xSPI Target IP

IP Version: v1.0.0

User Guide

FPGA-IPUG-02323-1.0

June 2026

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AHB-Lite	Advanced High-Performance Bus Lite
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CPHA	Clock Phase
CPOL	Clock Polarity
CS	Chip Select
CSR	Configuration Status Register
DUT	Device Under Test
EBR	Embedded Block RAM
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GPIO	General Purpose I/O
GUI	Graphical User Interface
Hi-Z	High Impedance
IP	Intellectual Property
JTAG	Joint Test Action Group
LSB	Least Significant Bit
LUT	Look-Up Table
MISO	Controller In, Target Out
MOSI	Controller Out, Target In
MSB	Most Significant Bit
SCK	SPI Clock
SPI	Serial Peripheral Interface
RTL	Register Transfer Level
Rx	Receiver
Tx	Transmitter
xSPI	eXpanded Serial Peripheral Interface

1. Introduction

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol that facilitates data transfer between microcontrollers and peripheral devices. SPI is commonly used in embedded systems to interface with sensors, memory devices, and display controllers, among other peripherals. The eXpanded Serial Peripheral Interface (xSPI) is an extension of SPI that supports multiple SPI protocols. This document describes the Lattice xSPI Target IP.

1.1. Overview of the IP

The xSPI Target IP is an SPI interface that supports standard and quad SPI protocols. Standard SPI is the legacy four-wire interface with separate data lines for input and output. Quad SPI (QSPI) is an extension of standard SPI designed to provide a higher data transfer rate using four data lines.

1.2. Quick Facts

Table 1.1. Summary of the xSPI Target IP

IP Requirements	Supported Devices	CrossLink™-NX, Certus™-NX, CertusPro™-NX, MachXO5™-NX, MachXO3D™
	IP Changes[1]	Refer to the xSPI Target IP Release Notes (FPGA-RN-02115).
Resource Utilization	Supported User Interface	Advanced Peripheral Bus (APB), Advanced High-Performance Bus (AHB-Lite)
	Resources	Refer to Appendix A. Resource Utilization.
Design Tool Support	Lattice Implementation[2]	IP Core v1.0.0 – Lattice Radiant™ Software 2026.1 Lattice Propel™ Design Environment 2026.1
	Synthesis	Synopsys® Synplify Pro® for Lattice, Lattice Synthesis Engine
	Simulation	Refer to the Lattice Radiant Software User Guide for the list of supported simulators.
Driver Support	API Reference	Refer to the xSPI Target Driver API Reference (FPGA-TN-02426).

Notes:

1. In some instances, the IP may be updated without changes to the user guide. This user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.
2. The features and functionality described in this document are software-agnostic. However, this user guide targets primarily the Radiant design flow. For MachXO3D devices, which are indirectly supported in the Lattice Diamond™ software through Propel, software-dependent details may differ.

1.3. IP Support Summary

Table 1.2. xSPI Target IP Support Readiness

Device Family	System Clock Frequency (MHz)	Maximum SPI Clock (MHz)	Implementation Preset	Max Data Lanes	Radiant Timing Model[1]
CrossLink-NX	100	50	Standard SPI (Legacy)	x1 (Standard)	Final
	100	50	Simple Protocol	x1 (Standard)	Final
	100	50	Simple Protocol	x4 (Quad)	Final
Certus-NX	100	50	Standard SPI (Legacy)	x1 (Standard)	Final
	100	50	Simple Protocol	x1 (Standard)	Final
	100	50	Simple Protocol	x4 (Quad)	Final
CertusPro-NX	100	50	Standard SPI (Legacy)	x1 (Standard)	Final
	100	50	Simple Protocol	x1 (Standard)	Final
	100	50	Simple Protocol	x4 (Quad)	Final

Device Family	System Clock Frequency (MHz)	Maximum SPI Clock (MHz)	Implementation Preset	Max Data Lanes	Radiant Timing Model[1]
MachXO5-NX	100	50	Standard SPI (Legacy)	x1 (Standard)	Final
	100	50	Simple Protocol	x1 (Standard)	Final
	100	50	Simple Protocol	x4 (Quad)	Final
MachXO3D	100	50	Standard SPI (Legacy)	x1 (Standard)	Final
	100	50	Simple Protocol	x1 (Standard)	Final
	100	50	Simple Protocol	x4 (Quad)	Final

Note:

1. MachXO3D devices use the Diamond timing model instead of the Radiant timing model.

The example design for the xSPI Target IP allows for simulation and deployment to development boards for testing. Refer to the Example Design section for more details on how to run the example design on hardware and simulation.

1.4. Features

Key features of the xSPI Target IP include:

- Legacy SPI support
- x1 and x4 I/O widths
- Basic SPI write and SPI read
- SPI clock frequencies up to 50 MHz
- AMBA APB and AHB-Lite bus interface
- FIFO depth up to 512

1.5. Licensing and Ordering Information

The xSPI Target IP is provided at no additional cost with the Lattice Radiant software.

1.6. Minimum Device Requirements

There is no limitation on device speed grade for the xSPI Target IP. Refer to Appendix A. Resource Utilization for the minimum required resources to instantiate this IP and maximum clock frequency supported.

1.7. Naming Conventions

1.7.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.7.2. Signal Names

Signal names that end with:

- *_n* are active low signals (asserted when value is logic 0)
- *_i* are input signals
- *_o* are output signals
- *_io* are bi-directional input/output signals

1.7.3. Attribute Names

- Attribute names in this document are formatted in title case and italicized (*Attribute Name*).

2. Functional Description

2.1. IP Architecture Overview

The xSPI Target IP includes the following blocks:

- Generic SPI Target – Implements the SPI protocol target-side behavior
- Rx and Tx FIFOs – Buffer incoming and outgoing data for SPI transactions
- Memory Map – Contains configuration, control, and status registers
- CSR Interface – Provides CPU access to the memory-mapped registers
- SPI Interface (xspi* signals) – Connects the intellectual property (IP) to external SPI devices

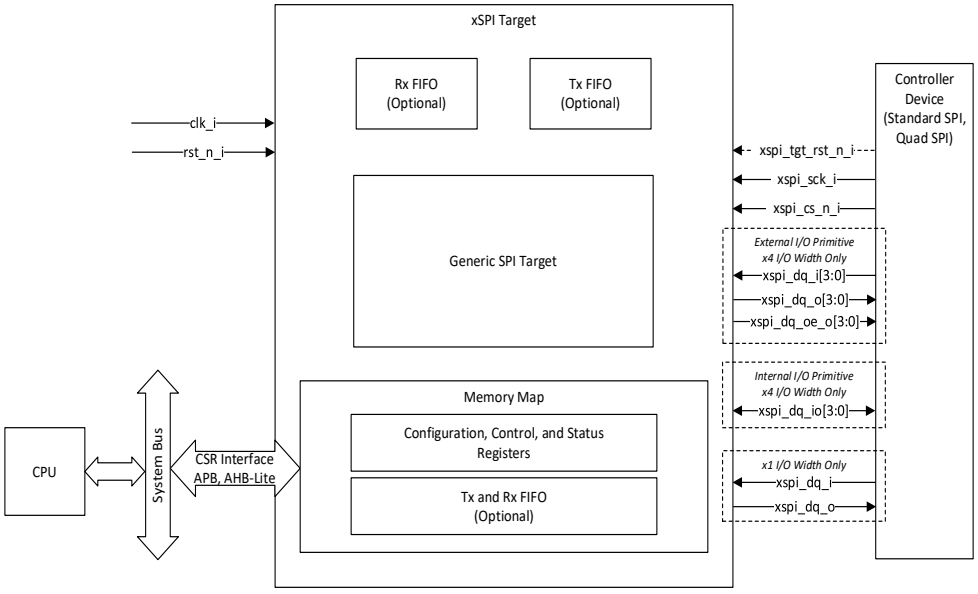

Figure 2.1. xSPI Target IP Block Diagram

2.2. Clocking

The xSPI Target IP has two clocks as illustrated in Figure 2.2:

- System Clock (clk_i) – Primary internal clock for core logic and data processing; supports frequencies up to 100 MHz.
- SPI Clock (xspi_sck_i) – Serial interface clock for SPI communication; supports frequencies up to 50 MHz.
  - For maximum SPI clock ≤ 25 MHz: System clock frequency should be greater than or equal to the maximum SPI clock frequency.
  - For maximum SPI clock > 25 MHz: System clock frequency should be at least twice the maximum SPI clock frequency.

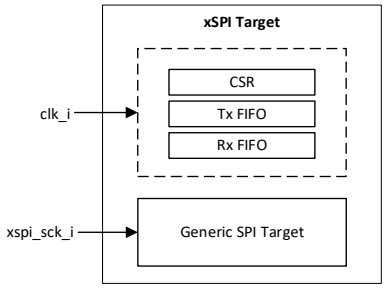

Figure 2.2. xSPI Target IP Clock Domain Block Diagram

2.3. Reset

The xSPI Target IP provides two reset ports:

- rst_n_i – Input port that serves as an asynchronous active-low reset for the entire IP including the configuration status register (CSR). The minimum reset pulse width is one clock cycle. This reset signal passes through an internal reset synchronizer. After deasserting the reset, wait for at least four clock cycles before starting any transaction.
- xspi_tgt_rst_n_i – Optional input port that provides an asynchronous active-low reset dedicated to the xSPI Target logic. This reset signal can be asserted by the xSPI Controller to independently reset the entire IP, including the CSR block, without affecting the rest of the system. Similar to rst_n_i, this signal passes through an internal synchronizer. After deasserting the reset, wait for at least four clock cycles before starting any transaction.

Besides the reset ports, the xSPI Target IP also has four programmable soft reset controls:

- IP Core Reset – Resets the xSPI Target logic except the CSR block. Resets the finite state machine (FSM), Tx FIFO, and Rx FIFO.
- IP CSR Reset – Resets the CSR block only. This resets all writeable registers to default.
- IP Tx FIFO Reset – Resets the Tx FIFO only. Clears FIFO contents and resets FIFO flags to default.
- IP Rx FIFO Reset – Resets the Rx FIFO only. Clears FIFO contents and resets FIFO flags to default.

The soft reset registers are automatically cleared after assertion. After any soft reset, wait for at least four clock cycles before starting any transaction. Refer to the Soft Reset Register section for more information.

The xSPI Target IP also optionally supports in-band reset when *Implementation Preset* is Simple Protocol. When in-band reset (FFh command followed by chip select (CS) deassertion) is received and the XSPI_TGT_CFG1. EN_SW_RESET register is set to 1, the IP core automatically resets and status register XSPI_TGT_INT_STS.RESET_DET_INT asserts. User software on the CPU shall handle processing of this interrupt and perform clean-up operations necessary for IP configuration, such as resetting the IP or flushing data, depending on the use case.

Figure 2.3 illustrates the IP reset controls and the internal blocks affected by each reset.

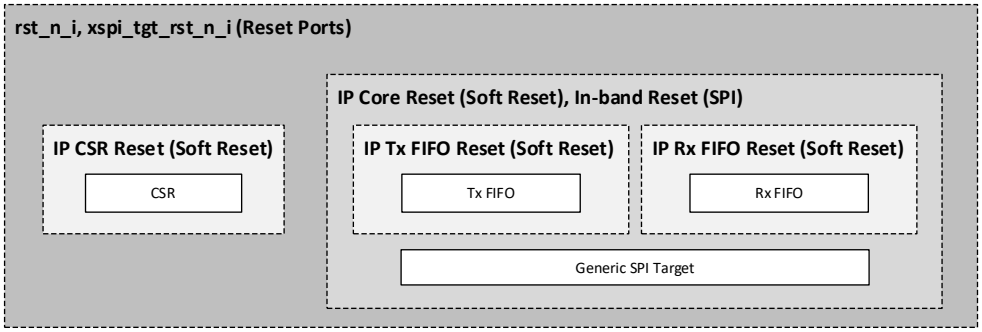

Figure 2.3. xSPI Target IP Reset Controls and Applicable Blocks

2.4. User Interfaces

Table 2.1 shows the user interface and supported protocols used to access the IP register space. The user interface of the xSPI Target IP is selectable through the *CSR Interface* attribute in the IP graphical user interface (GUI).

Table 2.1. User Interfaces and Supported Protocols

User Interface	Supported Protocols	Description
CSR Interface	APB, AHB-Lite	The CSR interface is used to configure the IP core. For the APB interface, refer to the AMBA 3 APB Protocol v1.0 Specification for information and the timing diagram. For the AHB-Lite interface, refer to the AMBA 3 AHB-Lite Protocol v1.0 Specification for information and the timing diagram.

2.5. SPI Transaction Support

2.5.1. SPI Transfer Modes

The default SPI transfer mode can be set through the *Default SPI Mode* attribute in the IP GUI. If the corresponding programmable capabilities are enabled in the IP GUI (*Programmable CPOL*, *Programmable CPHA*), clock polarity and clock phase can be set through the XSPI_TGT_CFG0.SPI_CPHA and XSPI_TGT_CFG0.SPI_CPOL registers.

The following figures show the clock idle state and how data is driven and sampled in each mode.

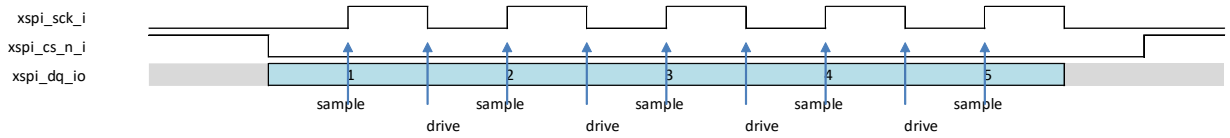

Figure 2.4. SPI Transfer Mode 0 (CPOL = 0, CPHA = 0)

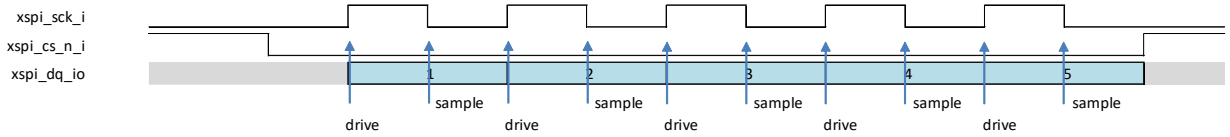

Figure 2.5. SPI Transfer Mode 1 (CPOL = 0, CPHA = 1)

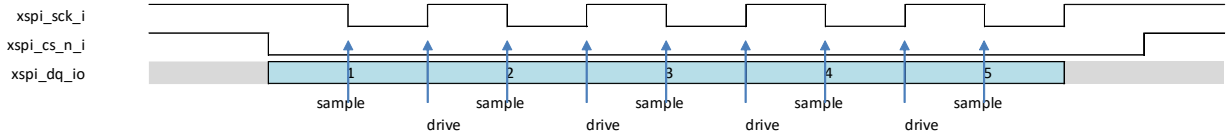

Figure 2.6. SPI Transfer Mode 2 (CPOL = 1, CPHA = 0)

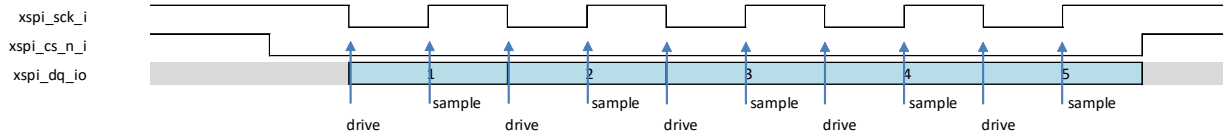

Figure 2.7. SPI Transfer Mode 3 (CPOL = 1, CPHA = 1)

2.5.2. Data I/O Width

The following figures show sample waveforms of a byte transfer across different I/O widths.
Note: The xSPI Target IP supports only the x1 and x4 I/O widths.

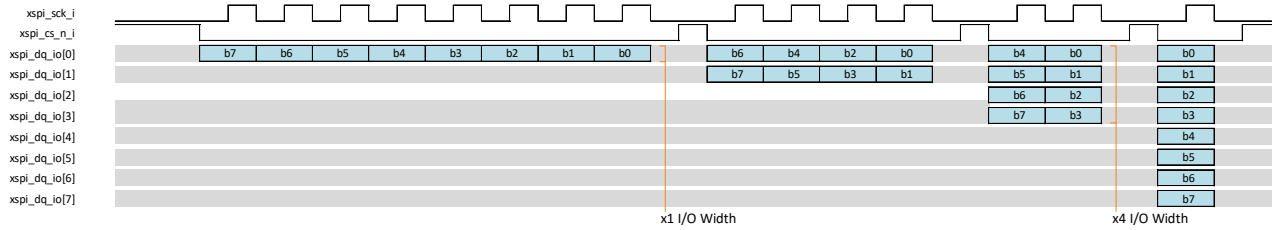

Figure 2.8. Byte Transfer across Different I/O Widths (MSB First)

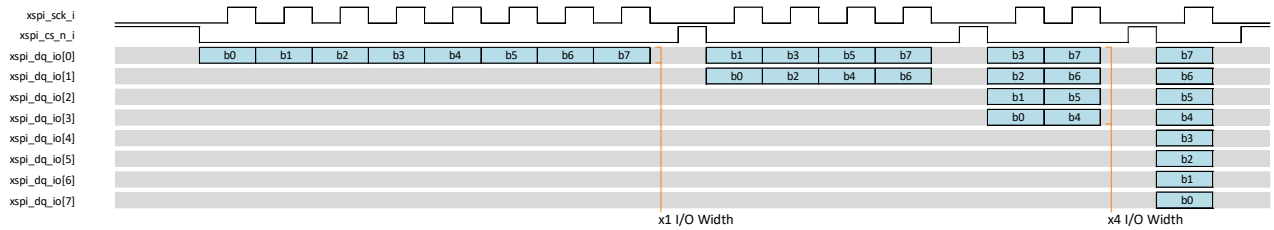

Figure 2.9. Byte Transfer across Different I/O Widths (LSB First)

If *Max Data Lanes* == x4 (Quad) but the IP is configured to operate with an I/O width of 1, the IP behavior follows standard SPI (legacy) pins: *xspi_dq_io[0]* = MOSI, *xspi_dq_io[1]* = MISO, and all other data lines are unused.

2.5.3. Generic SPI Transactions

In a generic SPI write transaction, write data is transmitted from the SPI Controller to the xSPI Target IP and stored in either the Rx FIFO or in the read data register (refer to the Rx Data Register section). Rx data is received on one or more SPI data I/O lines. In a generic SPI read transaction, the return data from the xSPI Target IP to the SPI Controller is stored in either the Tx FIFO or write data register (refer to the Tx Data Register section). Tx data is transmitted on one or more SPI data I/O lines. Dummy cycles may be inserted and controlled as required by the specific protocol or device timing requirements. All SPI data is processed and stored in bytes, regardless of I/O width.

The following figures show example waveform diagrams of typical SPI transactions. For illustration purposes, the *xspi_sck_i* and *xspi_dq_io* waveforms are simplified. The required number of *xspi_sck_i* clock cycles for SPI commands and data varies depending on the number of data lanes used.

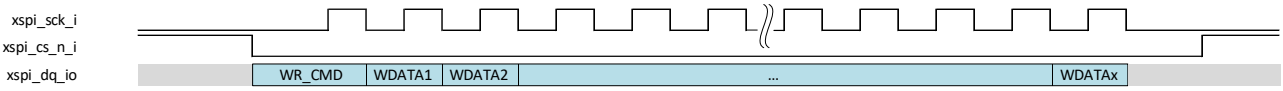

Figure 2.10. Simple Protocol Write Transaction

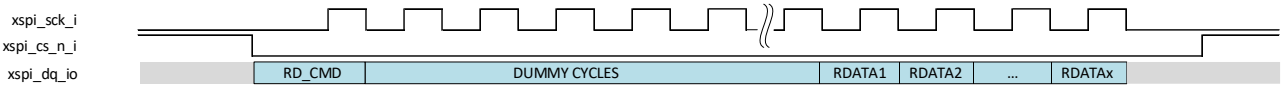

Figure 2.11. Simple Protocol Read Transaction

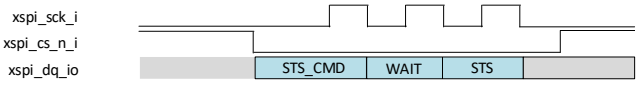

Figure 2.12. Simple Protocol Status Transaction

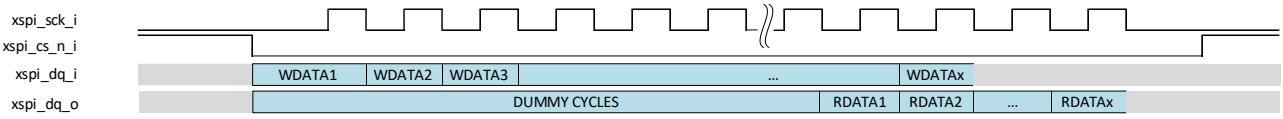

Figure 2.13. Standard SPI (Legacy) Write and Read Transaction with Dummy Cycles

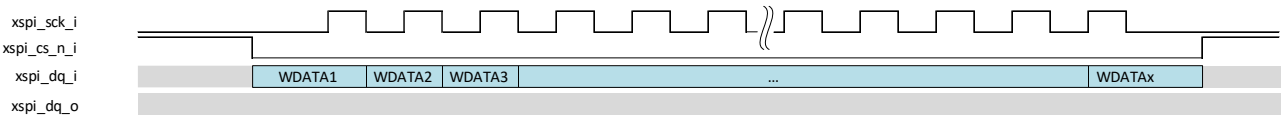

Figure 2.14. Standard SPI (Legacy) Write Only Transaction

2.5.4. Programming Flow

This section describes the programming flow for various SPI transactions. Refer to the Register Description section for details on the configuration and status registers.

Note: All configuration registers must be programmed before the first SPI transaction is initiated.

Reprogramming of configuration registers is required only when changes to the SPI configuration are needed and must be completed prior to any SPI transaction that uses the updated configuration.

Changing the configuration registers during an active SPI transaction is prohibited. The behavior of the IP is not guaranteed if reprogramming is performed while a transaction is ongoing.

2.5.4.1. Simple Protocol SPI Status/Write/Read Operation

This section describes the programming flow for SPI transactions when the IP is configured with *Implementation Preset* == Simple Protocol.

Initial Configuration

The following is the initial configuration operation:

1. Set XSPI_TGT_CFG0, XSPI_TGT_CFG1, XSPI_TGT_TIMING, and XSPI_TGT_CMD_DEF as needed.
2. Enable target interrupts in XSPI_TGT_INT_ENA.

SPI Status Command

The following is the SPI status command operation:

1. Perform the initial configuration operation.
2. Set XSPI_TGT_STATUS depending on user software readiness to process SPI transactions sent to the IP.
   - If from reset, the default values are XSPI_TGT_STATUS.READY = 0 and XSPI_TGT_STATUS.BUSY = 1.
   - If user software is ready to process xSPI Target IP data, set XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0.
3. Send the SPI status command from the SPI Controller to confirm that the IP is ready to process a transaction.
4. Wait for int_o assertion, then read XSPI_TGT_INT_STS to confirm the following interrupt:
   - XFER_START_INT – This indicates that the SPI transaction has started (when xspi_cs_n_i is asserted).
5. (Optional) To confirm the SPI transaction command code: Poll XSPI_TGT_XFER_STS to check FIRST_BYTE_VAL and FIRST_BYTE. FIRST_BYTE_VAL should be equal to the SPI status command value when FIRST_BYTE is 1’b1. This must be done before the end of the current SPI transaction.
6. Clear interrupts.
7. Wait for int_o assertion, then read XSPI_TGT_INT_STS to confirm the following interrupt:
   - XFER_DONE_INT – This indicates that the SPI transaction is complete (when transaction is completed and xspi_cs_n_i is deasserted).
8. Clear interrupts as needed.
9. Repeat Steps 2 to 8 and confirm the updated status.
10. Before performing any SPI write/read, confirm XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0.

SPI Write Command

The following is the SPI write command operation:

1. Perform the initial configuration operation.
2. Confirm XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0 through the SPI status command operation.
3. Send SPI write from the SPI controller with write command equal to the setting in the XSPI_TGT_CMD_DEF register.
4. Wait for int_o assertion, then read XSPI_TGT_INT_STS to confirm the following interrupt:
   - XFER_START_INT – This indicates that the SPI transaction has started (when xspi_cs_n_i is asserted).
5. Set XSPI_TGT_STATUS.READY = 0 and XSPI_TGT_STATUS.BUSY = 1.

6. (Optional) To confirm the SPI transaction command code: Poll XSPI_TGT_XFER_STS to check FIRST_BYTE_VAL and FIRST_BYTE. FIRST_BYTE_VAL should be equal to the SPI write command value when FIRST_BYTE is 1'b1. This should be done before the end of the current SPI transaction.
7. Clear interrupts.
8. Wait for int_o assertion, then read XSPI_TGT_INT_STS to confirm the following interrupts:
   - RX_AVAIL_INT – This indicates that SPI write data is available in the Rx FIFO. If the incoming SPI write data is expected to exceed the FIFO depth, read the XSPI_TGT_RX_DATA register immediately when this interrupt asserts to avoid data loss.
   - RX_FULL_INT – This indicates that Rx FIFO is full. It asserts when the received SPI write data is greater than or equal to the Rx FIFO depth. Use this interrupt only when the incoming SPI write data is expected to be equal to the FIFO depth. Otherwise, use RX_AVAIL_INT.
   - XFER_DONE_INT – This indicates that the SPI transaction is complete (when transaction is completed and xspi_cs_n_i is deasserted).
9. Read data from the XSPI_TGT_RX_DATA register.
10. Clear interrupts as needed.
11. Set XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0 to indicate that the software is ready to process a new SPI command.

SPI Read Command

The following is the SPI read command operation:

1. Perform the initial configuration operation.
2. Confirm XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0 through the SPI status command operation.
3. Send the SPI write command containing the following information: address, command, and other user-defined information. Interpretation of this user-defined information shall be handled by user software.
4. Set XSPI_TGT_STATUS.READY = 0 and XSPI_TGT_STATUS.BUSY = 1 if the software needs additional time to process the command.
5. Write data for SPI read to the XSPI_TGT_TX_DATA register according to information received from SPI write.
6. Set XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0 to indicate that the requested data is ready.
7. Confirm XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0 through the SPI status command operation.
8. Send SPI read from the SPI controller with read command equal to the setting in the XSPI_TGT_CMD_DEF register. SPI read data transmission will continue after the set dummy cycles.
9. Wait for int_o assertion, then read XSPI_TGT_INT_STS to confirm the following interrupt:
   - XFER_START_INT – This indicates that the SPI transaction has started (when xspi_cs_n_i is asserted).
10. (Optional) To confirm the SPI transaction command code: Poll XSPI_TGT_XFER_STS to check FIRST_BYTE_VAL and FIRST_BYTE. FIRST_BYTE_VAL should be equal to the SPI read command value when FIRST_BYTE is 1'b1. This should be done before the end of the current SPI transaction.
11. Clear interrupts.
12. Wait for int_o assertion, then read the XSPI_TGT_INT_STS register to confirm the following interrupts:
    - TX_EMPTY_INT – This indicates that all data written to Tx FIFO has been transmitted.
      If the required transmit length exceeds the Tx FIFO depth, for *Max Data Lanes* == x1 (Standard) and SPI clock frequency ≤ 1/8 of the system clock frequency:
      - Write the remaining data to XSPI_TGT_TX_DATA when the interrupt asserts, or when transmit FIFO space becomes available (XSPI_TGT_FIFO_STS.TX_FIFO_FULL deasserts).

      Otherwise:
      - Write the remaining data to XSPI_TGT_TX_DATA as soon as transmit FIFO space becomes available (XSPI_TGT_FIFO_STS .TX_FIFO_FULL deasserts).
    - XFER_DONE_INT – This indicates that the SPI transaction is complete (when transaction is completed and xspi_cs_n_i is deasserted).
13. Clear interrupts as needed.

14. Set XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0 to indicate that the software is ready to process a new SPI command.

I/O Mode Change

If the IP is configured with *Max Data Lanes* greater than x1, it can support any I/O width smaller than the *Max Data Lanes* setting.

1. Perform the initial configuration operation.
2. Confirm XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0 through the SPI status command operation.
3. Send the SPI write command containing the I/O mode change information: address, command, and other user-defined information. Interpretation of this user-defined information shall be handled by user software.
4. Set XSPI_TGT_STATUS.READY = 0 and XSPI_TGT_STATUS.BUSY = 1 to indicate I/O mode change is not yet completed.
5. Set XSPI_TGT_CFG0.IO_WIDTH accordingly.
6. Set XSPI_TGT_STATUS.READY = 1 and XSPI_TGT_STATUS.BUSY = 0 to indicate that I/O mode change is complete and software is ready to process the SPI command in the new I/O mode.

2.5.4.2. Standard SPI (Legacy) Transmit and Receive Operation

This section describes the programming flow for SPI transactions when the IP is configured with *Implementation Preset* == Standard SPI (Legacy).

Initial Configuration

The following is the initial configuration operation:

1. Program XSPI_TGT_CFG0 and XSPI_TGT_TIMING with the required configuration.
   - Set XSPI_TGT_TIMING.DUMMY_CYCLES to provide sufficient cycles to receive the SPI address and command, and to prepare the initial transmit response when applicable.
   - If required, preload XSPI_TGT_TX_DATA with initial or default data.
2. Enable the necessary target interrupts in XSPI_TGT_INT_ENA (for example, XFER_START_INT, RX_AVAIL_INT, RX_FULL_INT, and XFER_DONE_INT).

Receive Address and Command

The following is the receive address and command operation:

1. Start the SPI transaction from the SPI controller.
2. If FIFO is enabled:
   a. Wait for int_o assertion, then read XSPI_TGT_INT_STS to confirm the following interrupts:
      - RX_AVAIL_INT – This indicates that SPI write data is available in the Rx FIFO.
      - XFER_DONE_INT – This indicates that the SPI transaction is complete (when transaction is completed and xspi_cs_n_i is deasserted).
   b. Read all available bytes from XSPI_TGT_RX_DATA.
   c. Clear interrupts.
   d. Repeat Steps 2a through 2c until all address and command bytes have been received.
3. If FIFO is disabled:
   a. Wait for int_o assertion, then read XSPI_TGT_INT_STS to confirm the following interrupt:
      - RX_AVAIL_INT – This indicates that SPI write data is available in the Rx register.
   b. Read one byte from XSPI_TGT_RX_DATA.
   c. Clear interrupts.
   d. Repeat Steps 3a through 3c until all address and command bytes have been received.

Prepare Transmit Data (If Controller Read Is Expected)

The following is the prepare transmit data operation:

1. If FIFO is enabled:

- Based on the received address and command, load the initial response data into XSPI_TGT_TX_DATA.

2. If FIFO is disabled:
   - Based on the received address and command, write one byte of the initial response data into XSPI_TGT_TX_DATA.

Ensure this operation is completed before the programmed dummy cycles expire.

Receive or Transmit Operation

The following is the receive or transmit operation:

1. Wait for int_o assertion, then read XSPI_TGT_INT_STS.
2. Service the receive operation if XSPI_TGT_INT_STS.RX_AVAIL_INT or XSPI_TGT_INT_STS.RX_FULL_INT is asserted:
   - If FIFO is enabled: Read all available data from XSPI_TGT_RX_DATA.
   - If FIFO is disabled: Read one byte from XSPI_TGT_RX_DATA.
3. Service the transmit operation if controller read is expected.
   - If FIFO is enabled and the required transmit length exceeds the Tx FIFO depth: Continue writing data to XSPI_TGT_TX_DATA as space becomes available (XSPI_TGT_FIFO_STS .TX_FIFO_FULL deasserts).
   - If FIFO is disabled: Write data to XSPI_TGT_TX_DATA after XSPI_TGT_INT_STS.TX_EMPTY_INT asserts.
4. Clear all serviced interrupt status bits.
5. Repeat Steps 1 through 4 until XSPI_TGT_INT_STS.XFER_DONE_INT is asserted.
6. Read any remaining valid data from XSPI_TGT_RX_DATA.
7. Clear interrupts.

3. IP Parameter Description

The configurable attributes of the xSPI Target IP are shown in the following tables. You can configure the IP by setting the attributes accordingly in the IP Catalog Module/IP Block wizard of the Lattice Radiant software.

Wherever applicable, default values are in bold.

3.1. General

Table 3.1. General Attributes

Attribute	Selectable Values	Description
Implementation Preset		
Implementation Preset	Standard SPI (Legacy), **Simple Protocol**	Preconfigures the required settings for a particular protocol.
Clock Configuration		
System Clock Frequency (MHz)	12–**100**	Sets the system clock (clk_i) frequency. Refer to the Clocking section for more information on the relationship between system clock and SPI clock frequency.
Maximum SPI Clock (MHz)	12–**50**	Sets the maximum SPI (xspi_sck_i) frequency.
Bus Interface		
CSR Interface	**APB**, AHB-Lite	Selects the bus interface.
FIFO Configuration		
Enable FIFOs	**Checked**, Unchecked	Adds FIFO for transmit and receive data. Refer to the Tx Data Register and Rx Data Register sections for more information. Selectable only when *Implementation Preset* == Standard SPI (Legacy). Uncheck this option to minimize resource usage. Use only when SPI clock frequency is slow to avoid data loss. Ensure that all SPI data is handled correctly as specified in the Standard SPI (Legacy) Transmit and Receive Operation section.
FIFO Implementation	**EBR**, LUT, HARD IP	Specifies the FIFO implementation type. Applicable when *Enable FIFOs* == Checked.
FIFO Depth	16, 32, **64**, 128, 256, 512	Specifies the depth of transmit and receive FIFO. Applicable when *Enable FIFOs* == Checked.
IO Primitive		
Use IO Primitive	**Checked**, Unchecked	Includes the I/O primitive instance inside the IP. This means that SPI data pins are seen as bidirectional I/O ports at the top level. When this option is disabled, the SPI tristate data control signals are exposed at the top level as ports. Applicable only when *Max Data Lanes* > x1 (Standard).

3.2. xSPI Configuration

Table 3.2. xSPI Configuration Attributes

Attribute	Selectable Values	Description
SPI Configuration		
Max Data Lanes	**x1 (Standard)**, x4 (Quad)	Sets the maximum number of data I/O lines. x1 (Standard) – 1 data pin for transmit and 1 data pin for receive x4 (Quad) – 4 bidirectional data pins for transmit and receive. x4 is selectable only when *Implementation Preset* == Simple Protocol.
Default Dummy Cycles	**8**–255	Default dummy cycles during SPI transaction.
TX Empty Value	0x00, **0xFF**	Returned data when Tx FIFO is read while empty.

Attribute	Selectable Values	Description
CS Polarity	**Active Low (Default)**, Active High	Default value for chip select polarity. Applicable only when *Implementation Preset* == Standard SPI (Legacy).
Default SPI Mode	**Mode 0 (CPOL=0, CPHA=0)**, Mode 1 (CPOL=0, CPHA=1), Mode 2 (CPOL=1, CPHA=0), Mode 3 (CPOL=1, CPHA=1)	Default value for SPI clock polarity and phase. Applicable only when *Max Data Lanes* == x1 (Standard).
Default Bit Order	**MSB First**, LSB First	Default SPI data bit order. Applicable only when *Max Data Lanes* == x1 (Standard).
Programmable Settings		
Programmable CPOL	Checked, **Unchecked**	When checked, SPI clock polarity is programmable through configuration register 0. Refer to the Configuration Register 0 section for more information. Applicable only when *Max Data Lanes* == x1 (Standard).
Programmable CPHA	Checked, **Unchecked**	When checked, SPI clock phase is programmable through configuration register 0. Refer to the Configuration Register 0 section for more information. Applicable only when *Max Data Lanes* == x1 (Standard).
Programmable Bit Order	Checked, **Unchecked**	When checked, SPI data bit order is programmable through configuration register 0. Refer to the Configuration Register 0 section for more information. Applicable only when *Max Data Lanes* == x1 (Standard).
Programmable Commands	Checked, **Unchecked**	When checked, SPI command configurations are programmable through the command definition register. Refer to the Command Definition Register section for more information. Applicable only when *Implementation Preset* == Simple Protocol.
Command Configuration		
Default Write Command	02	Default value for the IP to recognize SPI write command. Applicable only when *Implementation Preset* == Simple Protocol.
Default Read Command	03	Default value for the IP to recognize SPI read command. Applicable only when *Implementation Preset* == Simple Protocol.
Default Status Command	05	Default value for the IP to recognize SPI status command. Applicable only when *Implementation Preset* == Simple Protocol.
Reset Configuration		
Enable In-band Reset	**Checked**, Unchecked	When checked, IP supports in-band reset. Applicable only when *Implementation Preset* == Simple Protocol.
Enable Reset Pin	Checked, **Unchecked**	When checked, enables optional SPI target reset pin xspi_tgt_rst_n_i.

3.3. DEBUG: IP Parameters

This tab shows read-only attributes. These attributes are informational only and are not enumerated in this document because they are redundant with the attributes in the General and xSPI Configuration sections.

3.4. IP Parameter Settings for Example Use Cases

Table 3.3. Attributes for Example Use Cases

Attribute	Standard SPI (Legacy)	Simple Protocol (Quad SPI)
Implementation Preset		
Implementation Preset	Standard SPI (Legacy)	Simple Protocol
Clock Configuration		
System Clock Frequency (MHz)	100	100
Maximum SPI Clock (MHz)	50	50
Bus Interface		
CSR Interface	APB	AHB-Lite
FIFO Configuration		
Enable FIFOs	Checked	Checked
FIFO Implementation	EBR	EBR
FIFO Depth	64	64
IO Primitive		
Use IO Primitive	—	Checked
SPI Configuration		
Max Data Lanes	x1 (Standard)	x4 (Quad)
Default Dummy Cycles	8	8
Tx Empty Value	0xFF	0xFF
CS Polarity	Active Low (Default)	Active Low (Default)
Default SPI Mode	Mode 0 (CPOL=0, CPHA=0)	Mode 0 (CPOL=0, CPHA=0)
Default Bit Order	MSB First	MSB First
Programmable Settings		
Programmable CPOL	Checked	—
Programmable CPHA	Checked	—
Programmable Bit Order	Checked	—
Programmable Commands	—	Checked
Command Configuration		
Default Write Command	—	02
Default Read Command	—	03
Default Status Command	—	05
Reset Configuration		
Enable In-band Reset	—	Checked
Enable Reset Pin	Unchecked	Checked

4. Signal Description

This section describes the xSPI Target IP ports.

Table 4.1. xSPI Target Ports

Port	Type	Reset[1]	Description
rst_n_i	Input	—	Asynchronous active low reset
clk_i	Input	—	System clock For more information, refer to the Clocking section.
SPI Interface			
xspi_sck_i	Input	—	SPI input clock from external SPI controller
xspi_cs_n_i	Input	—	SPI chip select Active-low signal for selecting the SPI target.
xspi_tgt_rst_n_i	Input	—	Asynchronous active low reset This is an optional pin to reset the SPI target. Available only when *Enable Reset Pin* is checked in the IP GUI.
SPI Interface – Internal I/O Primitive and Max Data Lanes == x4 (Quad)[2]			
xspi_dq_io[3:0]	Input/ Output	Hi-Z	Bidirectional SPI data signals Controller and target drivers operate in push-pull mode. The number of data pins depends on the selected maximum number of data lanes in the IP GUI. • x1 – Receive data on data bit[0]; transmit data on data bit[1] • x4 – Transmit data and receive data are 4 bits The default state of this port is undriven (tri-stated). If the I/O pin has a pull-up resistor, the default state becomes high (1); if it has a pull-down resistor, the default state becomes low (0). Applicable only when *Max Data Lanes* == x4 (Quad).
SPI Interface – External I/O Primitive and Max Data Lanes == x4 (Quad)[3]			
xspi_dq_i[3:0]	Input	—	SPI data input The number of data pins depends on the selected maximum number of data lanes in the IP GUI. • x1 – Receive data on data bit[0] • x4 – Receive data is 4 bits
xspi_dq_o[3:0]	Output	Low	SPI data output The number of data pins depends on the selected maximum number of data lanes in the IP GUI. • x1 – Transmit data on data bit[1] • x4 – Transmit data is 4 bits
xspi_dq_oe_o[3:0]	Output	Low	Active high SPI data output enable The number of data pins depends on the selected maximum number of data lanes in the IP GUI. • x1 – Transmit data output enable on data bit[1] • x4 – Transmit data output enable is 4 bits
SPI Interface – Max Data Lanes == x1 (Standard)			
xspi_dq_i	Input	—	SPI data input
xspi_dq_o	Output	Hi-Z	SPI data output
Interrupt Interface			
int_o	Output	Low	Level-sensitive active-high interrupt signal This signal asserts when any of the enabled interrupt status bits is asserted.
APB Interface			
s_apb_psel_i	Input	—	APB select signal Indicates that the target device is selected, and a data transfer is required.
s_apb_penable_i	Input	—	APB enable signal Indicates the second and subsequent cycles of an APB transfer.

Port	Type	Reset[1]	Description
s_apb_paddr_i[31:0]	Input	—	APB address signal Address must be DWORD aligned (bit[1:0] = 0).
s_apb_pwrite_i	Input	—	APB write signal
s_apb_pwdata_i[31:0]	Input	—	APB write data signal
s_apb_pready_o	Output	High	APB ready signal Indicates transfer completion. Completer uses this signal to extend an APB transfer.
s_apb_prdata_o[31:0]	Output	Low	APB read data signal
s_apb_pslverr_o	Output	Low	APB error signal This signal is unused and tied to 0.
AHB-Lite Interface			
s_ahbl_hsel_i	Input	—	AHB-Lite select signal Indicates the device is selected and transfer is required.
s_ahbl_haddr_i[31:0]	Input	—	AHB-Lite address signal Address must be DWORD aligned (bit[1:0] = 0).
s_ahbl_htrans_i[1:0]	Input	—	AHB-Lite transfer type signal Indicates the transfer type of the current transfer.
s_ahbl_hwrite_i	Input	—	AHB-Lite direction signal
s_ahbl_hsize_i[2:0]	Input	—	AHB-Lite transfer size signal 1-word transfer size is supported (SIZE = 3'd2).
s_ahbl_hburst_i[2:0]	Input	—	AHB-Lite burst type signal Indicates if the transfer is a single transfer or forms part of a burst. Only burst type 0 is supported.
s_ahbl_hprot_i[3:0]	Input	—	AHB-Lite protection control signal This signal is unused and should be tied to 0.
s_ahbl_hmastlock_i	Input	—	AHB-Lite lock signal This signal is unused and should be tied to 0.
s_ahbl_hwdata_i[31:0]	Input	—	AHB-Lite write data signal
s_ahbl_hready_i	Input	—	AHB-Lite ready input signal Indicates data phase of previous transfer is complete.
s_ahbl_hreadyout_o	Output	High	AHB-Lite ready output signal Indicates transfer completion.
s_ahbl_hresp_o	Output	Low	AHB-Lite transfer response signal This signal is unused and tied to 0.
s_ahbl_hrdata_o[31:0]	Output	Low	AHB-Lite read data signal

Notes:

1. For multi-bit signals, the indicated reset value or state applies to all bits.
2. Bidirectional SPI interface is available only when *Use IO Primitive* is checked in the IP GUI.
3. External I/O SPI interface is available only when *Use IO Primitive* is unchecked in the IP GUI.

5. Register Description

This section defines the configuration, status, and control registers of the xSPI Target IP. The xSPI Target IP registers have a total address space region of 1 KiB.

Table 5.1 defines the register access types. Table 5.2 lists the address map and specifies the registers available. These registers are accessible through the selected CSR interface (APB, AHB-Lite).

Table 5.1. Register Access Types

Access Type	Access Type Abbreviation	Behavior on Read Access	Behavior on Write Access
Read only	RO	Returns register value	Ignores write access
Write only	WO	Returns 0	Updates register value
Read and write	RW	Returns register value	Updates register value
Read and write 1 to clear	RW1C	Returns register value	Writing 1'b1 on register bit clears the bit to 1'b0. Writing 1'b0 on register bit is ignored.
Reserved	RSVD	Returns 0	Ignores write access

Table 5.2. Summary of xSPI Target IP Core Registers

Offset Address	Register Name	Description
Configuration Registers		
0x000	XSPI_TGT_IP_ID	IP Identification Register
0x004	XSPI_TGT_CFG0	Configuration Register 0
0x008	XSPI_TGT_CFG1	Configuration Register 1
0x00C	XSPI_TGT_TIMING	Timing Configuration Register
0x010	XSPI_TGT_INT_ENA	Interrupt Enable Register
0x014–0x02C	Reserved	Reserved
0x030	XSPI_TGT_CMD_DEF	Command Definition Register
Status Registers		
0x100	XSPI_TGT_INT_STS	Interrupt Status Register
0x104	XSPI_TGT_XFER_STS	Transaction Status Register
0x108	XSPI_TGT_FIFO_STS	FIFO Status Register
0x10C–0x1FC	Reserved	Reserved
Control Registers		
0x200	XSPI_TGT_TX_DATA	Tx Data Register
0x204	XSPI_TGT_RX_DATA	Rx Data Register
0x208–0x214	Reserved	Reserved
0x218	XSPI_TGT_INT_SET	Interrupt Set (Debug) Register
0x21C	XSPI_TGT_SOFT_RST	Soft Reset Register
0x220–0x228	Reserved	Reserved
0x22C	XSPI_TGT_STATUS	Target Status Auto Response Register
0x230–0x3FC	Reserved	Reserved

5.1. Configuration Registers

5.1.1. IP Identification Register

Table 5.3. IP Identification Register

Field	Name	Description	Access	Default
[31:0]	IP_ID	Unique IP identifier. Reads as ASCII 'XSPT' (0x58535054). Used by software to verify the correct IP is present at the expected address.	RO	0x58535054

5.1.2. Configuration Register 0

Table 5.4. Configuration Register 0

Field	Name	Description	Access	Default
[31:9]	Reserved	Reserved	RSVD	0x0
[8]	SPI_LSBF	Bit order within each byte 0 – MSB first (standard) 1 – LSB first Writeable only when *Programmable Bit Order* is checked.	RW	*Default Bit Order* in GUI
[7]	SPI_CPHA	Clock phase for legacy SPI mode 0 – Sample on first edge 1 – Sample on second edge Combined with CPOL to define SPI modes 0 through 3. Writeable only when *Programmable CPHA* is checked.	RW	CPHA from *Default SPI Mode* in GUI
[6]	SPI_CPOL	Clock polarity for legacy SPI mode 0 – Clock idle low 1 – Clock idle high Applicable only when *Implementation Preset* == Standard SPI (Legacy). Writeable only when *Programmable CPOL* is checked.	RW	CPOL from *Default SPI Mode* in GUI
[5:4]	IO_WIDTH	Active I/O width 00 – x1 (single) 10 – x4 (quad) User software updates this when the controller sends the mode change command. Hardware uses this information for data serialization.	RW	0x0
[3:1]	Reserved	Reserved	RSVD	0x0
[0]	EN_TARGET	Global enable for xSPI target 0 – All SPI write operations on the bus are not stored in the Rx data register. IP will not drive the SPI output data line during SPI read and status commands. 1 – Normal operation. Set after target configuration is complete.	RW	0x0

5.1.3. Configuration Register 1

Table 5.5. Configuration Register 1

Field	Name	Description	Access	Default
[31:25]	Reserved	Reserved	RSVD	0x0
[24]	EN_SW_RESET	Software reset command recognition. 0 – Disable; if using FFh for other purposes 1 – Enable; IP responds to the FFh reset command sequence Available only when *Implementation Preset* == Simple Protocol. Writeable only when *Enable In-band Reset* is checked.	RW	0x1 if *Enable In-band Reset* is checked; 0x0 otherwise.
[23:0]	Reserved	Reserved	RSVD	0x0

5.1.4. Timing Configuration Register

Table 5.6. Timing Configuration Register

Field	Name	Description	Access	Default
[31:8]	Reserved	Reserved	RSVD	0x0
[7:0]	DUMMY_CYCLES	Dummy (turnaround) cycles The number of clock cycles between receiving command or address and starting data transmission. A minimum of eight cycles is required for proper Tx pipeline timing. Allows time for bus turnaround and data fetch.	RW	*Default Dummy Cycles* in GUI

5.1.5. Interrupt Enable Register

Table 5.7. Interrupt Enable Register

Field	Name	Description	Access	Default
[31:18]	Reserved	Reserved	RSVD	0x0
[17]	RD_ON_RX_EMPTY_INT_EN	Enables interrupt when read is done from empty Rx FIFO. Available only when *Enable FIFOs* is checked.	RW	0x0
[16]	WR_ON_TX_FULL_INT_EN	Enables interrupt when write is done to full Tx FIFO. Available only when *Enable FIFOs* is checked.	RW	0x0
[15:11]	Reserved	Reserved	RSVD	0x0
[10]	RESET_DET_INT_EN	Enables interrupt when in-band reset sequence is detected on CS pin. Available only when *Enable In-band Reset* is checked.	RW	0x0
[9:8]	Reserved	Reserved	RSVD	0x0
[7]	ERR_INT_EN	Enables general error interrupt. Aggregates multiple error sources. Check ERR_CODE for the specific error type.	RW	0x0
[6]	RX_OFLOW_INT_EN	Enables interrupt on Rx overflow. Indicates received data lost because Rx buffer was full. Critical error. Available only when *Enable FIFOs* is checked.	RW	0x0
[5]	RX_FULL_INT_EN	Enables interrupt when Rx buffer becomes full. Service immediately to prevent overflow and data loss. Available only when *Enable FIFOs* is checked.	RW	0x0
[4]	RX_AVAIL_INT_EN	Enables interrupt when Rx data is available. Use for polling-free receive notification.	RW	0x0

Field	Name	Description	Access	Default
[3]	TX_UFLOW_INT_EN	Enables interrupt on Tx underflow. Indicates controller requested data but Tx buffer was empty. Critical error – transmitted data was undefined. Available only when *Enable FIFOs* is checked.	RW	0x0
[2]	TX_EMPTY_INT_EN	Enables interrupt when Tx buffer becomes empty. Use to refill Tx data before underflow occurs.	RW	0x0
[1]	XFER_START_INT_EN	Enables interrupt when SPI transaction starts (CS asserts). Use for low-latency response preparation.	RW	0x0
[0]	XFER_DONE_INT_EN	Enables interrupt when SPI transaction completes (CS deasserts). Use to trigger user software processing of received data.	RW	0x0

5.1.6. Command Definition Register

Table 5.8. Command Definition Register

Field	Name	Description	Access	Default
[31:24]	Reserved	Reserved	RSVD	0x0
[23:16]	CMD_STATUS	Status read command opcode When received, target automatically transmits XSPI_TGT_STATUS contents after four dummy cycles. Ensure that the value of this register is not equal to CMD_READ and CMD_WRITE. Available only when *Implementation Preset* == Simple Protocol. Writeable only when *Programmable Commands* is checked.	RW	*Default Status Command* in GUI
[15:8]	CMD_READ	Read command opcode When this byte is received, target transmits data after dummy cycles. Ensure that the value of this register is not equal to CMD_STATUS and CMD_WRITE. Available only when *Implementation Preset* == Simple Protocol. Writeable only when *Programmable Commands* is checked.	RW	*Default Read Command* in GUI
[7:0]	CMD_WRITE	Write command opcode When this byte is received as first byte, target only receives data (no transmission). Ensure that the value of this register is not equal to CMD_STATUS and CMD_READ. Available only when *Implementation Preset* == Simple Protocol. Writeable only when *Programmable Commands* is checked.	RW	*Default Write Command* in GUI

5.2. Status Registers

5.2.1. Interrupt Status Register

Table 5.9. Interrupt Status Register

Field	Name	Description	Access	Default
[31:18]	Reserved	Reserved	RSVD	0x0

Field	Name	Description	Access	Default
[17]	RD_ON_RX_EMPTY_INT	Rx data read error Set to 1 when read is done from empty Rx FIFO. Return data is invalid. Critical error. Available only when *Enable FIFOs* is checked.	RW1C	0x0
[16]	WR_ON_TX_FULL_INT	Tx data write error Set to 1 when write is done to full Tx FIFO. Write data may be lost. Critical error. Available only when *Enable FIFOs* is checked.	RW1C	0x0
[15:11]	Reserved	Reserved	RSVD	0x0
[10]	RESET_DET_INT	In-band reset detected Set to 1 when in-band reset is detected and EN_SW_RESET is enabled. Available only when *Enable In-band Reset* is checked.	RW1C	0x0
[9:8]	Reserved	Reserved	RSVD	0x0
[7]	ERR_INT	General error occurred Aggregates multiple error sources. Check XFER_STS.ERR_CODE to determine specific errors. Remains asserted until the SPI transaction associated with the error has completed (CS is deasserted).	RW1C	0x0
[6]	RX_OFLOW_INT	Rx overflow occurred Set to 1 when received SPI data lost because Rx buffer was full. Critical error. Available only when *Enable FIFOs* is checked.	RW1C	0x0
[5]	RX_FULL_INT	Rx buffer full interrupt Set to 1 when Rx FIFO becomes full. Service immediately. Available only when *Enable FIFOs* is checked.	RW1C	0x0
[4]	RX_AVAIL_INT	Rx data available interrupt When FIFO is enabled in GUI: Set to 1 when Rx FIFO receives data. When FIFO is disabled in GUI: Set to 1 when XSPI_TGT_RX_DATA register receives one byte data from the SPI controller.	RW1C	0x0
[3]	TX_UFLOW_INT	Tx underflow occurred Set to 1 when data requested but TX buffer is empty. Transmitted data is undefined. Critical error. Available only when *Enable FIFOs* is checked.	RW1C	0x0
[2]	TX_EMPTY_INT	Tx buffer empty interrupt. When FIFO is enabled in GUI: Set to 1 when Tx FIFO becomes empty from non-empty status. Indicates that the last data has been read from the Tx FIFO and it is now empty. When FIFO is disabled in GUI: Set to 1 when data in XSPI_TGT_TX_DATA has been transmitted.	RW1C	0x0
[1]	XFER_START_INT	Transaction started Set when CS asserts, beginning a transaction.	RW1C	0x0
[0]	XFER_DONE_INT	Transaction complete Set when CS deasserts, ending a transaction.	RW1C	0x0

5.2.2. Transaction Status Register

Table 5.10. Transaction Status Register

Field	Name	Description	Access	Default
[31:28]	Reserved	Reserved	RSVD	0x0
[27:24]	ERR_CODE	Last error code Valid when ERR interrupt set. 0 – No error 1 – Command error; command received is not equal to any commands defined in XSPI_TGT_CMD_DEF.	RO	0x0
[23:18]	RSVD	Reserved bits	RO	0x0
[17:16]	IO_MODE	Current I/O mode 00 – x1 single 10 – x4 quad May change during transaction for multi-I/O configurations.	RO	0x0
[15:8]	FIRST_BYTE_VAL	First byte value Contains command/opcode byte of current or last transaction. Use to determine transaction type.	RO	0x0
[7:2]	Reserved	Reserved	RSVD	0x0
[1]	FIRST_BYTE	First byte received flag Set after first byte of current transaction is captured. Indicates FIRST_BYTE_VAL is valid.	RO	0x0
[0]	XFER_ACTIVE	Transaction in progress Reads 1 while CS is asserted (low). Transaction is active.	RO	0x0

5.2.3. FIFO Status Register

Table 5.11. FIFO Status Register

Field	Name	Description	Access	Default
[31:28]	Reserved	Reserved	RSVD	0x0
[27]	RX_FIFO_FULL	Rx FIFO full flag Reads 1 when Rx FIFO is full. Read data immediately to prevent overflow. Available only when *Enable FIFOs* is checked.	RO	0x0
[26]	RX_FIFO_EMPTY	Rx FIFO empty flag Reads 1 when Rx FIFO contains no data. Do not read until data available. Available only when *Enable FIFOs* is checked.	RO	0x1
[25:12]	Reserved	Reserved	RSVD	0x0
[11]	TX_FIFO_FULL	Tx FIFO full flag Reads 1 when Tx FIFO is full. Do not write more data until space is available. Available only when *Enable FIFOs* is checked.	RO	0x0
[10]	TX_FIFO_EMPTY	Tx FIFO empty flag Reads 1 when Tx FIFO contains no data. Do not start transmission until data is loaded. Available only when *Enable FIFOs* is checked.	RO	0x1
[9:0]	Reserved	Reserved	RSVD	0x0

5.3. Control Registers

5.3.1. Tx Data Register

Table 5.12. Tx Data Register

Field	Name	Description	Access	Default
[31:8]	Reserved	Reserved	RSVD	0x0
[7:0]	TX_DATA	Tx data write port When EN_FIFO = 1: Writes to Tx FIFO. When EN_FIFO = 0: Writes to simple Tx register. Write the return data of the IP during SPI read transaction to this register.	WO	0x0

5.3.2. Rx Data Register

Table 5.13. Rx Data Register

Field	Name	Description	Access	Default
[31:8]	Reserved	Reserved	RO	0x0
[7:0]	RX_DATA	Rx data read port When EN_FIFO=1: Reads from Rx FIFO. When EN_FIFO=0: Reads from simple Rx register. Received data during SPI write transaction is stored in this register.	RO	0x0

5.3.3. Interrupt Set (Debug) Register

Table 5.14. Interrupt Set (Debug) Register

Field	Name	Description	Access	Default
[31:18]	Reserved	Reserved	RO	0x0
[17]	RD_ON_RX_EMPTY_INT_SET	Force RD_ON_RX_EMPTY interrupt Write 1 to set interrupt for testing. Available only when *Enable FIFOs* is checked.	WO	0x0
[16]	WR_ON_TX_FULL_INT_SET	Force WR_ON_TX_FULL interrupt Write 1 to set interrupt for testing. Available only when *Enable FIFOs* is checked.	WO	0x0
[15:11]	Reserved	Reserved	RSVD	0x0
[10]	RESET_DET_INT_SET	Force RESET_DET interrupt Write 1 to set interrupt for testing Available only when *Enable In-band Reset* is checked.	WO	0x0
[9:8]	Reserved	Reserved	RSVD	0x0
[7]	ERR_INT_SET	Force ERR interrupt Write 1 to set interrupt for testing.	WO	0x0
[6]	RX_OFLOW_INT_SET	Force RX_OFLOW interrupt Write 1 to set interrupt for testing. Available only when *Enable FIFOs* is checked.	WO	0x0
[5]	RX_FULL_INT_SET	Force RX_FULL interrupt Write 1 to set interrupt for testing. Available only when *Enable FIFOs* is checked.	WO	0x0
[4]	RX_AVAIL_INT_SET	Force RX_AVAIL interrupt Write 1 to set interrupt for testing. Available only when *Enable FIFOs* is checked.	WO	0x0

Field	Name	Description	Access	Default
[3]	TX_UFLOW_INT_SET	Force TX_UFLOW interrupt Write 1 to set interrupt for testing. Available only when *Enable FIFOs* is checked.	WO	0x0
[2]	TX_EMPTY_INT_SET	Force TX_EMPTY interrupt Write 1 to set interrupt for testing. Available only when *Enable FIFOs* is checked.	WO	0x0
[1]	XFER_START_INT_SET	Force XFER_START interrupt Write 1 to set interrupt for testing.	WO	0x0
[0]	XFER_DONE_INT_SET	Force XFER_DONE interrupt Write 1 to set interrupt for testing.	WO	0x0

Note: Refer to the Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide (FPGA-UG-02039) for more information on the interrupt signal.

5.3.4. Soft Reset Register

Table 5.15. Soft Reset Register

Field	Name	Description	Access	Default
[31:4]	Reserved	Reserved	RSVD	0x0
[3]	CSR_RST	CSR block reset Write 1 to reset all configuration, status, and control registers to defaults. Self-clearing. Does not affect FIFOs or IP state machines.	RW	0x0
[2]	RX_FIFO_RST	Rx FIFO reset Write 1 to clear Rx FIFO only. Self-clearing. Configuration preserved. Available only when *Enable FIFOs* is checked.	RW	0x0
[1]	TX_FIFO_RST	Tx FIFO reset Write 1 to clear Tx FIFO only. Self-clearing. Configuration preserved. Available only when *Enable FIFOs* is checked.	RW	0x0
[0]	IP_RST	IP core reset Write 1 to reset IP logic (FIFOs and state machines). Self-clearing. CSR configuration preserved. Use CSR_RST to reset registers.	RW	0x0

5.3.5. Target Status Auto Response Register

Table 5.16. Target Status Auto Response Register

Field	Name	Description	Access	Default
[31:8]	Reserved	Reserved	RSVD	0x0
[7]	READY	Ready flag Software sets to 1 when device is ready for new commands. Allows host to check readiness before sending command. Available only when *Implementation Preset* == Simple Protocol.	RW	0x0
[6]	Reserved	Reserved	RSVD	0x0
[5:3]	ERR_CODE	Error code Software writes error type when ERR = 1. 0 – None 1 – Invalid command Available only when *Implementation Preset* == Simple Protocol.	RW	0x0

Field	Name	Description	Access	Default
[2]	ERR	Error flag Software sets to 1 if request failed. Controller checks after BUSY clears. Available only when *Implementation Preset* == Simple Protocol.	RW	0x0
[1]	ACK	Acknowledge flag Software sets to 1 to confirm request received. Provides immediate feedback to the controller that command has been received. When this is set, the controller may opt to reduce frequency of status polling. Available only when *Implementation Preset* == Simple Protocol.	RW	0x0
[0]	BUSY	Busy flag Software sets to 1 when processing request. Clears to 0 when ready. Available only when *Implementation Preset* == Simple Protocol.	RW	0x1

6. Example Design

The xSPI Target example design allows you to compile, simulate, and test the xSPI Target IP on the following Lattice evaluation board:

- MachXO5-NX Development Board

6.1. Example Design Supported Configuration

Table 6.1. xSPI Target IP Configuration Supported by the Example Design

xSPI Target IP GUI Attribute	xSPI Target IP Configuration		
	Legacy SPI	**Standard SPI**	**Quad SPI**
Implementation Preset			
Implementation Preset	Standard SPI (Legacy)	Simple Protocol	Simple Protocol
Clock Configuration			
System Clock Frequency (MHz)	100	100	100
Maximum SPI Clock (MHz)	50	50	50
Bus Interface			
CSR Interface	APB	APB	APB
FIFO Configuration			
Enable FIFOs	Checked	Checked	Checked
FIFO Implementation	EBR	EBR	EBR
FIFO Depth	64	64	64
IO Primitive			
Use IO Primitive	—	—	Checked
SPI Configuration			
Max Data Lanes	x1	x1	x4
Default Dummy Cycles	8	8	8
Tx Empty Value	0xFF	0xFF	0xFF
CS Polarity	Active Low (Default)	Active Low (Default)	Active Low (Default)
Default SPI Mode	Mode 0 (CPOL=0, CPHA=0)	Mode 0 (CPOL=0, CPHA=0)	Mode 0 (CPOL=0, CPHA=0)
Default Bit Order	MSB First	MSB First	MSB First
Programmable Settings			
Programmable CPOL	Checked	Checked	—
Programmable CPHA	Checked	Checked	—
Programmable Bit Order	Checked	Checked	—
Programmable Commands	—	Checked	Checked
Command Configuration			
Default Write Command	—	02	02
Default Read Command	—	03	03
Default Status Command	—	05	05
Reset Configuration			
Enable In-band Reset	—	Checked	Checked
Enable Reset Pin	Unchecked	Checked	Checked

6.2. Overview of the Example Design and Features

Key features of the example design include:

- xSPI target device under test (DUT) instantiated in a complete system with software and SPI interface
- Software code for testing
- Optional SPI controller for sending SPI transactions

6.3. Example Design Components

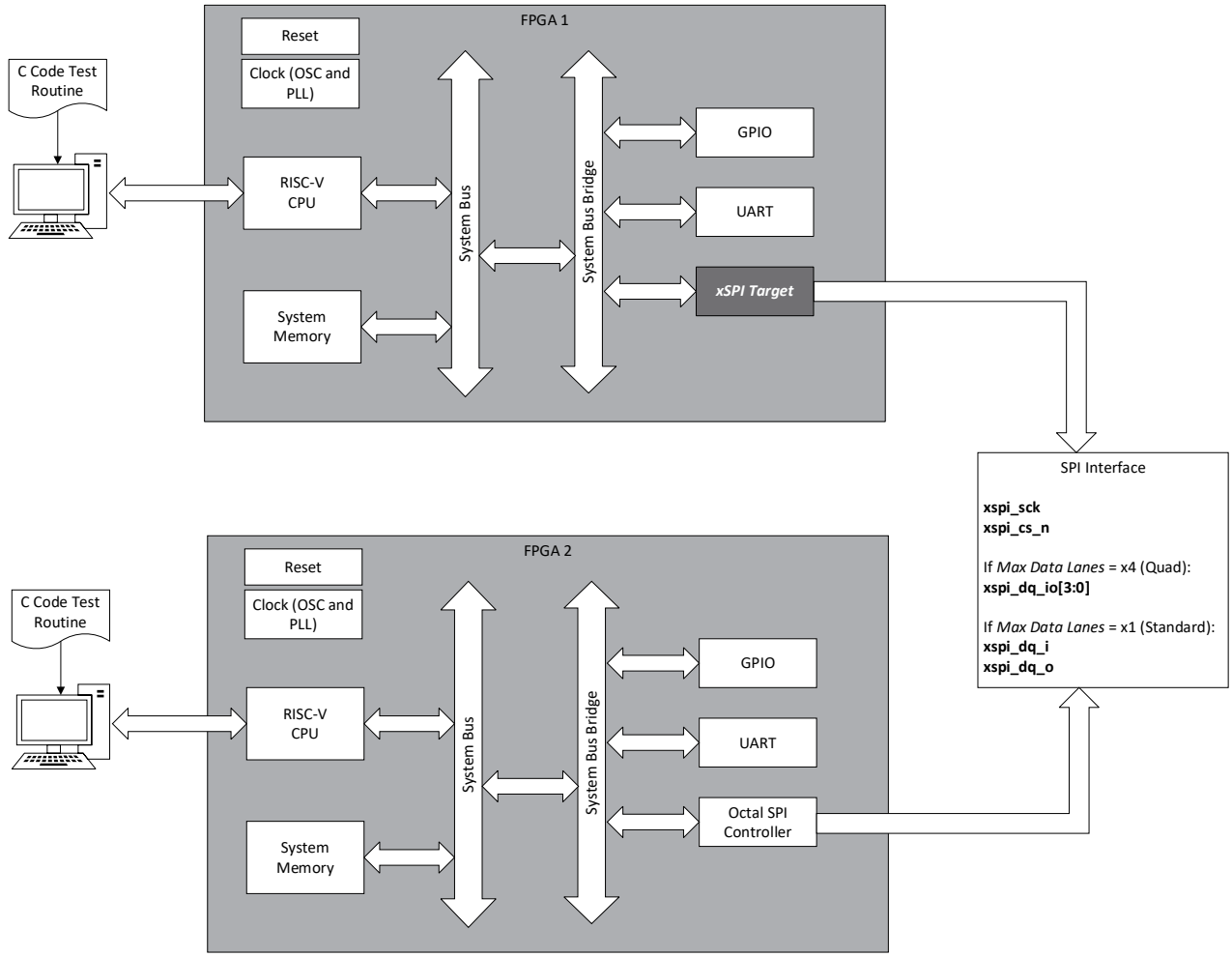

Figure 6.1. xSPI Target Example Design Block Diagram

The xSPI Target example design includes the following blocks:

- xSPI Target (DUT)
- RISC-V CPU and system memory for command processing
- System bus and system bus bridge for interfacing with all IPs in the system
- GPIO for debugging with LEDs, switches, and GPIO headers
- UART for Propel logging
- Reset and clock
- SPI interface for external connections between the controller and target

The optional SPI controller project includes the same blocks as the xSPI target project.

6.4. Generating the Example Design

You can use the Lattice Propel software to generate and use the example design. A sample Lattice Propel software project file for the MachXO5-NX device is provided in the package. By using the sample project, you can run the software implementation flow and hardware tests.

Table 6.2 lists the files generated with the IP required for using the example design.

Table 6.2. Generated File List for Example Design

Attribute	Description
eval/ed/soc/xspi_target_soc	Sample Lattice Propel Builder SoC project containing xSPI Target IP.
eval/ed/sw/main.c	Sample Propel SDK file for hardware testing.
eval/ed/constraints/*.pdc	Sample post-synthesis constraint files in .pdc format for the example design. Pin location constraints are pre-generated for the MachXO5-NX development board only.
eval/ed/ospi_controller_avant	Sample Lattice Propel Builder SoC project containing Octal SPI Controller IP, with sample constraints and main.c.

6.4.1. Using the Example Design Sample Project

The sample project includes all the files required by the example design including the .pdc file. To use the example design sample project, follow these steps:

1. In Propel Builder, open the sample project provided at *eval/ed/soc/xspi_target_ed.sbx*.

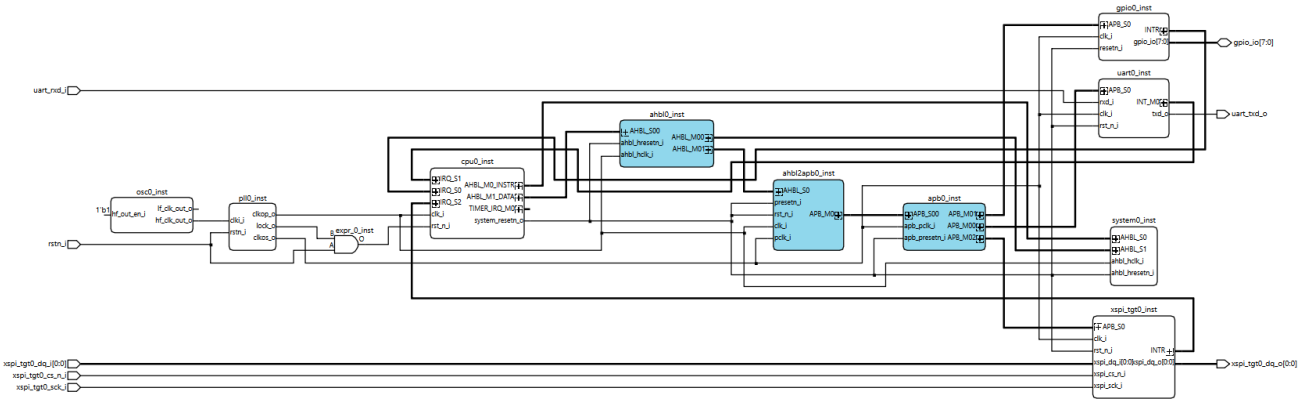

Figure 6.2. Example Design Project

2. Click the **Run Radiant** button located on the toolbar to generate and open the equivalent Lattice Radiant project.
3. Add *eval/ed/constraints/xspi_target_ed_xo5.pdc* to the Radiant project.
4. In the Radiant software, click the **Run All** button located on the toolbar to perform the Radiant software full design compilation, which generates the example design bitstream file for the hardware test.

6.4.2. Changing Configuration of the Example Design

The sample project generated instantiates the standard SPI configuration in Table 6.1. If you want to test a different xSPI Target IP configuration, double-click the IP instance in the Propel Builder project and configure accordingly. Refer to the corresponding sections in *eval/ed/constraints/xspi_target_ed_xo5.pdc* for your IP configuration.

6.4.3. Example Design Limitations

The following are limitations of the example design:

- When changing configuration to x4 I/O width, you must manually connect the SPI data lines to external ports.
- When adding xspi_tgt_rst_n_i, you must manually add this port in the project and assign it to a device pin for connection to external reset.
- The current example design does not incorporate the application programming interfaces (APIs) located in the *driver* folder. The design will be updated to use these APIs in a future release.

6.5. Hardware Testing

Note: This section does not cover the steps for using the Lattice Propel SDK software. For usage details, refer to the Lattice Propel Tool Flow section of the Lattice Propel SDK User Guide.

6.5.1. Hardware Testing Setup

To set up the hardware for testing:

1. Download the generated bitstream from the Generating the Example Design section to the evaluation board using the Lattice Radiant Programmer. When the design is successfully programmed, the 7-segment display lights up.
2. Before running tests, perform or ensure the following:
   - Connect the xSPI target to the external SPI controller device using flywire.
   - Verify that the pin mapping between the DUT and the external SPI controller is correct to avoid communication errors or hardware damage.
   - Keep wire lengths short to maintain signal integrity and minimize noise or crosstalk.
3. In Propel Builder, click the **Generate** button  , then click the **Run Propel** button  located on the toolbar to open the Lattice Propel software.
4. Create your Lattice C/C++ project and copy the contents of *eval/ed/sw/main.c* to *<your_project_path>/src/main.c.*
5. Build your project.
6. Launch a serial terminal with baud rate set to 9600. The test logs will be displayed on this monitor.
7. Start a debug session and execute main.c.
8. Send SPI transactions from SPI Controller.

6.5.2. Expected Output

The following figure shows the expected output on the serial terminal during testing.

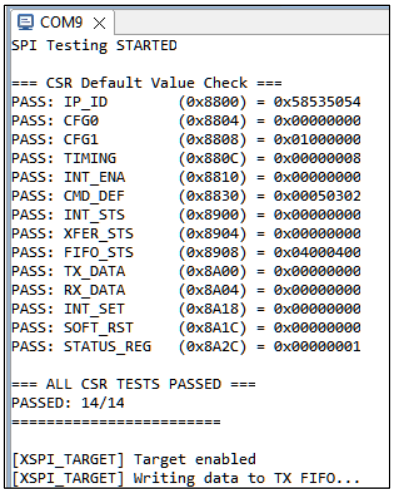

Figure 6.3. Hardware Testing Output

7. Designing with the IP

This section provides information on how to generate the IP core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

Notes:

- The screenshots provided are for reference only. Details may vary depending on the version of the IP or software being used. If there have been no significant changes to the GUI, a screenshot may reflect an earlier version of the IP.
- For MachXO3D devices, which are indirectly supported in the Lattice Diamond software through Propel, software-dependent details may differ.

7.1. Generating and Instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device architecture. The steps below describe how to generate the xSPI Target IP in the Lattice Radiant software.

To generate the xSPI Target IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. Click the **IP Catalog** button to view the **IP Catalog** pane.
3. On the **IP on Local** tab, double-click **xSPI Target** under the **IP/…/Processors_Controllers_and_Peripherals** category. The **Module/IP Block Wizard** opens.
   Note: If the IP is not available on the **IP on Local** tab, download the IP from the **IP on Server** tab.

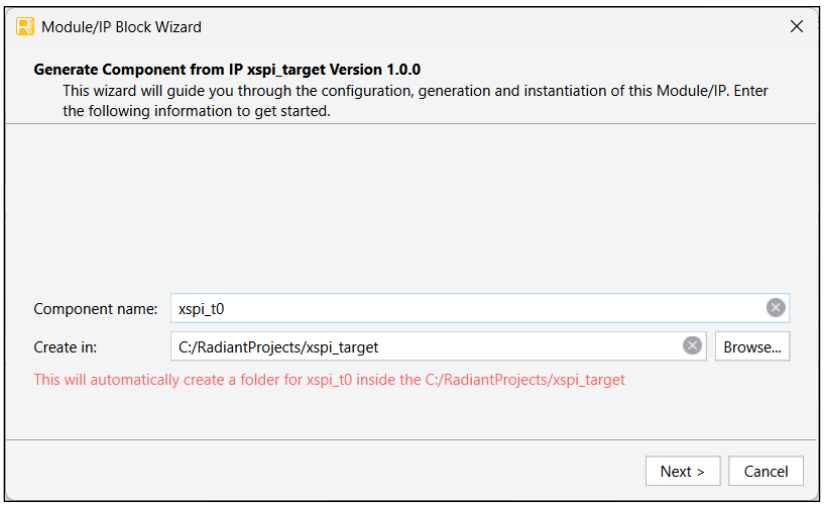

Figure 7.1. Module/IP Block Wizard

4. Enter values in the **Component name** and **Create in** fields, then click **Next**.
5. Customize the selected xSPI Target IP using drop-down lists and check boxes. Figure 7.2 shows an example configuration of the xSPI Target IP. For details on the configuration options, refer to the IP Parameter Description section.

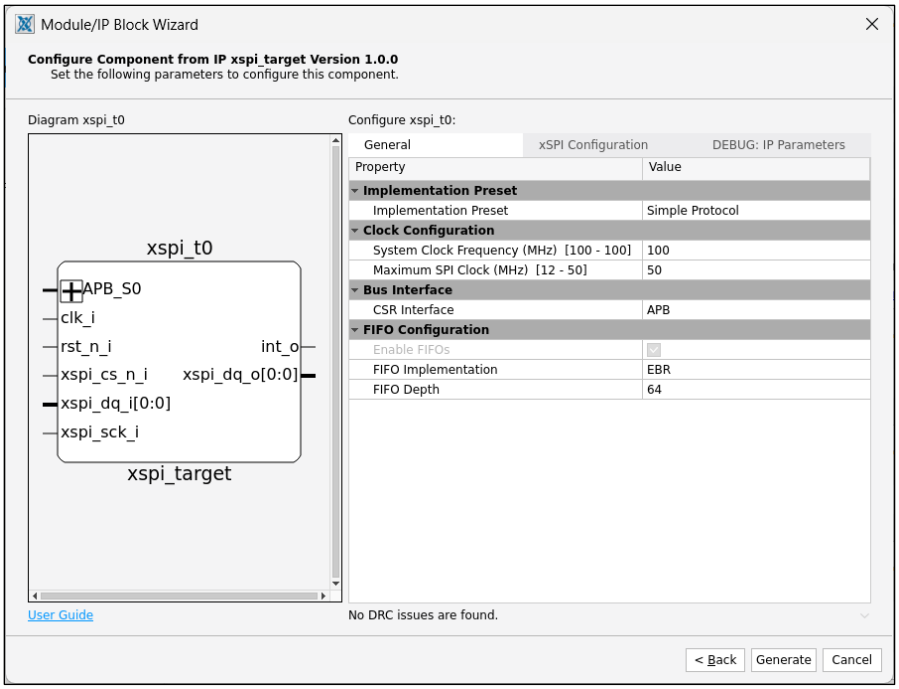

Figure 7.2. IP Configuration

6. Click **Generate**. The **Check Generated Result** window opens. This window shows design block messages and results.

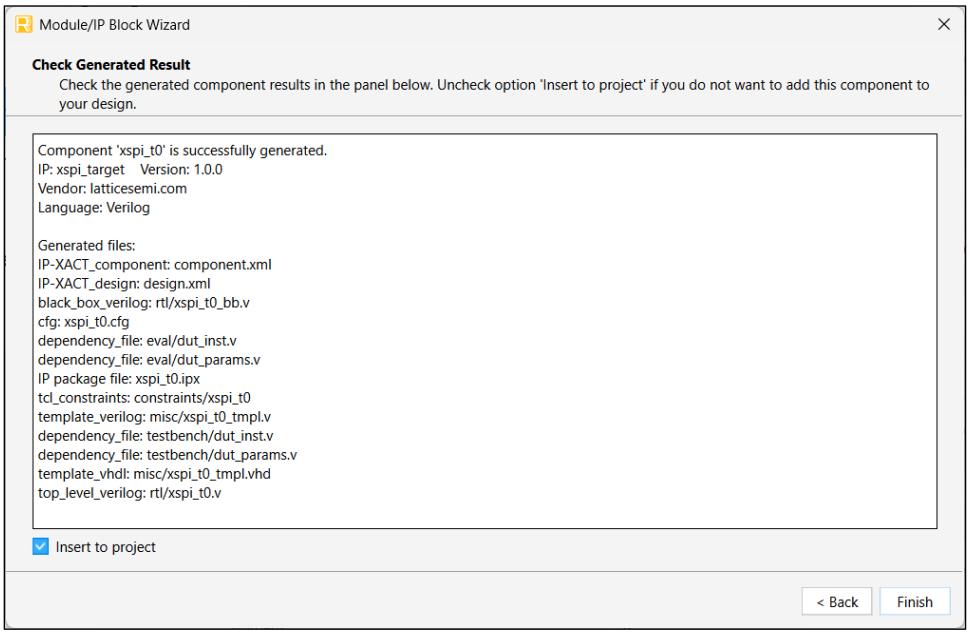

Figure 7.3. Check Generated Result

7. Click **Finish**. All generated files are placed in the directory specified by the **Component name** and **Create in** fields shown in Figure 7.1.

7.1.1. Generated Files and File Structure

The generated xSPI Target module package includes the closed-box (*<Component name>_bb.v*) and instance templates (*<Component name>_tmpl.v/vhd*) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (*<Component name>.v*) that can be used as an instantiation template for the module is also provided. You may also use this example as the starting template for your top-level design. The generated files are listed in Table 7.1.

Table 7.1. Generated File List

Attribute	Description
<Component name>.ipx	Contains the information on the files associated with the generated IP.
<Component name>.cfg	Contains the parameter values used in IP configuration.
component.xml	Contains the ipxact: component information of the IP.
design.xml	Documents the configuration parameters of the IP in the IP-XACT 2014 format.
rtl/<Component name>.v	Provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	Provides the synthesis closed-box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	Provide instance templates for the module.

7.2. Design Implementation

Completing your design includes additional steps to specify analog properties, pin assignments, and timing and physical constraints. You can add and edit the constraints using the Device Constraint Editor or by manually creating a .pdc file.

Post-synthesis constraint files (.pdc) contain both timing and non-timing *constraint.pdc* source files for storing logical timing/physical constraints. Constraints that are added using the Device Constraint Editor are saved to the active .pdc file. The active post-synthesis design constraint file is then used as input for post-synthesis processes.

Refer to the relevant sections in the Lattice Radiant Software User Guide for more information on how to create or edit constraints and how to use the Device Constraint Editor.

7.3. Timing Constraints

The timing constraints are based on the clock frequency used. Provide proper timing and physical design constraints to ensure the design meets the desired performance goals on the FPGA.

A sample constraint file is generated with the IP and available at *<IP_Instance_Path>/<IP_Instance_Name>/eval/constraint.pdc*. This constraint file has been verified during IP evaluation with the IP instantiated directly in the top-level module. You should only modify the constraints in this file with a thorough understanding of the effect of each constraint. To use this constraint file, copy the contents of constraint.pdc to the top-level design constraint for post-synthesis.

For more information on timing constraints, refer to the Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059).

7.4. Running Functional Simulation

You can run functional simulation after the IP is generated.

To run functional simulation:

1. Click the **Simulation Wizard** button  located on the toolbar to initiate the **Simulation Wizard** shown in Figure 7.4.

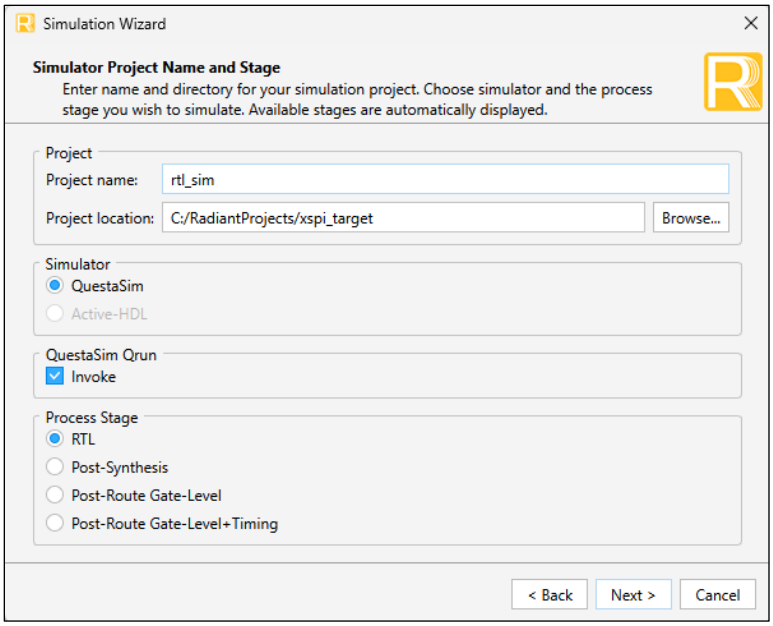

Figure 7.4. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 7.5.

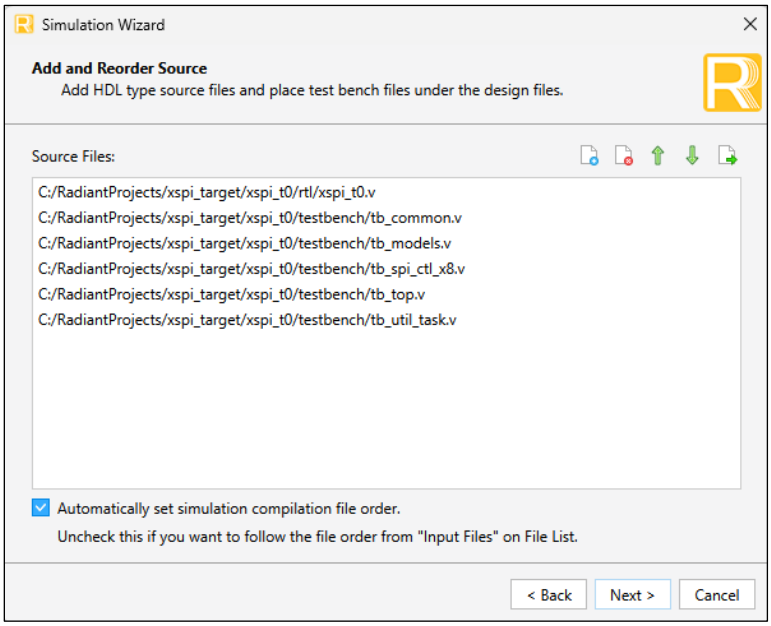

Figure 7.5. Add and Reorder Source

3. Click **Next**. The **Summary** window is shown.

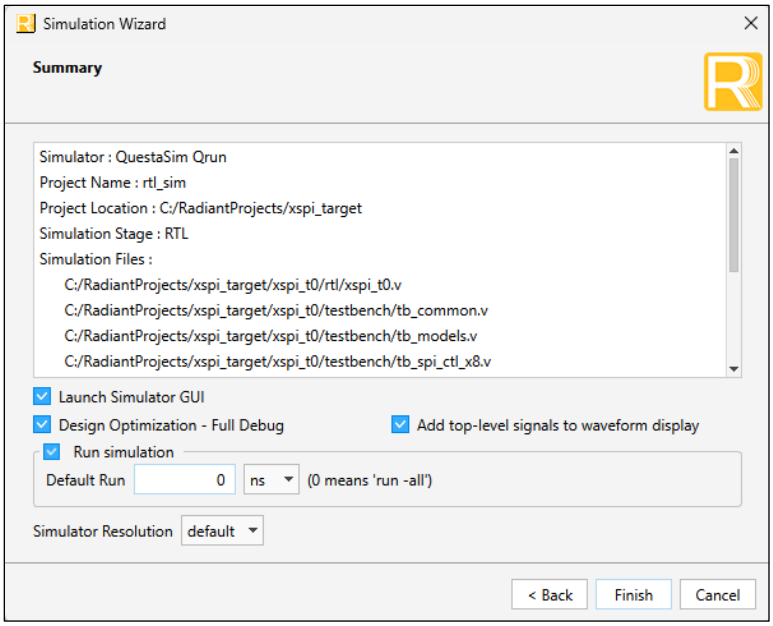

Figure 7.6. Simulation Wizard Summary

4. Set **Default Run** to **0 ns**.
5. Click **Finish** to run the simulation.

The waveform in Figure 7.7 shows an example simulation result.

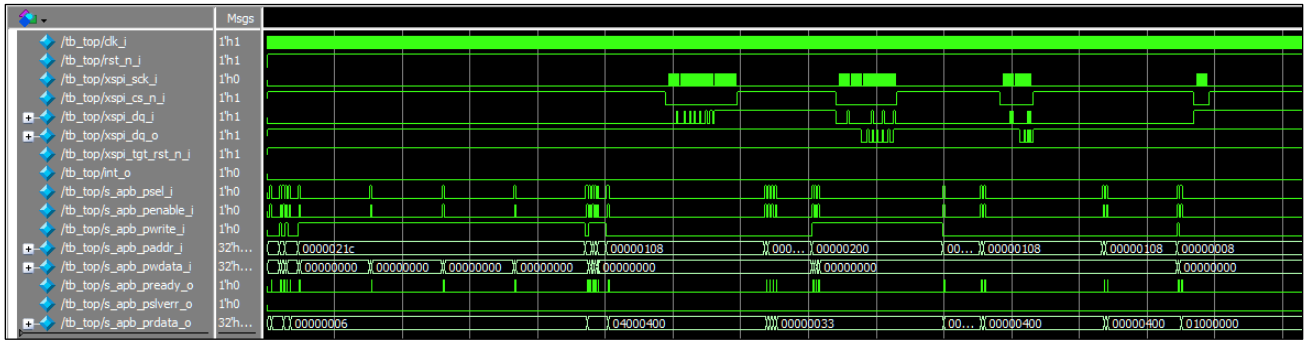

Figure 7.7. Simulation Waveform

7.4.1. Simulation Results

Figure 7.8 shows an example log after the simulation is done.

```
# [     8155000]:[Info   ]---     dummy_data: 1 (0 cycles)---[tb_top.spi_read_with_command]
# [     8155000]:[Info   ]---     NUM1SCK:    0 (0 cycles)---[tb_top.spi_read_with_command]
# [     8155000]:[Info   ]---   Lane Width:   x1       ---[tb_top.spi_read_with_command]
# [     8155000]:[Info   ]---   Rate:         STR---[tb_top.spi_read_with_command]
# [     8155000]:[Info   ]--- ----------------------------------------[tb_top.spi_read_with_command]
# [     9465000]:[Info   ]--- SPI Read Complete---[tb_top.spi_read_with_command]
# [    10015000]:[Info   ]---   RX Byte[0]: 0x11 (exp: 0x11)    OK---[tb_top.test_simple_read_command]
# [    10015000]:[Info   ]---   RX Byte[1]: 0x22 (exp: 0x22)    OK---[tb_top.test_simple_read_command]
# [    10015000]:[Info   ]---   RX Byte[2]: 0x33 (exp: 0x33)    OK---[tb_top.test_simple_read_command]
# [    10015000]:[Info   ]--- SIMPLE Mode READ Command PASSED!---[tb_top.test_simple_read_command]
# [    10515000]:[Info   ]--- ----------------------------------------[tb_top]
# [    10515000]:[Info   ]--- TEST 3: X1 SIMPLE Mode - STATUS Command---[tb_top]
# [    10515000]:[Info   ]--- ----------------------------------------[tb_top]
# [    10515000]:[Info   ]--- Sending STATUS command (0x05)---[tb_top.test_simple_status_command]
# [    10565000]:[Info   ]--- Configured STATUS register value: 0x15---[tb_top.test_simple_status_command]
# [    10615000]:[Info   ]--- Sending STATUS command: 0x05 (guard=0)---[tb_top.spi_status_command]
# [    11825000]:[Info   ]--- STATUS command response: 0x15---[tb_top.spi_status_command]
# [    12375000]:[Info   ]--- Status response matches configured value: 0x15---[tb_top.test_simple_status_command]
# [    12425000]:[Info   ]--- TX FIFO status unchanged (correct - STATUS doesn't use FIFO)---[tb_top.test_simple_status_command]
# [    12425000]:[Info   ]--- SIMPLE Mode STATUS Command PASSED!---[tb_top.test_simple_status_command]
# [    13425000]:[Info   ]--- ----------------------------------------[tb_top]
# [    13425000]:[Info   ]--- TEST 4: X1 SIMPLE Mode - In-band RESET Command---[tb_top]
# [    13425000]:[Info   ]--- ----------------------------------------[tb_top]
# [    13515000]:[Info   ]--- CSR Check PASSED: Addr=0x008, Data=0x01000000---[tb_top.csr_check]
# [    13515000]:[Info   ]--- Sending STATUS command: 0xff (guard=0)---[tb_top.spi_inband_reset]
# [    20635000]:Ending Simulation...
# ************************
# ***Message Counters*****
#      Info    : 86
#      Warning: 0
#      Debug   : 0
#      Error   : 0
# ************************
# ***SIMULATION PASSED***
# ** Note: $stop    : C:/RadiantProjects/xspi_target/xspi_t0/testbench/tb_top.v(404)
#    Time: 20635 ns  Iteration: 1  Instance: /tb_top
```

Figure 7.8. Simulation Results Transcript

8. Debugging

This section describes methods for identifying and resolving issues when using the xSPI Target IP.

8.1. Debug Methods

8.1.1. Internal Register Read for Debug

The following registers can be monitored to diagnose SPI communication issues:

- XFER_STATUS.ERR_CODE – Indicates whether an error occurred during the most recent SPI transaction. An error is reported when the received SPI command does not match the configuration defined in XSPI_TGT_CMD_DEF.
- FIFO_STATUS and INT_STATUS – Provide visibility into FIFO conditions and help detect data integrity issues. These registers can be used to observe the following conditions:
  On the SPI interface:
  - Loss of received SPI write data due to Rx FIFO overflow
  - Invalid transmitted SPI read data due to Tx FIFO underflow

  On the user interface:
  - Write attempts to Tx FIFO rejected due to Tx FIFO full condition
  - Read attempts from Rx FIFO returning no valid data due to Rx FIFO empty condition

If FIFO overflow or underflow conditions are observed, consider the following:

- Reduce SPI clock frequency
  Lowering the SPI clock allows more margin for the system to service Tx and Rx FIFOs.
- Increase FIFO depth
  A larger FIFO can help absorb burst traffic and reduce the likelihood of overflow or underflow.

8.1.2. Signal Capture

Use an on-chip analyzer or an external logic analyzer to monitor SPI interface signals during operation. This method is useful for verifying the timing relationship between data and clock signals.

8.2. Debug Tools

The following tools can be used to debug xSPI Target IP design issues.

8.2.1. Reveal Analyzer

The Reveal™ Analyzer monitors internal FPGA signals based on user-defined trigger conditions, ranging from simple to complex events. When a trigger condition is met, the analyzer captures signal activity before, during, and after the event and presents it as a waveform.

Captured data can be exported in the following formats:

- Value change dump file (.vcd) – Compatible with tools such as QuestaSim™.
- ASCII tabular format – Usable in tools such as Microsoft® Excel.

Before running the Reveal Analyzer, use the Reveal Inserter to add Reveal modules to your design. In these modules, specify the signals to monitor, define the trigger conditions, and set other capture options. The Reveal Analyzer supports multiple logic analyzer cores using hard or soft JTAG interface. You can have up to 15 modules, typically one for each clock region of interest. When the modules are set up, regenerate the bitstream and program the FPGA.

Refer to the Reveal User Guide for Radiant Software for details on how to use the Reveal Analyzer.

Appendix A. Resource Utilization

Note: Resource utilization values in this section are provided for reference only and may change based on the compilation strategy and selected tool options.

Table A.1 shows a sample resource utilization of the xSPI Target IP core on the LFMXO5-25-7BBG400I device using Synplify Pro of the Lattice Radiant Software 2026.1. The default configuration is used. Some attributes are changed from the default value to show the effect on resource utilization.

Table A.1. Resource Utilization on LFMXO5-25-7BBG400I Device

IP Configuration	f_{MAX} (MHz)[1]	LUTs	Registers	EBRs
Default	163	521	448	2
Implementation Preset = Standard SPI (Legacy) *Enable FIFOs* = Unchecked	200	182	225	0
Implementation Preset = Standard SPI (Legacy) *FIFO Depth* = 512 *Enable Reset Pin* = Checked	144	601	472	2
Implementation Preset = Simple Protocol *FIFO Depth* = 16 *Max Data Lanes* = x4 (Quad) *Enable In-band Reset* = Unchecked	166	500	404	2
Implementation Preset = Simple Protocol *FIFO Depth* = 512 *Max Data Lanes* = x4 (Quad) *Enable Reset Pin* = Checked	137	799	557	2

Note:

1. f_{MAX} is generated when the FPGA design contains only the IP. These values may be reduced when user logic is added to the FPGA design.

Table A.2 shows a sample resource utilization of the xSPI Target IP core on the LFCPNX-100-7LFG672I device using Synplify Pro of the Lattice Radiant Software 2026.1. The default configuration is used. Some attributes are changed from the default value to show the effect on resource utilization.

Table A.2. Resource Utilization on LFCPNX-100-7LFG672I Device

IP Configuration	f_{MAX} (MHz)[1]	LUTs	Registers	EBRs
Default	174	521	448	2
Implementation Preset = Standard SPI (Legacy) *Enable FIFOs* = Unchecked	200	184	225	0
Implementation Preset = Standard SPI (Legacy) *FIFO Depth* = 512 *Enable Reset Pin* = Checked	134	596	472	2
Implementation Preset = Simple Protocol *FIFO Depth* = 16 *Max Data Lanes* = x4 (Quad) *Enable In-band Reset* = Unchecked	143	500	404	2
Implementation Preset = Simple Protocol *FIFO Depth* = 512 *Max Data Lanes* = x4 (Quad) *Enable Reset Pin* = Checked	129	799	557	2

Note:

1. f_{MAX} is generated when the FPGA design contains only the IP. These values may be reduced when user logic is added to the FPGA design.

Table A.3 shows a sample resource utilization of the xSPI Target IP core on the LFD2NX-15-7BG400I device using Synplify Pro of the Lattice Radiant Software 2026.1. The default configuration is used. Some attributes are changed from the default value to show the effect on resource utilization.

Table A.3. Resource Utilization on LFD2NX-15-7BG400I Device

IP Configuration	f_{MAX} (MHz)[1]	LUTs	Registers	EBRs
Default	168	521	448	2
Implementation Preset = Standard SPI (Legacy) *Enable FIFOs* = Unchecked	200	182	225	0
Implementation Preset = Standard SPI (Legacy) *FIFO Depth* = 512 *Enable Reset Pin* = Checked	130	596	472	2
Implementation Preset = Simple Protocol *FIFO Depth* = 16 *Max Data Lanes* = x4 (Quad) *Enable In-band Reset* = Unchecked	146	500	404	2
Implementation Preset = Simple Protocol *FIFO Depth* = 512 *Max Data Lanes* = x4 (Quad) *Enable Reset Pin* = Checked	137	799	557	2

Note:

1. f_{MAX} is generated when the FPGA design contains only the IP. These values may be reduced when user logic is added to the FPGA design.

Table A.4 shows a sample resource utilization of the xSPI Target IP core on the LIFCL-40-7BG400I device using Synplify Pro of the Lattice Radiant Software 2026.1. The default configuration is used. Some attributes are changed from the default value to show the effect on resource utilization.

Table A.4. Resource Utilization on LIFCL-40-7BG400I Device

IP Configuration	f_{MAX} (MHz)[1]	LUTs	Registers	EBRs
Default	173	521	448	2
Implementation Preset = Standard SPI (Legacy) *Enable FIFOs* = Unchecked	200	182	225	0
Implementation Preset = Standard SPI (Legacy) *FIFO Depth* = 512 *Enable Reset Pin* = Checked	132	601	472	2
Implementation Preset = Simple Protocol *FIFO Depth* = 16 *Max Data Lanes* = x4 (Quad) *Enable In-band Reset* = Unchecked	137	500	404	2
Implementation Preset = Simple Protocol *FIFO Depth* = 512 *Max Data Lanes* = x4 (Quad) *Enable Reset Pin* = Checked	131	799	557	2

Note:

1. f_{MAX} is generated when the FPGA design contains only the IP. These values may be reduced when user logic is added to the FPGA design.

Table A.5 shows a sample resource utilization of the xSPI Target IP core on the LCMXO3D-9400HC-5BG400I device using Synplify Pro of the Lattice Diamond Software 3.14. The default configuration is used. Some attributes are changed from the default value to show the effect on resource utilization.

Table A.5. Resource Utilization on LCMXO3D-9400HC-5BG400I Device

IP Configuration	f_{MAX} (MHz)[1]	LUTs	Registers	EBRs
Default	86	352	305	2
Implementation Preset = Standard SPI (Legacy) *Enable FIFOs* = Unchecked	113	182	225	2
Implementation Preset = Standard SPI (Legacy) *FIFO Depth* = 512 *Enable Reset Pin* = Checked	96	289	275	2
Implementation Preset = Simple Protocol *FIFO Depth* = 16 *Max Data Lanes* = x4 (Quad) *Enable In-band Reset* = Unchecked	94	403	296	2
Implementation Preset = Simple Protocol *FIFO Depth* = 512 *Max Data Lanes* = x4 (Quad) *Enable Reset Pin* = Checked	97	388	296	2

Note:

1. f_{MAX} is generated when the FPGA design contains only the IP. These values may be reduced when user logic is added to the FPGA design.

References

- xSPI Target IP Release Notes (FPGA-RN-02115)
- xSPI Target Driver API Reference (FPGA-TN-02426)
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide (FPGA-UG-02039)
- AMBA 3 APB Protocol v1.0 Specification
- AMBA 3 AHB-Lite Protocol v1.0 Specification
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- MachXO5-NX web page
- MachXO3D web page
- MachXO5-NX Development Board web page
- Lattice Radiant Software web page
- Lattice Propel Design Environment web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Lattice Insights for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Note: In some instances, the IP may be updated without changes to the user guide. The user guide may reflect an earlier IP version but remains fully compatible with the later IP version. Refer to the IP Release Notes for the latest updates.

Revision 1.0, June 2026

Section	Change Summary
All	Initial release.

www.latticesemi.com