
Test Bench	System Verilog
Constraints File	XDC
Simulation Model	Verilog Source Code
Supported S/W Driver	N/A
Tested Design Flows[1]	
Design Entry	Standalone
Simulation	For supported simulators, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973).
Synthesis	AMD Vivado™ Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: N/A
Support web page	

Notes:

1. For the supported versions of third-party tools, see the *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973).

Chapter 2

Overview

Navigating Content by Design Process

AMD Adaptive Computing documentation is organized around a set of standard design processes to help you find relevant content for your current development task. See the Vivado Design Flow Overviews section in *Vivado Design Suite User Guide: Design Flows Overview* (UG892) to better understand the design flows and find content that is specific to your intended design needs. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, functional simulation, and evaluating the AMD 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
  - Chapter 5: Example Design

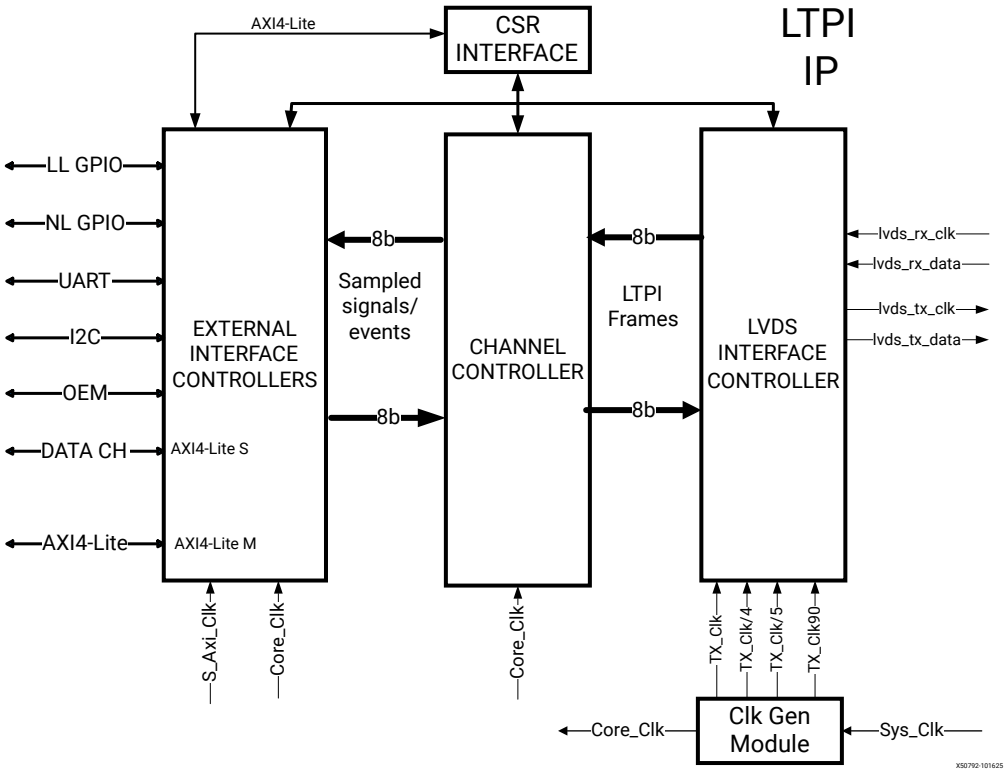
Core Overview

The LTPI IP is designed to enable the mapping and transport of various low speed I/O interfaces (known as channels) such as a GPIO, UART, I2C/SMBus, OEM and memory mapped data channel (that is, AXI4-Lite) via a high-speed Low Voltage Differential Signaling (LVDS) serial link. LTPI makes PCB routing simple and efficient using a two-pin LVDS bus. The LVDS interface is designed for low cost and low power FPGAs, making it ideal for resource constrained designs. As an industry standard protocol, LTPI ensures broad compatibility and long-term support across vendors. Additionally, LTPI's configurable and scalable architecture allows designers to customize the interface to meet varying performance and integration needs.

Product Specification

The functional block diagram of the core is shown in the following figure.

Figure 2: **LTPI Architecture**



Sub-blocks description:

- **CSR:** The Configuration and Status registers provide a way for the BMC or other device on the SCM or HPM to control the operation of the LTPI logic. Access to this block is always available in SCM mode but requires DEBUG_MODE active to be present on HPM mode.
- **External Interface Controllers:** Responsible for capturing and reconstructing of physical interfaces on LTPI channels:
  - GPIO
  - UART
  - I2C/SMBus

  - OEM
  - Data Channel
- **Channel Controller:** Responsible for:
  - Link State Machine Control (Link Training and Interface Configuration)
  - Generation of outgoing LTPI Frames
  - Parsing of incoming LTPI Frames
  - CRC Checksum Generation and Verification
- **LVDS Interface Controller:** Responsible for:
  - Serialization and Deserialization of the data on the LVDS Link
  - Comma Symbol Chasing and Locking
  - 8B/10B Encoding and Decoding

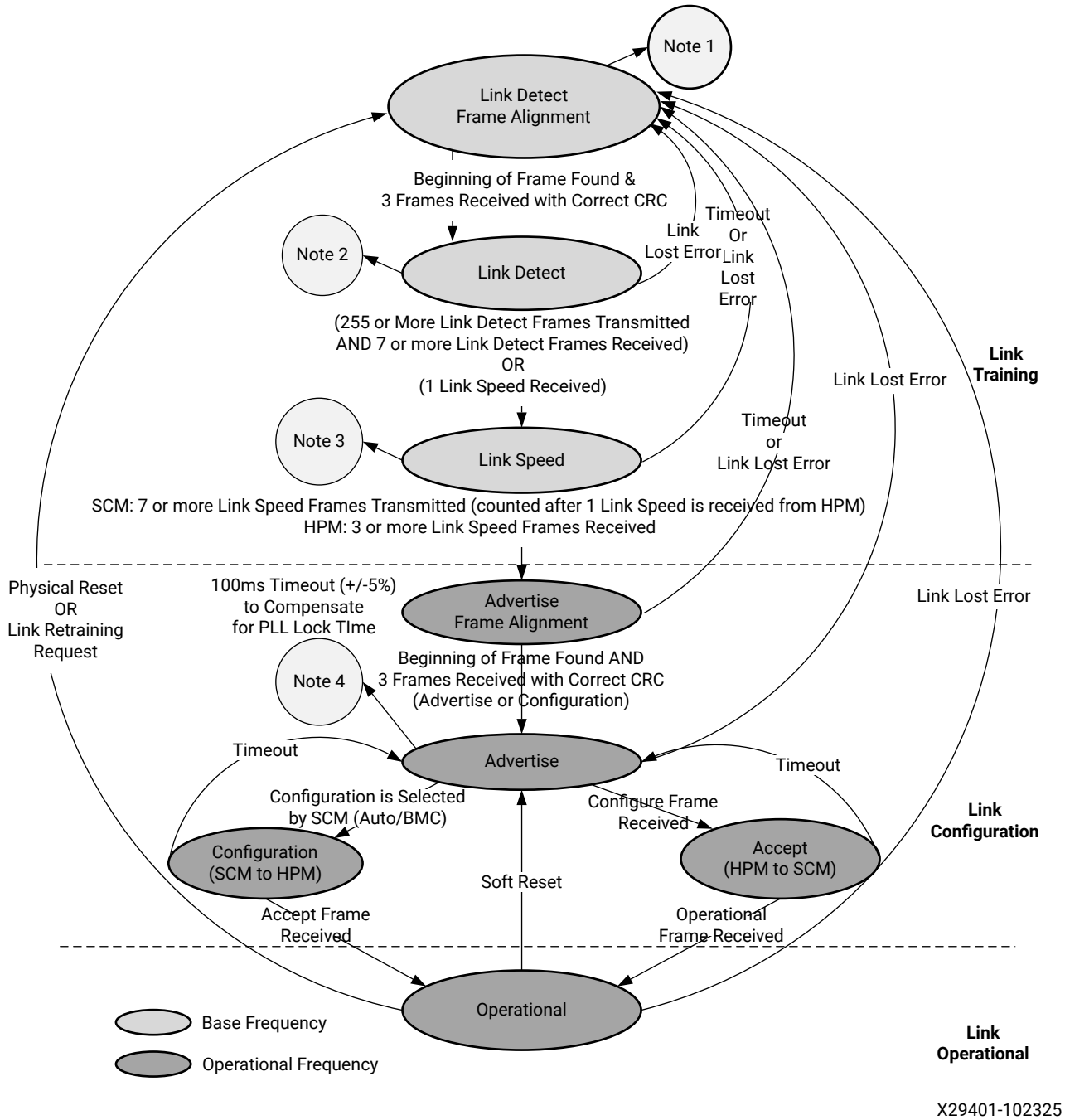
IP Functionality

Different aspects of the LTPI IP functionality are described in the following sections.

LTPI Link Initialization and Operation

The full block diagram and explanation of LTPI link initialization and operation is described in section 4 of the *LTPI Specification r 1.2, v 1.0RC3*. The following block diagram is taken from that document, with relevant notes.

Figure 3: **Link Training and Initialization Flow**



- **Note 1:** The time it takes to align to the beginning of the frame depends on time needed to achieve the DC-balance and how efficiently the comma symbol can be found (depends on the SERDES design). This stage is the source of initial misalignment between the SCM LTPI and HPM LTPI by means of number of frames sent and received. The number of frames sent should not be included in the minimum Frame Detect frames transmitted (255).

- **Note 2:** This stage is expected to be entered by SCM and HPM at different timings due to the initial misalignment in the first stage. This means that one side can complete this stage earlier and move to the Link Speed stage. If it happens, then the 'slower' part switches to the Link Speed even if the required number of TX and RX frames is not achieved.
- **Note 3:** Both sides keep sending Link Speed. SCM is required to send a minimum of seven Link Speed Frames, while HPM is required to receive three Link Speed Frames.
- **Note 4:** Because the LTPI clock is reconfigured to a higher speed, the LTPI needs to realign to the beginning of the frames again, similar to the Link Detect stage. Typically, the phase-locked loop (PLL) is going to be reconfigured and requires some time to lock. To compensate for that, 100 ms timeout is used on both sides.

Upon reset, the LTPI IP goes into Link Detect state and goes through the various states, as described in the preceding figure, to reach the Operational state. In the Operational state, the LTPI IP provides tunneling of GPIO, UART, SMBus/I2C, OEM and data channel signals according to standards described in the *LTPI Specification r 1.2, v 1.0RC3* and *Datacenter - Secure Control Module (DC-SCM) r 2.2, v 1.0RC2*.

The maximum LVDS rate is determined by the SPD_CAP parameter. The chosen LVDS rate can be lower than that, depending on the partner side capabilities. The operational LVDS rate that is chosen in the Link Speed state determines some of the maximum rates of the external interfaces, as described in the following sections.

GPIO

16 low latency (LL) and up to 1008 normal latency (NL) GPIO digital signals can be tunneled between the SCM and HPM side.

In the absence of CRC errors, the performance of LL GPIO is:

- Latency lower than five frame periods. The frame period is calculated as $160/DR$, where DR is the LVDS data rate (that is, DR = 400 MHz for 200 MHZ DDR LVDS, hence the frame period is 400 ns).
- The maximum toggling rate of any GPIO signal is half that of the frame rate, that is, $DR/320$.

The performance of NL GPIO is dependent on NL_GPIO parameter and in the absence of CRC errors and data channel frames:

- Latency is lower than $ceil(NL\_GPIO / 16) + 5$ frame periods. Frame period is calculated as $160/DR$, where DR is the LVDS data rate (that is, DR = 400 MHz for 200 MHZ DDR LVDS, hence the frame period is 400 ns).
- The maximum toggling rate of any GPIO signal is $1/(2 \times ceil(NL\_GPIO / 16))$ of the frame rate, that is, $DR / (320 \times ceil(NL\_GPIO / 16))$.

UART

Up to two UART channels are supported. As per the *LTPI Specification r 1.2, v 1.0RC3*, the UART TXD signals are sampled three times per frame at the `uart_rxd` input, while the UART CTS signals are sampled once per frame at the `uart_cts` input. The reproduced UART TXD signal is generated three times per frame on the `uart_txd` output, while the RTS signal is reproduced once per frame on the `uart_rts` output.

The latency and maximum baud rate of the UART, when operating with AMD LTPI IP on both the SCM and HPM side and in absence of active Data Channel:

- The TXD/RTS signal latency is less than five frame periods.
- The maximum baud rate is the rate that is advertised for UART channels, and it depends on the LVDS rate; see the following table.

Table 1: **Maximum UART Baud Rate**

LVDS Rate [Mb/s]	Max. UART Baud Rate	Baud Rate Encoding
25	115200	0xA
50	230400	0xB
75	460800	0xC
100	576000	0xD
150 and higher	921600	0xE

SMBus/I2C

As defined in the *LTPI Specification*, the SMBus/I2C bus requires a more complex, event based, transfer than GPIO and UART. SMBus/I2C has a controller device on one LTPI side (SCM or HPM) and the target device on the other (HPM or SCM). LTPI provides a transfer of commands and data for SMBus/I2C between SCM and HPM according to the *LTPI specification r 1.2, v 1.0RC3*.

There are two possible SMBus/I2C rates:

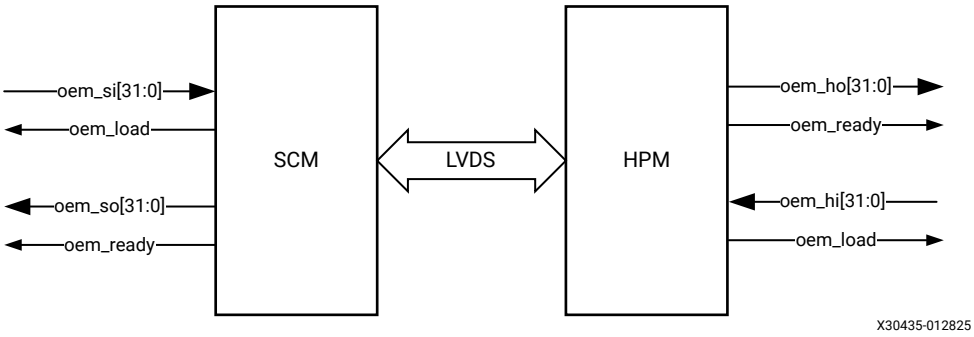
- 100 kb/s
- 400 kb/s

OEM Channel

When enabled, you can send up to 32 bits of data in every IO frame through the OEM channel. The OEM channel is bidirectional and symmetrical, so the same number of bits is transferred from the SCM to the HPM as it is in the opposite direction. The width of the OEM data is defined in the GUI.

It is not mandatory to synchronize the OEM input to `core_clk`, but if the OEM is synchronized with `core_clk`, an additional signal `oem_load` is provided on input side of the OEM. That signal has a pulse of one `core_clk` cycle and indicates the loading of the user-provided OEM data. Use of this signal is optional but it can help in determining when the best time is to update the register from which OEM data is sampled. Similarly, on the output side of OEM there is a signal `oem_ready`, which pulses for one `core_clock` cycle when the OEM extracted from the IO frame is updated.

Figure 4: **OEM Channel**

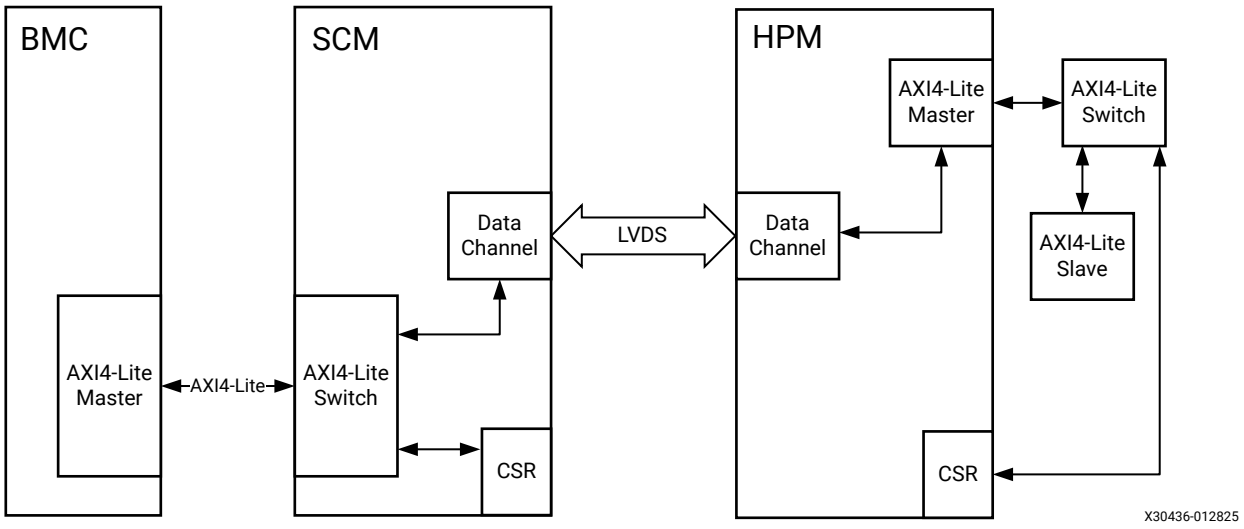


Data Channel

When Data Channel is enabled, you can access memory mapped registers located on the HPM through the AXI4-Lite slave port on the SCM. Memory mapped access requests are issued on the SCM AXI4-Lite port, packaged into an LTPI frame, transferred to the HPM via LTPI and reproduced into an AXI4-Lite memory mapped request on the AXI4-Lite master port on the HPM. Upon request on the AXI4-Lite master port on the HPM, the AXI4-Lite slave responds, the response is packaged into the LTPI frame, transferred via LTPI to the SCM, where it is reproduced on the AXI4-Lite slave port. Only one outstanding transaction is permitted, meaning that a request for a new transaction can be made only after the response for the previous one has been returned. See the following figure which describes the data path for a memory mapped data channel over LTPI.

***Note*:** Access to CSR on HPM is optional and requires enabling DEBUG_MODE. When DEBUG_MODE is enabled, the HPM CSR is visible through the AXI4-Lite slave port. You can then accesses the HPM CSR like any other port. See the following figure.

Figure 5: **Memory Mapped Access through Data Channel**



LTPI frame as well as memory mapped request commands are described in the *LTPI Specification r 1.2, v 1.0RC3* in tables 12 and 13.

***Note*:** As per the specification, the CRC is calculated for entire payload fields after the comma symbol of the LTPI Frame. CRC error is described as a command in Table 12, and it is supposed to be sent as a response from the HPM in case it received the data frame with the CRC error. The default data frame type is defined by the frame subtype which is the part of the payload. That means that the frame subtypes cannot be distinguished in the case of CRC errors, hence the HPM cannot determine that the CRC error happened for the data frame and therefore cannot generate the CRC error command. The same principle applies in the opposite direction, that is, the SCM cannot generate CRC error commands for the data frame.

Data channel allows abstraction of the memory mapped registers which are located on a separate board (HPM) by packaging those requests into LTPI data frames. This means that each request takes the time necessary to:

1. Place a request on the SCM
2. Form a frame
3. Send it to the HPM
4. Form a request on the far side (HPM)
5. Wait for the memory mapped register's response
6. Receive a response
7. Form a return frame
8. Send it to the SCM
9. Extract it from the frame and generate a response

This process takes time and means that each request on SCM side takes more than 10 LTPI frame periods to get a response. In the case of a slow AXI4-Lite slave on the HPM side, it could take significantly longer and in the case of an AXI4-Lite slave that is unresponsive, a timeout is necessary in order avoid stalling the system. The timeout is set so that:

- A lack of response data frame received on the SCM for 500 µs causes a timeout. When a timeout occurs on the SCM side, an error response (code 10) is returned via `bresp` or `rresp` pins.
- A lack of response on HPM AXI4-Lite master for 400 µs causes a timeout and generates a pulse on `axil_to` output, which you can use to reset the AXI4-Lite slaves. In this case a response data frame with information that a timeout has occurred on the HPM is sent to the SCM.

Port Descriptions

The core external interfaces are described in the following tables.

External Signals Interface

Table 2: **External Signals Interface**

Port Name	I/O	Clock	Description
sys_clk	I		System clock, default 25 MHz.
rst_n	I	N/A	Active-Low system reset.
ll_gpi[15:0]	I	core_clk	Low latency General Purpose inputs.
ll_gpo[15:0]	O	core_clk	Low latency General Purpose outputs.
nl_gpi[NL_GPIO-1:0]	I	core_clk	Normal latency General Purpose inputs.
nl_gpo[NL_GPIO-1:0]	O	core_clk	Normal latency General Purpose outputs.
smb_scl_in[5:0]	I	core_clk	SMBus/I2C SCL input.
smb_sda_in[5:0]	I	core_clk	SMBus/I2C SDA input.
smb_scl_oe[5:0]	O	core_clk	SMBus/I2C SCL tristate output enable.
smb_sda_oe[5:0]	O	core_clk	SMBus/I2C SDA tristate output enable.
uart_rxd[1:0]	I	core_clk	UART RXD input.
uart_cts[1:0]	I	core_clk	UART CTS input.
uart_txd[1:0]	O	core_clk	UART TXD output.
uart_rts[1:0]	O	core_clk	UART RTS output.
aligned	O	core_clk	Signal goes high once the Link Operational state is reached and stays high until link loss.
intrpt	O	core_clk	Interrupt signal is used only when IP_MODE = SCM and it is a level signal where all enabled interrupts are ORed. See interrupt status and enabled registers in Table 6.

Table 2: **External Signals Interface** *(cont'd)*

Port Name	I/O	Clock	Description
diag_led[7:0]	O	core_clk	Diagnostics LEDs.
i2c_timeout_err	O	core_clk	SMBus/I2C timeout error indication.
oem_din[OEM_DW-1:0]	I	core_clk	OEM data input. Exists only if OEM channel is enabled.
oem_load	O	core_clk	Indication when OEM input data is loaded. Exists only if OEM channel is enabled.
oem_dout[OEM_DW-1:0]	O	core_clk	OEM data output. Exists only if OEM channel is enabled.
oem_ready	O	core_clk	Indication when OEM output data is available. Exists only if OEM channel is enabled.
core_clk_out	O		Core clock which runs most of the logic and sampling of all the IO signals that are being tunnelled over the LTPI.

LVDS Signals Interface

Table 3: **LVDS Signals Interface**

Port Name	I/O	Clock	Description
lvds_rx_clk	I		LVDS RX clock.
lvds_rx_data	I	lvds_rx_clk	LVDS RX data.
lvds_tx_clk	O		LVDS TX clock.
lvds_tx_data	O	lvds_tx_clk	LVDS TX data. This data signal is internally generated at the clock edge of a clock that is 90 degrees shifted to ltpi_tx_clk, but ltpi_tx_clk should be used to sample ltpi_tx_data at the far end.

AXI4-Lite Signal Interface

Table 4: **AXI4-Lite Signal Interface**

Port Name	I/O	Clock	Description
s_axi_aclk[1]	I		SCM AXI4-Lite clock, default 100 MHz.
s_axi_aresetn[1]	I	s_axi_aclk	Active-Low AXI4-Lite reset.
s_axi_awaddr[31:0][1]	I	s_axi_aclk	AXI4-Lite write address. Width of this bus is 32 b when Data Channel is enabled compared to 14 b not enabled.
s_axi_awvalid[1]	I	s_axi_aclk	AXI4-Lite write address valid.
s_axi_awready[1]	O	s_axi_aclk	AXI4-Lite write address ready.
s_axi_wdata[31:0][1]	I	s_axi_aclk	AXI4-Lite write data.
s_axi_wvalid[1]	I	s_axi_aclk	AXI4-Lite write valid.
s_axi_wready[1]	O	s_axi_aclk	AXI4-Lite write ready.
s_axi_bresp[1:0][1]	O	s_axi_aclk	AXI4-Lite write response.

Table 4: **AXI4-Lite Signal Interface** *(cont'd)*

Port Name	I/O	Clock	Description
s_axi_bvalid[1]	O	s_axi_aclk	AXI4-Lite write response valid.
s_axi_bready[1]	I	s_axi_aclk	AXI4-Lite write response ready.
s_axi_araddr[31:0][1]	I	s_axi_aclk	AXI4-Lite read address. Width of this bus is 32 b when Data Channel is enabled compared to 14 b when not enabled.
s_axi_arvalid[1]	I	s_axi_aclk	AXI4-Lite read address valid.
s_axi_arready[1]	O	s_axi_aclk	AXI4-Lite read address ready.
s_axi_rdata[31:0][1]	O	s_axi_aclk	AXI4-Lite read data.
s_axi_rresp[1:0][1]	O	s_axi_aclk	AXI4-Lite read response.
s_axi_rvalid[1]	O	s_axi_aclk	AXI4-Lite read valid.
s_axi_rready[1]	I	s_axi_aclk	AXI4-Lite read ready.
m_axi_aclk[2]	I		HPM AXI4-Lite clock, default 100 MHz.
m_axi_aresetn[2]	I	m_axi_aclk	Active-Low AXI4-Lite reset.
m_axi_awaddr[31:0][2]	O	m_axi_aclk	AXI4-Lite write address.
m_axi_awvalid[2]	O	m_axi_aclk	AXI4-Lite write address valid.
m_axi_awready[2]	I	m_axi_aclk	AXI4-Lite write address ready.
m_axi_wdata[31:0][2]	O	m_axi_aclk	AXI4-Lite write data.
m_axi_wvalid[2]	O	m_axi_aclk	AXI4-Lite write valid.
m_axi_wready[2]	I	m_axi_aclk	AXI4-Lite write ready.
m_axi_bresp[1:0][2]	I	m_axi_aclk	AXI4-Lite write response.
m_axi_bvalid[2]	I	m_axi_aclk	AXI4-Lite write response valid.
m_axi_bready[2]	O	m_axi_aclk	AXI4-Lite write response ready.
m_axi_araddr[31:0][2]	O	m_axi_aclk	AXI4-Lite read address.
m_axi_arvalid[2]	O	m_axi_aclk	AXI4-Lite read address valid.
m_axi_arready[2]	I	m_axi_aclk	AXI4-Lite read address ready.
m_axi_rdata[31:0][2]	I	m_axi_aclk	AXI4-Lite read data.
m_axi_rresp[1:0][2]	I	m_axi_aclk	AXI4-Lite read response.
m_axi_rvalid[2]	I	m_axi_aclk	AXI4-Lite read valid.
m_axi_rready[2]	O	m_axi_aclk	AXI4-Lite read ready.
axil_to[2]	O	m_axi_aclk	AXI4-Lite master timeout. This signal (pulse) can be used to reset AXI4-Lite slaves on HPM side.

Notes:

1. Port available when IP_MODE=SCM or DEBUG_MODE = 1.
2. Port only available if Data Channel is enabled, and only on HPM.

Register Space

The LTPI IP uses the register address space as described in the *LTPI Specification r 1.2, v 1.0RC3*, section 3.2 LTPI Control and Status Registers (CSR).

Table 5: **Section of Register Map table from the LTPI Specification**

Offset	Name	Description
0x0000_0000 - 0x0000_01FF	CPLD/FPGA Information	The offset is reserved for OEM-defined CPLD/FPGA information such as CPLD/FPGA image version numbers, device identifiers, OEM specific information, and so on.
0x0000_0200 - 0x0000_02FF	LTPI Control and Status	LTPI Status and Link Training Control Registers defined in the *LTPI Specification r 1.2, v 1.0RC3*, section 3.2 LTPI Control and Status Registers (CSR).
0x0000_0300 - 0x0000_03FF	Reserved	Reserved for future use
0x0000_0400 - 0xFFFF_FFFF	Data Channel	Data Channel memory space. AXI4-Lite slaves need to have their address offset set accordingly to be accessed from SCM via data channel.

The proprietary registers for this IP are shown in the following table:

Table 6: **Proprietary Registers**

Offset (hex)	Register Name	Access	Default (hex)	Register		
				Bit Range	Field	Description
0x00	LTPI and IP versions	RO	0x01000300	31:24	Major	LTPI Specification Version number
				23:16	Minor	LTPI Specification Version number
				15:8	Major	LTPI IP Version
				7:4	Medium	LTPI IP Version
				3:0	Minor	LTPI IP Version
0x08	Scratch register	RW	0x0	31:0	Scratch	Scratch register for testing data access
0x10	Interrupt status register	RC	0x0	31:6	Reserved	Reserved
				5	ucs_err	Unknown comma symbol error interrupt
				4	ll_err	Link loss error interrupt
				3	crc_err	CRC error interrupt
				2	FRCNT_OOB	Out of bound error on received Frame Counter and no CRC error
				1	I2C_TO	I2C TO - see I2C TO registers for more details
				0	tr_conf	Trigger configuration State interrupt. Cleared by writing 1 to bit 11 of 0x280

Table 6: **Proprietary Registers** *(cont'd)*

Offset (hex)	Register Name	Access	Default (hex)	Register		
				Bit Range	Field	Description
0x14	Interrupt enable	RW	0x1	31:6	Reserved	Reserved
				5	en_ucs_err	Enable unknown comma symbol error interrupt
				4	en_ll_err	Enable link loss error interrupt
				3	en_crc_err	Enable CRC error interrupt
				2	en_FRCNT_OOB	Enable FRCNT_OOB interrupt
				1	en_I2C_TO	Enable I2C_TO interrupt
				0	Reserved	Reserved
0x80	I2C TO IF0	RC	0x0	31	TO_IND	Timeout indicator = 1 if timeout happened
				30	Cntr/Trgt	Controller/Target
				29:12	Reserved	Reserved
				11	scl_tri	Values of signal at the point of timeout
				10	sda_tri	
				9	scl_in	
				8	sda_in	
				7:4	EVT_IN	Event in state
				3:0	EVT_OUT	Event out state
0x84	I2C TO IF1	RC	0x0	31:0		I2C timeout information, same as for IF0
0x88	I2C TO IF2	RC	0x0	31:0		I2C timeout information, same as for IF0
0x8C	I2C TO IF3	RC	0x0	31:0		I2C timeout information, same as for IF0
0x90	I2C TO IF4	RC	0x0	31:0		I2C timeout information, same as for IF0
0x94	I2C TO IF5	RC	0x0	31:0		I2C timeout information, same as for IF0

Notes:

1. RO – Read Only
2. RW – Read/Write
3. RC – Clear on Read
4. Local LTPI Detect Capabilities and Advertise Capabilities are RO registers when CONF_MODE = AUTO and RW registers when CONF_MODE = BMC.

Configuration Parameters

The following table describes the user configurable parameters.

Table 7: **Configuration Parameters**

Parameter	Allowed Values	Description
PLTF_ID	16-bit value Default: 0	OEM-defined SCM/HPM platform ID.
IP_MODE	HPM/SCM Default: HPM	Specifies whether the IP is used in SCM or HPM mode.
CONF_MODE	AUTO/BMC Default: AUTO	AUTO: The LTPI IP instance decides what capabilities to use on the link with the HPM and enters the Link configuration state automatically. BMC: The interrupt is used to get software to make the decision and trigger the entrance into the Link configuration state.
SYS_CLK_FREQ	25/50/100MHz Default: 25	Frequency of the sys_clk input to the IP.
SPD_CAP	16-bit value Default: 0x803F	Link speed capability based on Table 21 of the *Datacenter – Secure Control Module (DC-SCM) r 2.2, v 1.0RC2*, representing supported LVDS rates. The default is DDR with all standard frequencies supported up to 200 MHz. The value of this parameter depends on the chosen FPGA part and some values would not be allowed; for example, the 7 series family maximum supported rate is 400 MHz DDR (800 Mb/s). **RECOMMENDED:** *It is recommended to set the link capability as high as necessary, do not set it too high or leave as default. Lower maximum speed capability requires lower logic clock frequency, saves power, and reduces timing issues.* ***Note***: Due to dynamic clock configuration limitations on 7 series, certain `SPD_CAP` combinations on SCM and HPM can prevent alignment. Avoid setting the `SPD_CAP_150` or `CPD_CAP_75` as the highest supported value.
NL_GPIO	1 – 1008 Default: 128	Number of normal latency GPI and GPO ports to be provided on the IP. This number, rounded up to the nearest number divisible by 16, is advertised by the IP during link initialization. Link initialization might result in a smaller number being decided upon. If number of GPI signals to connect the IP to is less than NL_GPIO, then the unused GPI ports are driven to a fixed Low logic level.
UART_EN	2-bit value Default: 11	UART channel enable. The default LTPI frame supports 2 UART channels. This parameter defines whether these channels are enabled, for example,“01” means that only channel 0 is enabled. This information is advertised by the IP during link initialization. Only channels that are enabled on both SCM and HPM sides shall be supported in the Operational state. The LTPI IP always advertises UART flow control as enabled. Each provided UART port shall have an RTS output and CTS input. If flow control is not required, then the CTS input should be driven to LOW level and the RTS output should be left unconnected.
SMB_EN	6-bit value Default: 111111	SMBus/I2C channel enables. The default LTPI frame supports channel 0 (bit 0) to channel 5 (bit 5). This parameter defines if each channel is enabled, but the port is always provided on the IP. This information is advertised by the IP during link initialization. After link initialization, the channel is enabled only if enabled on both SCM and HPM.

Table 7: **Configuration Parameters** *(cont'd)*

Parameter	Allowed Values	Description
SMB_SPD	6-bit value Default: 000000	Specifies whether SMBus/I2C class is 100 kb/s “0” or 400 kb/s “1” per SMBus/I2C channel. This information is advertised by the IP during link initialization. If the same channel has different class on SCM and HPM, the lower one shall be chosen in the Operational state. Setting SMBus/I2C class to 400 kb/s for expected LVDS rates below 200 Mb/s significantly reduces maximum SMBus/I2C rate, so it is not recommended.
SMB_MODE	6-bit value Default: 111111	SMBus/I2C interfaces modes – bit set to 1 for Controller, 0 for Target. After link initialization, the correct behavior of SMBus/I2C channel is achieved only if the interface modes for the channel on SCM and HPM are different.
DEBUG_MODE	0/1 Default: 0	Only relevant if IP_MODE = HPM. If set to 1, an AXI4-Lite slave interface is provided to improve the in-system debugability of the design.
DATA_CHN_EN	1-bit value Default: 0	Data channel enabled when set to 1.
OEM_DW	0-32 Default:0	OEM data width. When set to 0, OEM is disabled.
OEM_CAP	16-bit value Default: 0x0000	OEM Capabilities presented in Advertise Frame.

Resource Utilization

The following table lists the resource utilization values for typical configurations:

Table 8: **Resource Utilization for Typical Configurations**

Device	IP Mode	SPD_CAP	Other Params	LUTs	FFs
AMD Spartan™ UltraScale+™	HPM	0x83FF	Default	2075	2682
AMD Spartan™ UltraScale+™	SCM	0x83FF	Default	4100	5069
AMD Spartan™ 7	HPM	0x81FF	Default	2107	2774
AMD Spartan™ 7	SCM	0x81FF	Default	4119	4973
AMD Spartan™ UltraScale+™	HPM	0x83FF	DATA_CHN_EN=1, OEM_DW=32, DEBUG_MODE=1	4287	6106

***Note*:** Utilization figures for all supported FPGA families can vary by 1 to 2%.

Designing with the Core

This section includes guidelines and additional information to facilitate designing with the core.

Clocking

All timing analysis has been performed on a -1 speed grade. The LTPI clock domains are shown in the following table. The following figure shows the clocking topology.

Figure 6: **Clock Topology**

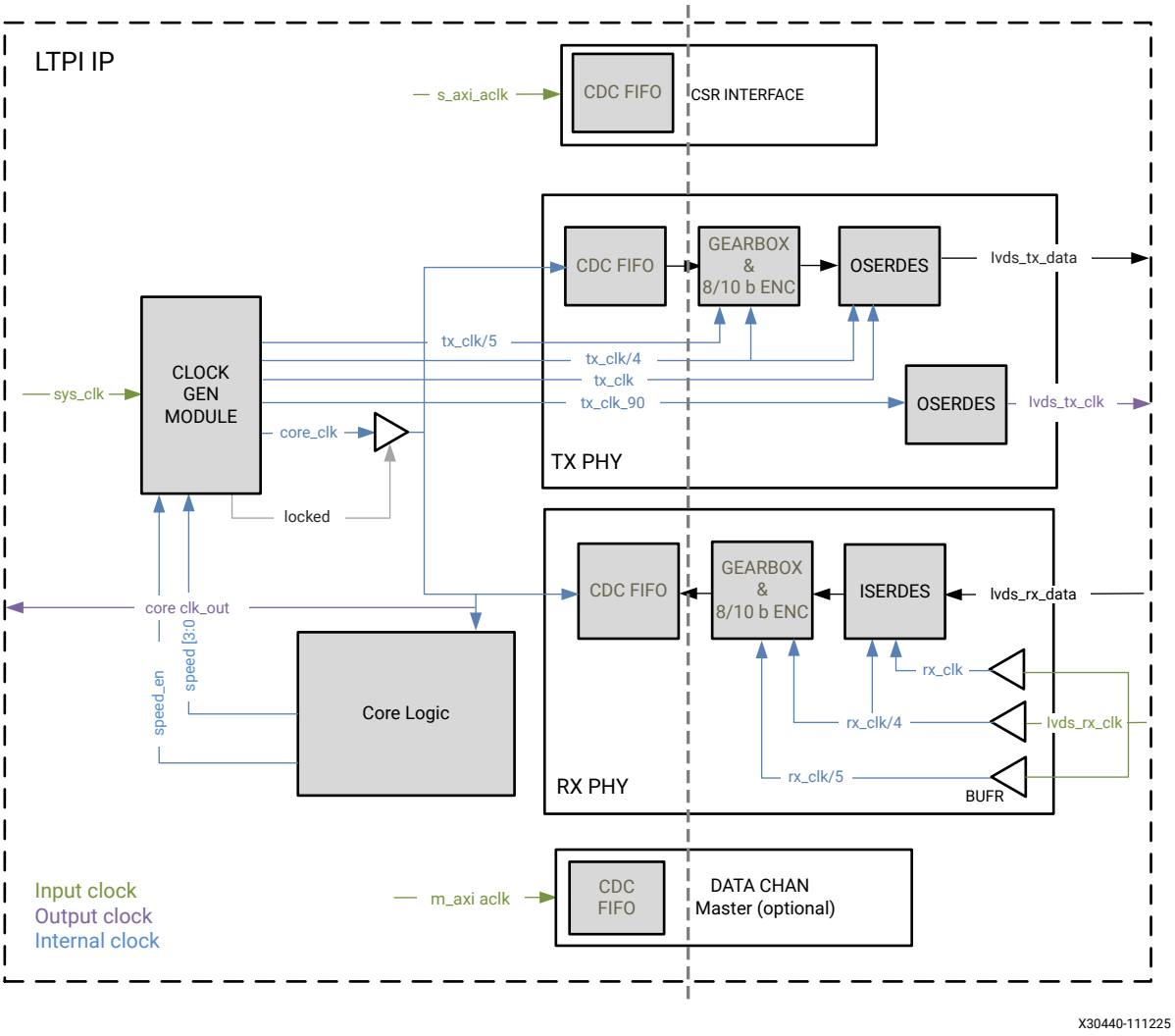


Table 9: **Clocks**

Clock	Description
sys_clk	System clock, default 25 MHz frequency.
core_clk_out	Core clock which runs most of the logic and sampling of all the IO signals that are being tunnelled over the LTPI. Its frequency depends on the expected LVDS rate and can range from 25 to 150 MHz.
s_axi_aclk	AXI4-Lite clock. Default frequency 100 MHz, but any frequency between 50 MHz and 150 MHz is supported. Optional for HPM mode. Drives all the logic in the CSR interface.
m_axi_aclk	Same characteristics as the s_axi_aclk. It can only exist on the HPM when the data channel is enabled. Drives all the logic related to AXI4-Lite Master on the HPM.
lvds_rx_clk	Supplied on the RX LVDS interface. Default frequency is 25 MHz, and all standard LTPI frequencies are supported up to 400 MHz for 7 series or 600 MHz for UltraScale+ series.
lvds_tx_clk	Generated from `sys_clk` to supply the TX LVDS interface. Default frequency is 25 MHz, and all standard LTPI frequencies are supported up to 400 MHz for 7 series or 600 MHz for UltraScale+ series. Shifted from TX data signal by 90 degrees.

You can use `core_clk_out` as a clock source for other parts of the design. The clock is active when the `aligned` output signal is high, so the `aligned` output signal can be used as a low active reset. While changing MMCM output frequencies during the LTPI negotiation, `core_clk_out` is inactive. When active, `core_clk_out` frequency is determined by the negotiated LVDS frequency.

Resets

External resets are as follows:

- System reset `rst_n`, active-Low, must have its rising edge synchronized to the `sys_clk`. Resets all the main logic and puts the LTPI IP into the Link Detect state. When activated, the reset signal must be kept Low for at least 10 `sys_clk` clock cycles.
- AXI4-Lite reset `s_axi_aresetn`, active-Low, must have its rising edge synchronized to the `s_axi_aclk`. Resets the whole CSR block, so that all registers go back to their default values. Optional in HPM mode.
- AXI4-Lite reset `m_axi_aresetn`, active-Low, must have its rising edge synchronized to the `m_axi_aclk`. It exists only on the HPM and only if the data channel is enabled. Resets the HPM's AXI4-Lite master.

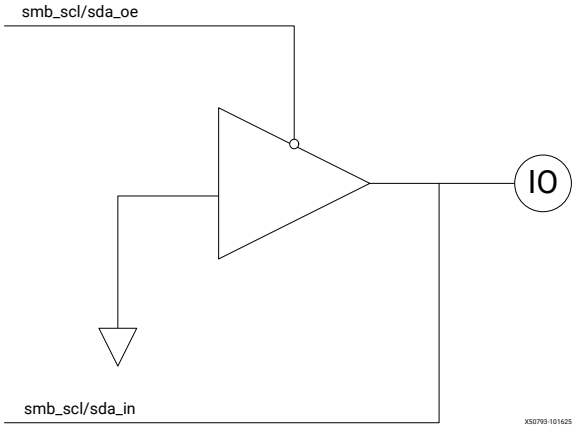
External Ports

GPIO and UART signals should be connected as GPIOs.

SMBus/I2C signals should be connected in the following way (see the following figure):

1. External ports should be open-drain tri-state I/O.
2. Inputs `smb_scl_in` and `smb_sda_in` should be connected to their corresponding I/O.
3. Outputs `smb_scl_oe` and `smb_sda_oe` should be connected to the tri-state control of the I/O, where the input of the tri-state buffer should be connected to logical zero.
4. If `smb_scl/sda_oe` = 0 then I/O = 0, else I/O = High Z

Figure 7: **SMBus/I2C I/O Connections**



LVDS Ports

For more details on LVDS connections, see the:

- *Artix 7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS181)
- *Kintex 7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS182)
- *Spartan 7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS189)
- *Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS922)
- *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930)
- *Artix UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS931)

The Example design describes the LVDS IO connection and delay compensation in detail.

GUI

You can use the GUI to input all the required parameters to generate the IP. Following are the screenshots of the GUI tabs used to setup the IP.

Figure 8: **LTPI Mode**

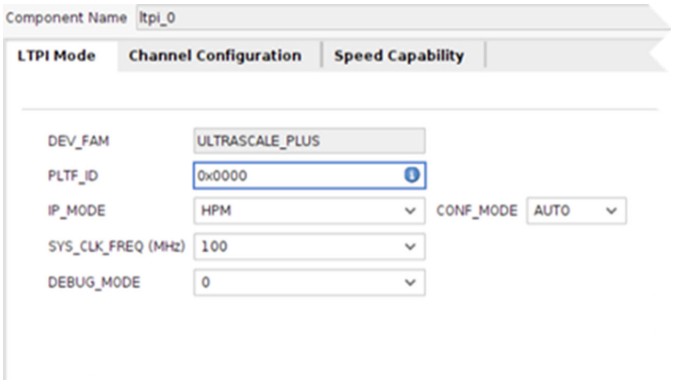


Figure 9: **Channel Configuration**

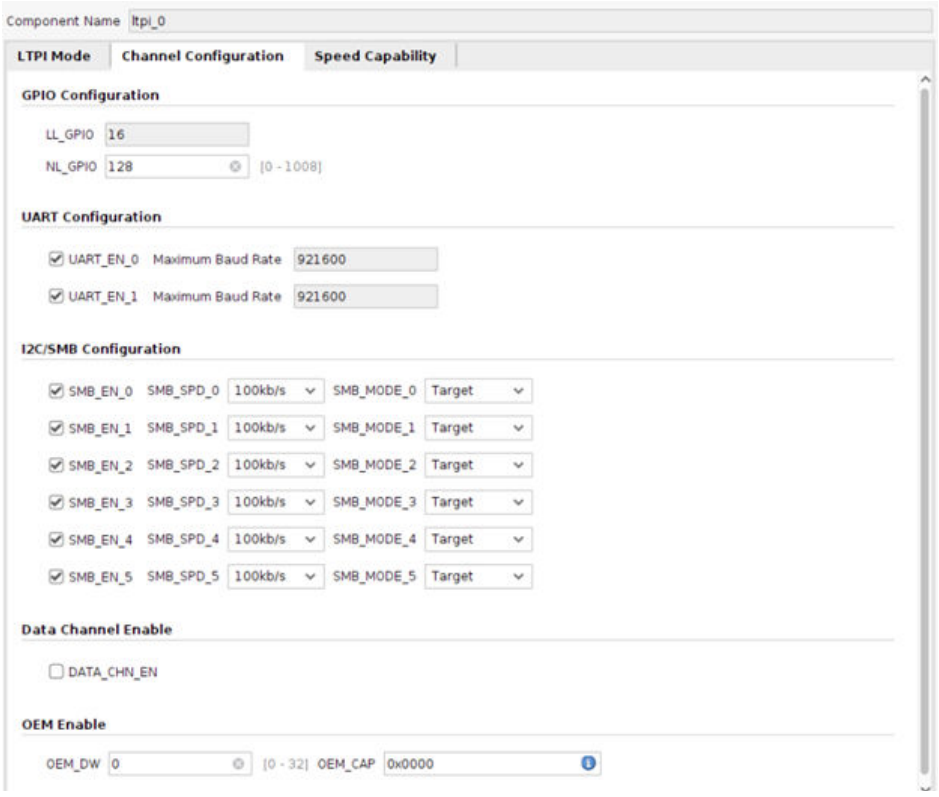
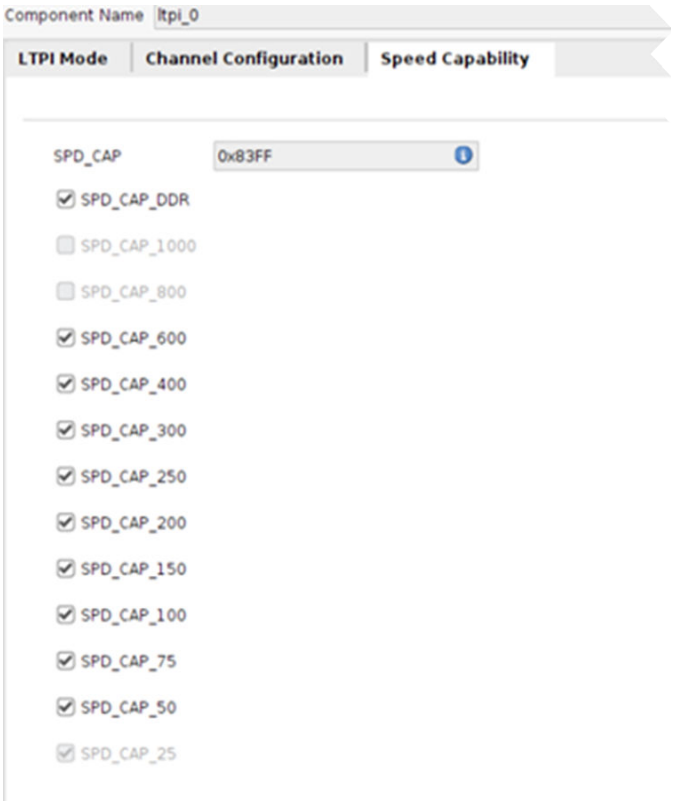


Figure 10: **Speed Capability**

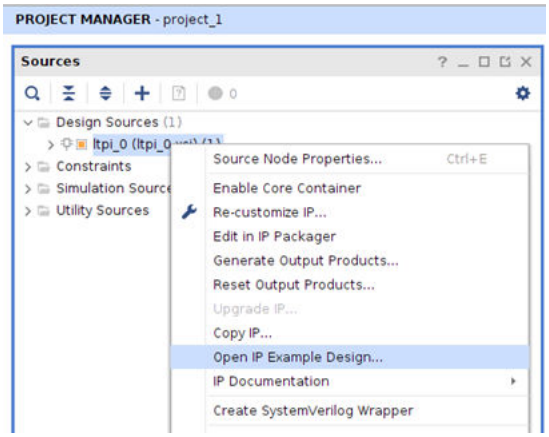


Example Design

Generating an Example Design

An example design project for the LTPI IP is available in the AMD Vivado™ IDE. Follow these steps to generate an example design project.

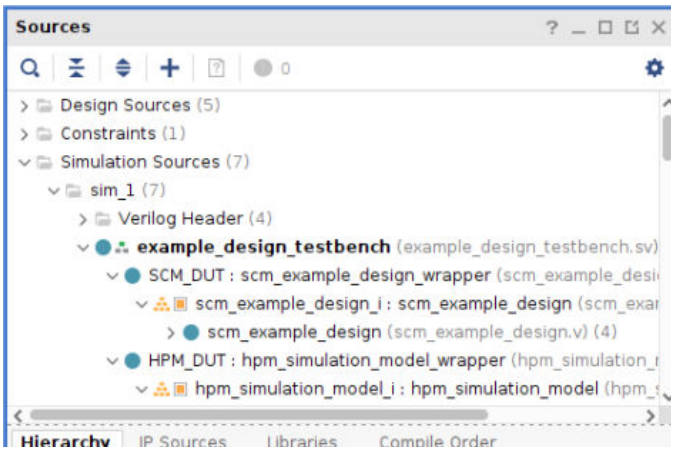
1. In Vivado IDE, add an LTPI instance to the current project using the IP catalog or by creating a new block design.
2. Double-click the instance and configure the LTPI parameters.
3. Right-click the instance and select **Open IP Example Design** from the popup menu.



4. Specify the location to place the example project directory and click **OK**.

Generates a new Vivado project that contains two block designs within an example design test bench.

Figure 11: **SCM Example Block Design**



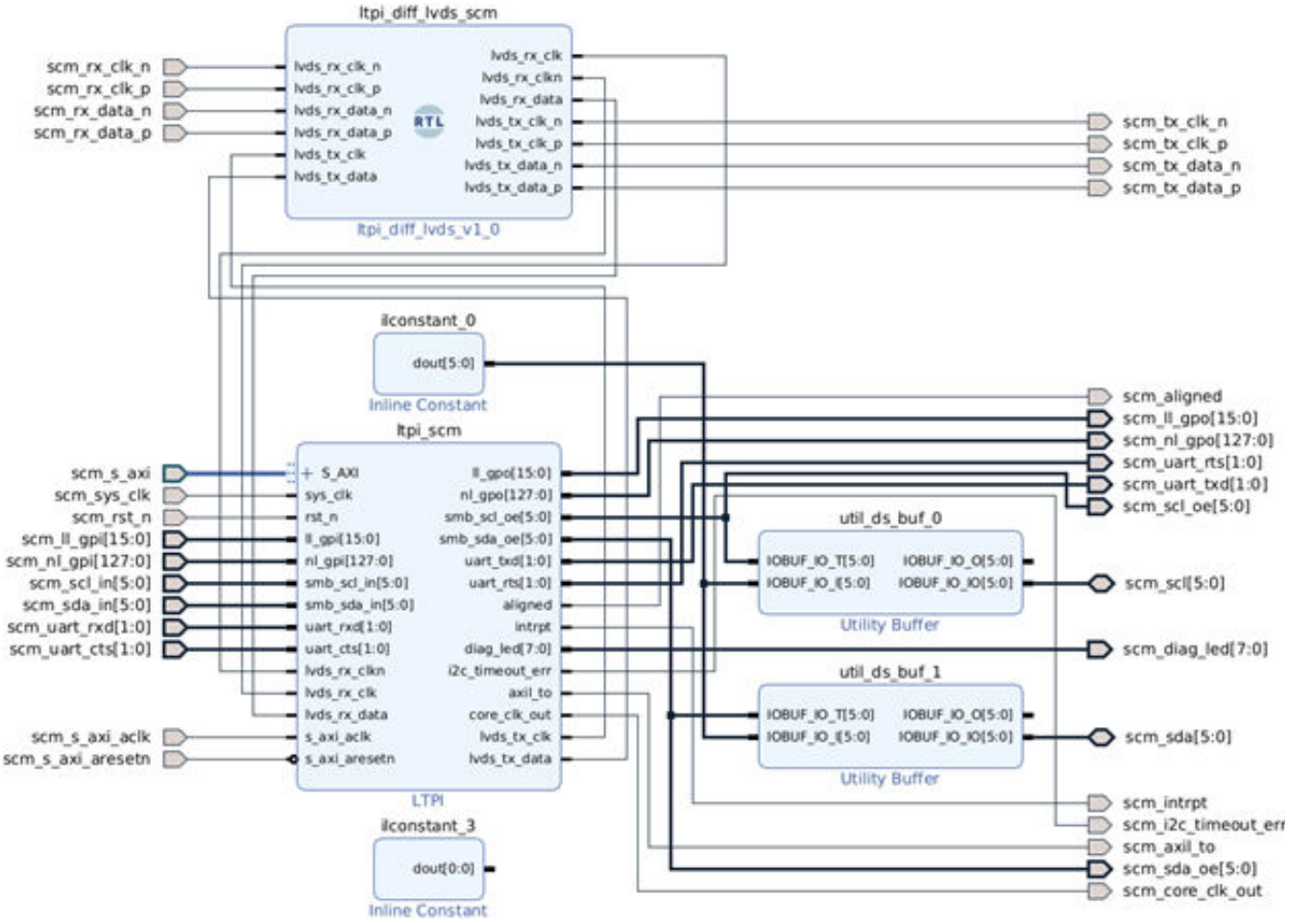
The first block design parameters are taken from the LTPI instance of the project you launched. It is synthesizable. The second BD contains a partner LTPI that pairs with the first LTPI and is the simulation model within the test bench.

For example, if the project is launched from an LTPI configured as an SCM with I2C channels set to controllers, the LTPI in the simulation model is configured as an HPM LTPI with I2C channels set as targets.

LTPI Example Design Block Diagram

Each LTPI's block design contains one instance of the LTPI, an RTL module for the LVDS logic according to the device family chosen, and some tri-state buffers for the I2C channels. Constants are used to configure some of the logic in the block design.

Figure 12: **SCM LTPI Block Diagram**

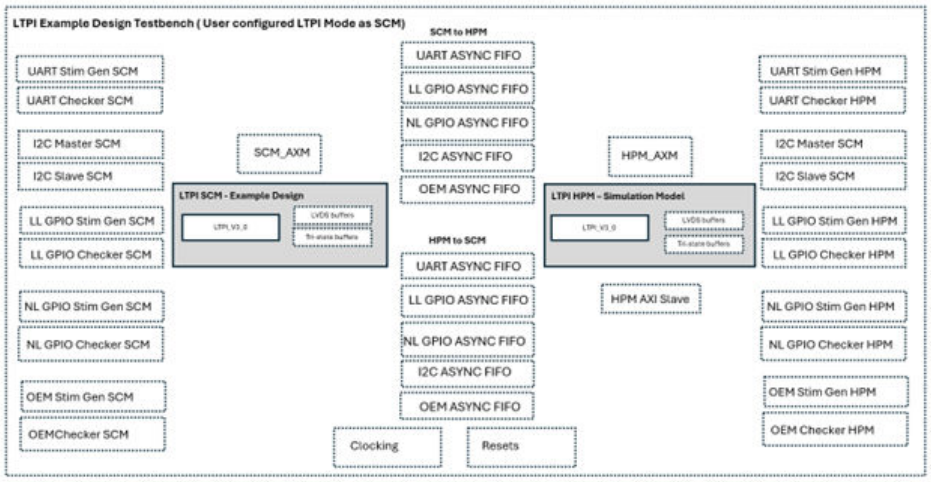


Test Bench

LTPI Example Design Test Bench

The example design project generates a top-level SystemVerilog file named `example_design_testbench.sv` that instantiates both block designs, the synthesizable example design, and the partner simulation model connecting them together.

Figure 13: **LTPI Example Design Test Bench for SCM Mode**



LTPI example design test bench connects the synthesizable example design to its partner simulation model. LTPI provides basic stimulus generators, FIFOs, and checker blocks for each of the channel interfaces on the two LTPI's interfaces. During simulation, channel traffic is tunneled in both directions between the two LTPIs.

Running a behavioral simulation causes the example design test case to initiate traffic from each channel's stimulus generator block into an LTPI. Traffic then gets tunneled over LVDS to its partner LTPI and out to the corresponding checker block for comparison with a copy of the generated traffic bypassing both LTPIs via a FIFO.

Test Bench Components

LTPI Example Design

- The previous figure shows the Secure Control Mode (SCM) implementation of the LTPI protocol.

  ***Note*:** You can also configure an example design for synthesis with an IP_MODE of HPM.

- SCM instantiates the LTPI_v3_0 core, which handles tunneling of low-speed protocols like UART, I2C, GPIO, and OEM.
- It includes LVDS buffers and tri-state buffers for physical layer signaling and direction control.

LTPI Simulation Model

- The previous figure also shows the High-Performance Mode (HPM) implementation of the LTPI protocol in simulation. HPM instantiates an LTPI_v3_0 core.

  ***Note*:** Simulation model always configures the IP_MODE value to be opposite of the example design value.

- HPM block designs are equipped with LVDS buffers and tri-state buffers (as shown in the example design) for bidirectional communication.

SCM_AXM / HPM_AXM

These are AXI transactors that connect LTPI cores to system-level buses for configuration, control, and monitoring. They allow simulation of real-world traffic and register accesses.

Stimulus and Checker Modules for Different Protocols

These modules generate input traffic (Stim Gen) and verify output correctness (Checker) for tunnelled protocols:

1. UART Stim Gen SCM / HPM & UART Checker SCM / HPM
   - Stimulus generators produce UART traffic for tunneling.
   - Checkers verify that the tunneled UART data is received correctly.
2. I2C Master SCM / HPM & I2C Slave SCM / HPM
   - I2C traffic tunnels through LTPI between master and slave test modules.
   - Ensures proper tunneling of clock stretching, addressing, and data integrity.
3. LL GPIO Stim Gen SCM / HPM & LL GPIO Checker SCM / HPM
   - Low-latency GPIO modules to test quick-response GPIO tunneling.
   - Check correctness of toggling, edge detection, and latency through LTPI.

4. NL GPIO Stim Gen SCM / HPM & NL GPIO Checker SCM / HPM
   - Normal-latency GPIO (General Purpose IO signals).
   - Validates LTPI's ability to carry less time-sensitive GPIO traffic.
5. OEM Stim Gen SCM / HPM & OEM Checker SCM / HPM
   - OEM protocol traffic is custom/user-defined signalling tunnelled through LTPI.
   - Provides flexibility to test proprietary tunnelling needs.

Asynchronous FIFOs

These bridge data transfer between SCM and HPM by handling clock domain crossing:

- SCM to HPM - UART, GPIO (LL/NL), I2C, OEM async FIFOs.
- HPM to SCM - UART, GPIO (LL/NL), I2C, OEM async FIFOs.
- Ensures correct data buffering and synchronization across SCM and HPM.

Clocking

- Supplies clocks to the SCM and HPM cores, FIFOs, and stimulus/checker modules.
- Critical for simulating realistic protocol timing.

Resets

Provides resets to bring modules into known initial states during test bench operation.

HPM AXI Slave

Provides an AXI Slave interface on the HPM side for testing data channel R/W when data channel is enabled in the example design.

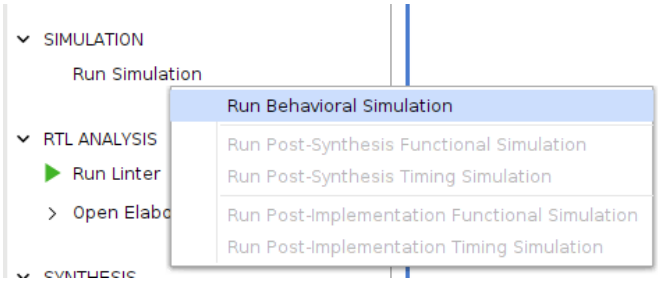
LVDS Buffers

LVDS buffers and their connections to the LTPI IP are defined in the file `ltpi_diff_lvds.v`, which can be used as an example to create the connections and design the LVDS block.

Running Behavioral Simulation

You can run behavioral simulation on your example design to check channel traffic.

1. Open your example design project in Vivado IDE.
2. On the **SIMULATION** section, click **Run Simulation** and select **Run Behavioural Simulation** from the popup menu as shown in the following figure.



A waveform window appears containing some top-level signals.

3. In the Scope window, select the test bench components.
4. In the Object window, select signals from a selected test bench component and drag them to the waveform for display.

   For example, you can select the following values:

   a. In the Scope window, select **SCM_DUT**.

   b. In the Object window, search for `scm_aligned`.

   c. Right-click **scm_aligned** and select **Add to Wave Window**. You can also drag **scm_aligned** to the **Waveform** window.

   d. Repeat the previous steps for **HPM_DUT/hpm_aligned**.
5. In Tcl Console, type `run 10ms` and press enter.

   Simulation can take a few minutes. After both SCM and HPM align signals transition from logic 0 to 1, the LTPIs tunnel traffic between them. You can see channel traffic toggling in the Waveform window.

Test Bench Files

You can find the main test bench files within the `imports` subdirectory of the main example design project directory.

The traffic generators and checkers are encrypted, but you can view and edit the following files:

- **`example_design_testbench.sv`:** Vivado generates this file based on the parameters passed from the LTPI GUI. The top level test bench instantiates the two LTPIs and all the test bench components for traffic generation and checking. Each LTPI has an example design and a partner simulation model.
- **`ltpi_diff_lvds.v` and `ltpi_i2c_tristate_buf.v`:** Vivado generates these two files based on your device family. You can copy these and instantiate them in your design to use LTPI.
- **`tb_exdes_test.svh`:**

This header file contain packages that wait for alignment of SCM and HPM LTPIs, then sends traffic over multiple channels in parallel.

Debug Logging

Test bench contains a module called the Link Interposer that placed between the two LTPIs to monitors traffic on the LVDS. Link Interposer does not inject any changes to traffic. By default, the debug messages from the Link Interposer module are turned off as these can slow simulation. Follow these steps to turn on logging of the contents of each LTPI frame.

1. In the Scope window, select **example_design_testbench**.
2. In the Object window, search for **lmd_enable**.
3. Right-click **lmd_enable**, select **Force constant** and set the value to `1`.

Now when you run simulations, Tcl Console shows frame debug information from the Link Interposer.

Synthesis of Example Design

You can synthesize block designs that contain LTPI project launched from the example design and implement it out of context. However, you cannot synthesize block designs that contain a simulation model.

Upgrading

Automated upgrade of this IP is not possible for this version. It is recommended that you regenerate and replace the IP directories when upgrading to this version.

Appendix B

Additional Resources and Legal Notices

Finding Additional Documentation

Technical Information Portal

The AMD Technical Information Portal is an online tool that provides robust search and navigation for documentation using your web browser. To access the Technical Information Portal, go to https://docs.amd.com.

Documentation Navigator

Documentation Navigator (DocNav) is an installed tool that provides access to AMD Adaptive Computing documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

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

***Note*:** For more information on DocNav, refer to the *Documentation Navigator User Guide* (UG968).

Design Hubs

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

- In DocNav, click the **Design Hubs View** tab.
- Go to the Design Hubs web page.

Support Resources

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

References

These documents provide supplemental material useful with this guide:

1. *Artix 7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS181)
2. *Kintex 7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS182)
3. *Spartan 7 FPGAs Data Sheet: DC and AC Switching Characteristics* (DS189)
4. *UltraScale Architecture and Product Data Sheet: Overview* (DS890)
5. *Kintex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics* (DS892)
6. *Kintex UltraScale+ FPGAs Data Sheet: DC and AC Switching Characteristics* (DS922)
7. *Spartan UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS930)
8. *Artix UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics* (DS931)
9. *AMD UltraScale+ FPGAs Product Selection Guide* (XMP103)
10. *Vivado Design Suite User Guide: Design Flows Overview* (UG892)
11. *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing* (UG973)
12. *LTPI Specification Revision r 1.2, v 1.0RC3* (https://drive.google.com/file/d/1XX5PTTd_rSwYCa0aRel2zOJoQ2QlhdiU/view)
13. Datacenter – Secure Control Module (DC-SCM) r 2.2, v 1.0RC2 (https://drive.google.com/file/d/1-SdSQvSWy5pNN_kBiyztblxE4jdyUe9W/view)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
11/20/2025 Version 3.0	
General updates	Added support for AMD Artix™ 7 and AMD Kintex™ 7 and AMD Artix™ and AMD Kintex™ UltraScale+™ series devices.
Chapter 5: Example Design	Updated the section.
Chapter 6: Test Bench	Added this section.

Section	Revision Summary
03/12/2025 Version 2.0	
General updates	Added support for: • Data Channel and OEM • AMD Spartan™ UltraScale+™ • Configuration through GUI
07/12/2024 Version Early Access Draft	
AMD Confidential Draft. Approved for external release under NDA only.	N/A

Please Read: Important Legal Notices

OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

Copyright