Octal SPI Controller IP

IP Version: v1.4.0

User Guide

FPGA-IPUG-02273-1.4

December 2025

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
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
AXI4	Advanced eXtensible Interface 4
CPHA	SPI Clock Phase
CPOL	SPI Clock Polarity
CPU	Central Processing Unit
CS	Chip Select
CSR	Configuration Status Register
DRC	Design Rule Checking
DSPI	Dual Serial Peripheral Interface
DTR	Double Transfer Rate
DW	Double Word
EBR	Embedded Block RAM
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GPIO	General Purpose I/O
GUI	Graphical User Interface
HDL	Hardware Description Language
IP	Intellectual Property
JEDEC	Joint Electron Device Engineering Council
KiB	Kibibyte (1024 bytes)
LSB	Least Significant Bit
MiB	Mebibyte (1024 kilobytes)
MSB	Most Significant Bit
OSC	Oscillator
PLL	Phase-Locked Loop
PVT	Process, Voltage, Temperature
QSPI	Quad Serial Peripheral Interface
SCK	SPI Clock
SoC	System-on-Chip
SPI	Serial Peripheral Interface
STR	Single Transfer Rate
UART	Universal Asynchronous Receiver Transmitter
XiP	Execute-in-Place
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. This document describes the Octal SPI Controller IP.

1.1. Overview of the IP

The Lattice Octal SPI Controller IP is an SPI interface that supports different types of SPI protocols: standard, dual, quad, and xSPI. Standard SPI is the legacy four-wire interface with separate data lines for input and output. Dual SPI (DSPI) is an extension of standard SPI using two data lines. Quad SPI (QSPI) is another extension of standard SPI designed to provide a higher data transfer rate using four data lines. Expanded SPI (xSPI) is an advanced extension of standard SPI designed to provide higher data throughput and improved performance while maintaining backward compatibility with standard SPI devices. xSPI uses eight data lines and is capable of double data rate transfer.

1.2. Quick Facts

Table 1.1. Summary of the Octal SPI Controller IP

IP Requirements	Supported Devices	CrossLink™-NX, Certus™-NX (LFD2NX-17, LFD2NX-40), CertusPro™-NX, MachXO5™-NX (LFMXO5-25, LFMXO5-55T, LFMXO5-100T), Lattice Avant™ (LAV-AT-E70, LAV-AT-G70, LAV-AT-X70)
	IP Changes[1]	Refer to the Octal SPI Controller IP Release Notes (FPGA-RN-02011).
Resource Utilization	Supported User Interface	Advanced eXtensible Interface 4 (AXI4), AXI4-Lite, Advanced Peripheral Bus (APB)
	Resources	Refer to Appendix A. Resource Utilization.
Design Tool Support	Lattice Implementation	IP Core v1.4.0 – Lattice Radiant™ Software 2025.2
	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 Octal SPI Controller Driver API Reference (FPGA-TN-02400).

Note:

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.

1.3. IP Support Summary

Table 1.2. Octal SPI Controller IP Support Readiness

Device Family	I/O Width	SCK Frequency (MHz)	Radiant Timing Model	Hardware Validated
Avant	x1, x8	200[1]	Preliminary	Yes
	x1, x4	100[2]	Preliminary	Yes
	x2	100	—	No
CertusPro-NX	x1, x4	50[2, 3]	Preliminary	Yes
	x2, x8	50[3]	—	No
Certus-NX	x1, x2, x4, x8	50[3]	—	No
CrossLink-NX	x1, x2, x4, x8	50[3]	—	No
MachXO5-NX	x1, x2, x4, x8	50[3]	—	No

Notes:

2. Hardware validated with UMXSPI primitive (*JEDEC xSPI* selected) and double transfer rate (DTR).
3. Hardware validated with regular I/O primitive (no UMXSPI) and single transfer rate (STR) only.
4. Nexus devices can support 50 MHz in DTR mode and 100 MHz in STR mode. The performance of a Nexus device is limited by I/O (higher roundtrip delay) and the absence of a UMXSPI primitive to handle high-speed I/O.

1.4. Features

Key features of the Octal SPI Controller IP include:

- JEDEC JESD251C standard xSPI (x8, double transfer rate) support
- SPI (x1), dual SPI (x2), and quad SPI (x4) support
- AMBA AXI4 protocol bus interface
  - 32-bit data and 32-bit address
  - Single and burst transfers
- Dual bus interface option for separate configuration status register (CSR) and target data path interfaces
- SPI clock frequencies up to 200 MHz
- Double transfer rates up to 400 MT/s with 200 MHz clock
- Generic SPI controller function
- Flash command processing
- Up to 32 (target) chip select lines
- Execute-in-place (XiP) mode
- Optional target reset pin
- Configurable clock phase and polarity
- FIFO up to 512 depth

1.5. Licensing and Ordering Information

The Octal SPI Controller IP is provided at no additional cost with the Lattice Radiant software.

1.6. Hardware Support

Refer to the Example Design section for more information on the boards used.

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

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

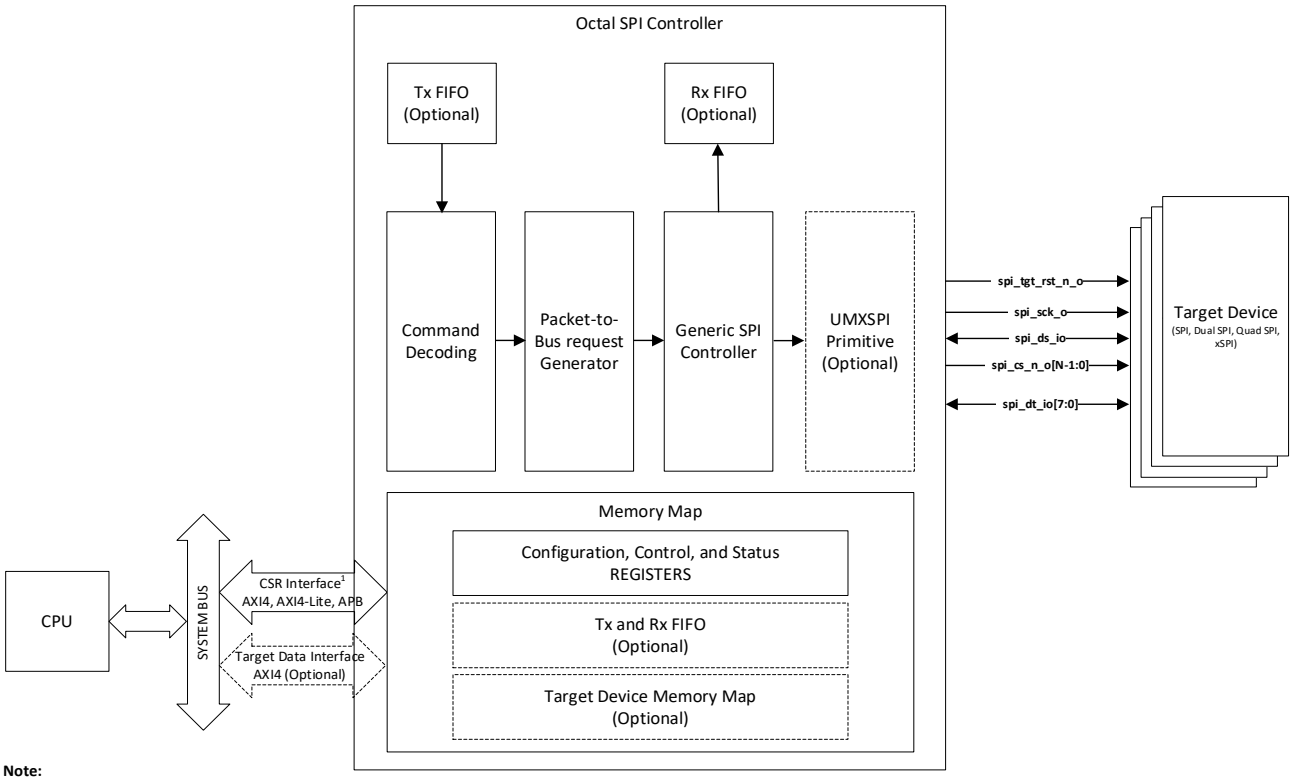
The Octal SPI Controller IP core supports the JEDEC standard xSPI and can also be configured to support standard SPI, dual SPI, and quad SPI. Double transfer rate (DTR) is supported in xSPI and can run up to 400 MT/s (per I/O) at a clock frequency of 200 MHz (using the UMXSPI primitive). The UMXSPI primitive handles the I/O logic and converts single transfer rate (STR) to DTR. It also includes several delay primitives that can be used to adjust the timing of I/O signals (clock, data, and data strobe).

The IP can be configured to enable the soft logic for processing of flash commands. XiP mode is also available for applications that require the target (flash device) to be mapped as if it is local memory.

You may send the SPI transactions by:

- Writing to the registers.
- Enabling the FIFOs for burst transfer and command queuing.

Figure 2.1 shows the IP functional diagram.



Note:

1. In single bus mode, the CSR interface provides access to the CSR, indirect access to the target, and direct target data path access.

Figure 2.1. Octal SPI Controller IP Core Functional Diagram

2.2. Clocking

The Octal SPI Controller IP has a single input clock operating at up to 200 MHz. The IP generates the SPI clock using the internal clock divider. The SPI clock frequency depends on the divider setting (sck_rate) and can be calculated as follows:

SPI Clock Frequency = Input Clock Frequency, when sck_rate = 0
SPI Clock Frequency = Input Clock Frequency / (2 × sck_rate), when sck_rate = 1, 2, 3, …

For example, if the input clock is 200 MHz and the divider setting is 1, then the SPI clock frequency is 100 MHz. If the divider setting is 0, then the SPI clock frequency is the same as the input clock frequency.

Refer to the General and Octal SPI Configuration 0 Register sections for more information.

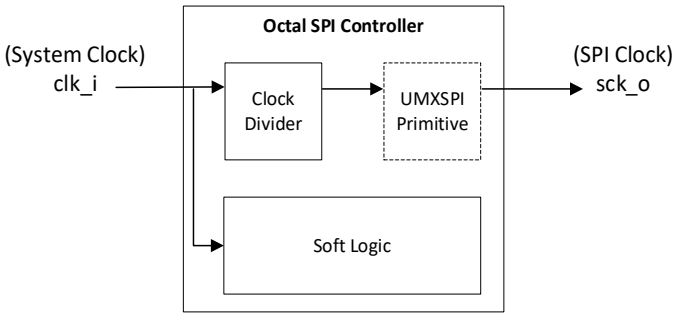


Figure 2.2. Octal SPI Controller IP Clock Domain Block Diagram

2.3. Reset

The Octal SPI Controller IP provides an 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. The reset goes through an internal reset synchronizer and may take several clock cycles to propagate. After de-asserting the reset, wait for at least four clock cycles before starting any transaction.

Besides the reset port, five programmable soft reset controls are provided (refer to the Soft Reset Register section):

- IP Core Reset – Resets the Octal SPI Controller logic except the CSR block.
- IP CSR Reset – Resets the CSR block only.
- Tx FIFO Reset – Resets the Tx FIFO only.
- Rx FIFO Reset – Resets the Rx FIFO only.
- SPI Target Reset – Resets the target device through the output port spi_tgt_rst_n_o.

2.4. User Interfaces

The Octal SPI Controller IP offers the flexibility of using either a single bus or dual bus interface. In single bus mode, you can choose between the AXI4, AXI4-Lite, and APB buses. This single bus provides access to the IP register space (CSR) and indirect access to the target. The same bus also provides direct access to the target memory space when target address mapping is enabled.

When target address mapping is enabled, the dual bus option also becomes available. In dual bus mode, one bus provides direct access (data path) to the target memory space while the other bus provides access to the CSR and indirect access to the target.

Note: In the Signal Description section, AXI4 and AXI4-Lite signals use the same signal names (s_axi4_<port_name>) in single bus mode. In dual bus mode, AXI4 and AXI4-Lite signals for the bus providing access to the CSR are denoted with _c (s_axi4_c_<port_name>).

Table 2.1. User Interfaces and Supported Protocols

User Interface	Supported Protocols	Description
CSR Interface	AXI4, AXI4-Lite, APB	The CSR interface is used to configure the IP core as well as send indirect transactions to the SPI target device using either the generic SPI command packet or supported flash command packet. In single bus mode, this interface provides access to the CSR, indirect access to the target, and direct target data path access. In dual bus mode, this interface provides access to the CSR and indirect access to the target. When *FIFO Interface Mapping* is set to **Map to Data Interface** in the IP GUI, the FIFO is not accessible through the CSR interface.
Target Data Interface (Optional)	AXI4	The target data interface is an optional dedicated interface for the target data path. When target address mapping is enabled (*Enable Target Address Map* is checked in the IP GUI) and dual bus mode is selected, the target device can be directly accessed (using AXI4 write/read transaction) at the allocated address space. An application example is when booting directly from the target flash device with XiP mode enabled. Note that the CSR interface is used for indirect access to the target. However, in dual bus mode, when the *FIFO Interface Mapping* is set to **Map to Data Interface** in the IP GUI, the FIFO is accessible through the target data interface.

2.4.1. Address Space Mapping

The base address of the register block and target device can be set anywhere in the 32-bit address space but must be aligned to the size of allocation. The size of a register block is 1 KiB so the register base address should be aligned to 1 KiB. The target memory map size is configurable. The target base address must be aligned to this size.

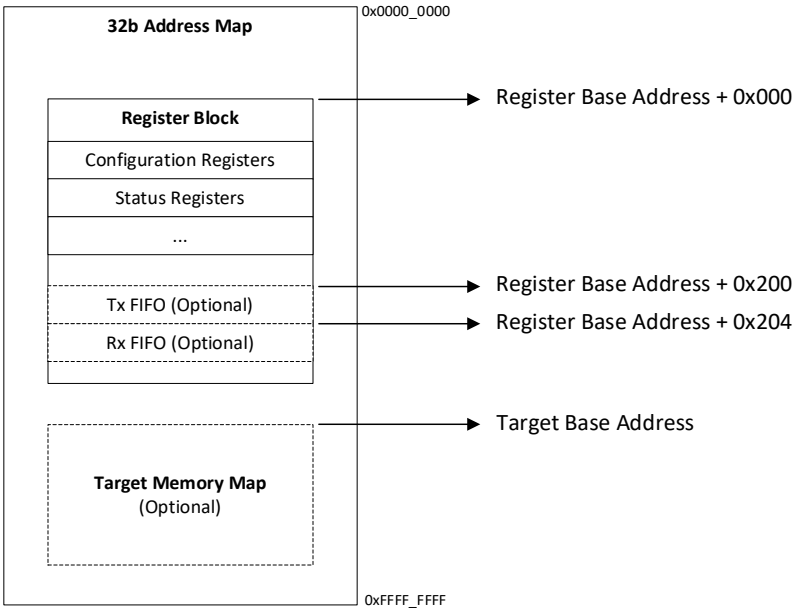


Figure 2.3. Octal SPI Controller Address Space Mapping

2.4.2. Data Byte Order

The data byte order in the bus can be configured by selecting the data endianness. In little endian, the lowest significant byte is stored in the lowest address. In big endian, the most significant byte is stored in the lowest address. Table 2.2 shows an example of AXI4 Data[31:0] = 0x1234_5678 stored in the respective addresses based on data endianness (AXI4 Address[31:0] = 0x0000_1000).

Table 2.2. Example of Data Byte Order by Data Endianness

Byte Address	Data Stored	
	Little Endian	Big Endian
0x0000_1000	0x78	0x12
0x0000_1001	0x56	0x34
0x0000_1002	0x34	0x56
0x0000_1003	0x12	0x78

2.5. User Packets

This section describes the user packets used by the IP to generate SPI transactions. These user packets are translated by the Octal SPI Controller IP into SPI serial data. A user packet is written to the Tx FIFO through the AXI4 interface. Read data response from the target device is stored in the Rx FIFO and can be read though the AXI4 interface.

2.5.1. Packet Frame Structure

There are four packet frame structures or formats that can be used depending on the command and information needed to perform the SPI transaction. The supported flash command packet formats are applicable only if *Enable Command Processing* is selected in the IP GUI.

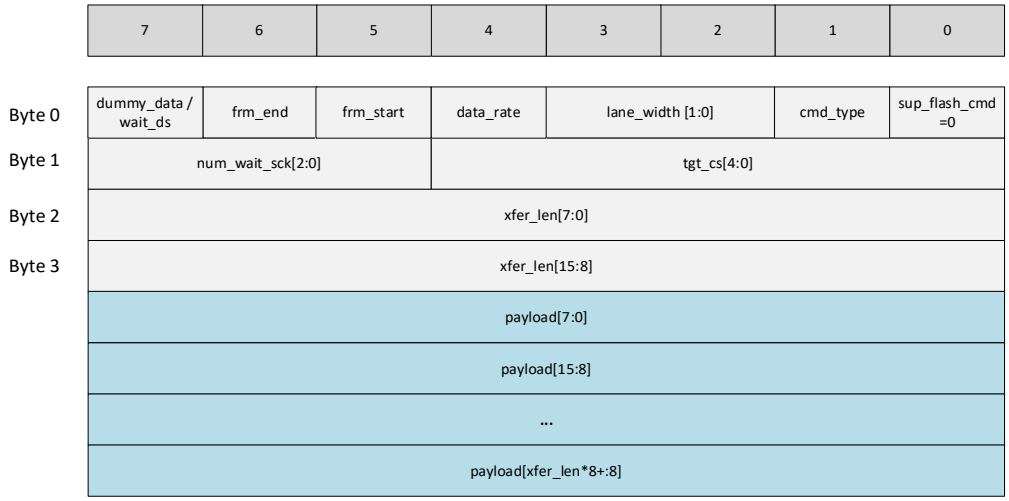


Figure 2.4. Generic SPI Command Packet

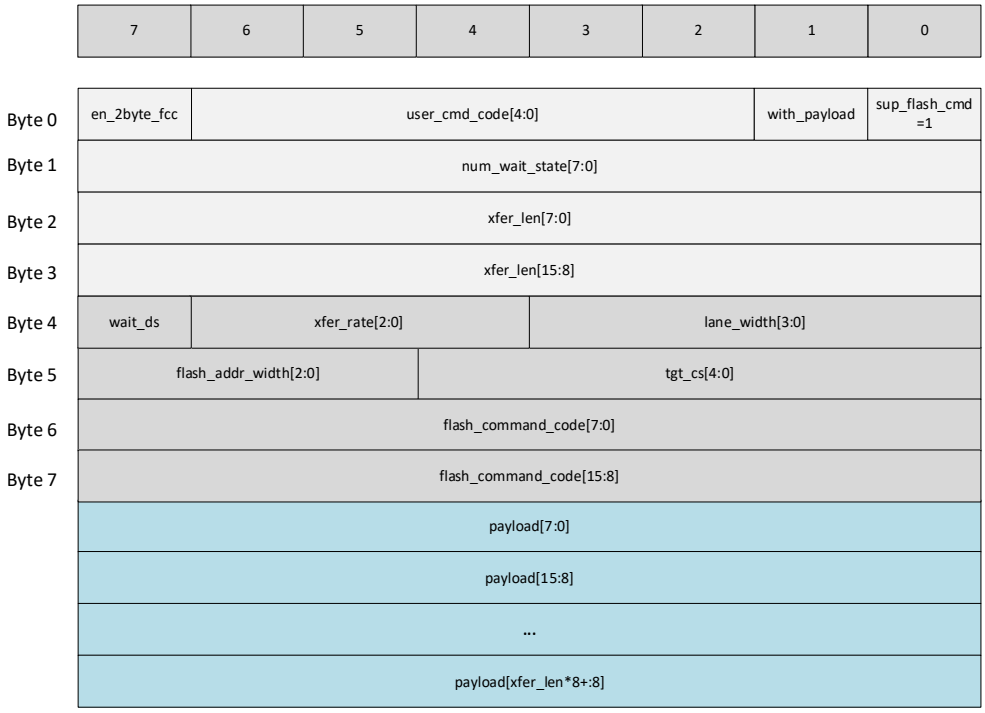


Figure 2.5. Supported Flash Command Packet with 2DW Header

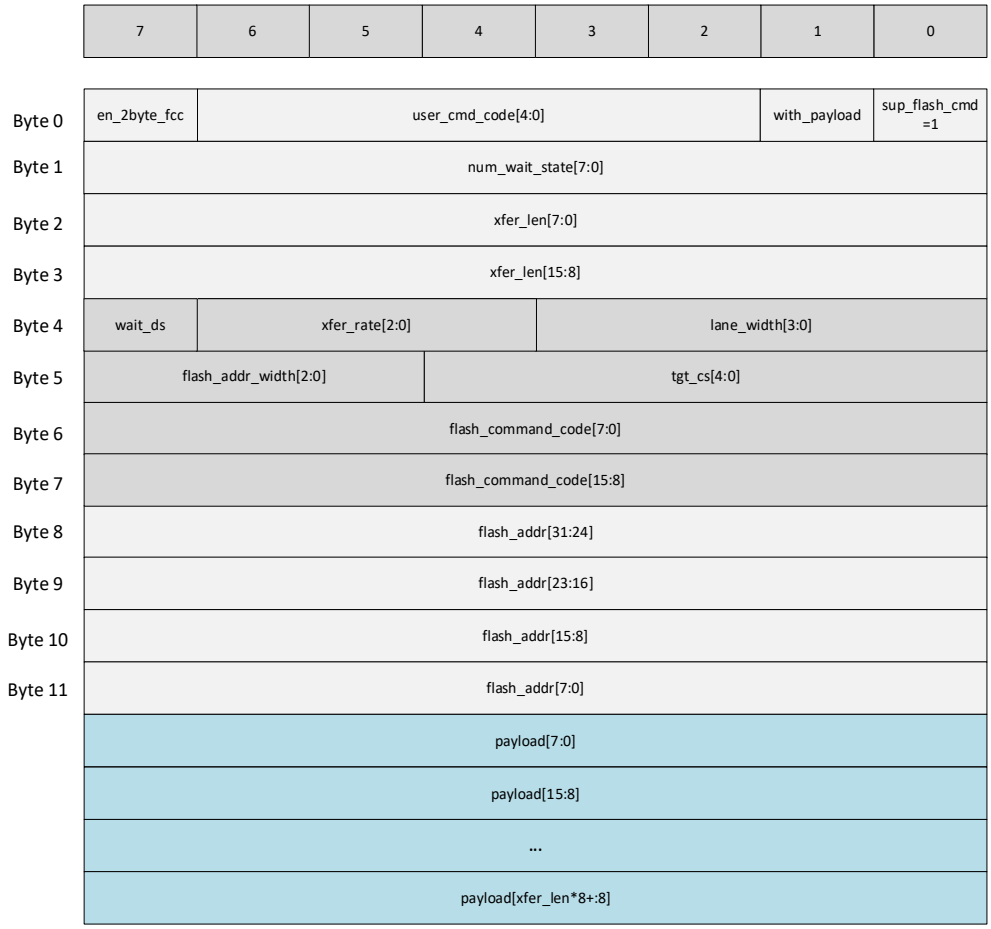


Figure 2.6. Supported Flash Command Packet with 3DW Header

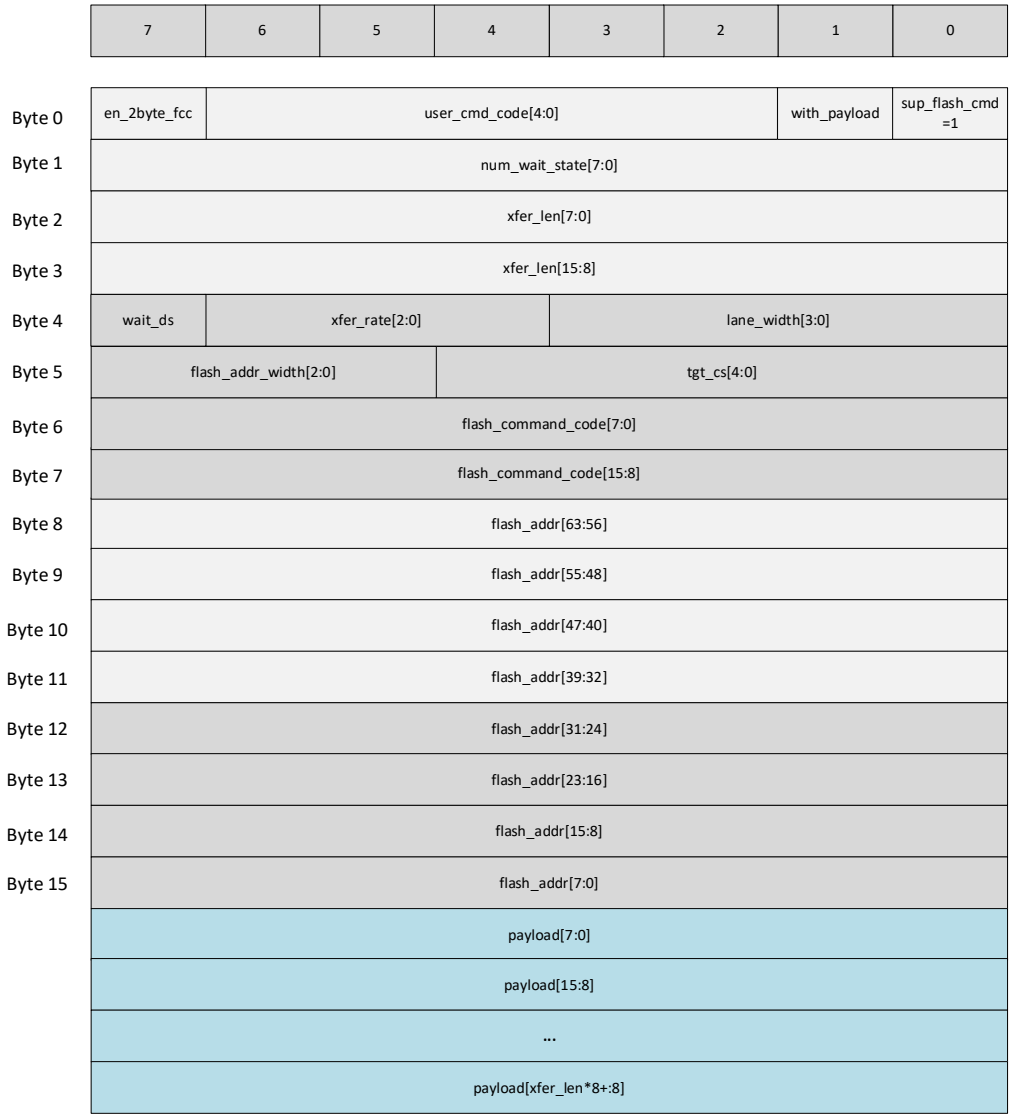


Figure 2.7. Supported Flash Command Packet with 4DW Header

2.5.2. Packet Field Description

Table 2.3. Packet Fields – Generic SPI Command Format

Byte	Name	Field	Description
0	sup_flash_cmd	[0]	0 – Generic SPI command.
	cmd_type	[1]	0 – Generic SPI read transaction. No payload in the packet but read data is sampled and stored in the FIFO or register. 1 – Generic SPI write transaction. Payload is included in the packet except if dummy_data = 1. Applicable when sup_flash_cmd = 0.
	lane_width[1:0]	[3:2]	SPI I/O data width Indicates the number of lanes to use when transmitting or receiving data. 0 – x1 (Standard SPI) 1 – x2 (Dual SPI) 2 – x4 (Quad SPI) 3 – x8 (xSPI) Applicable when sup_flash_cmd = 0. In standard SPI, data is transmitted in spi_dt_io[0] and received from spi_dt_io[1]. In other modes (x2/x4/x8), data lanes are the same for transmit and receive.
	data_rate	[4]	0 – Transaction done using single transfer rate (STR). 1 – Transaction done using double transfer rate (DTR). Applicable when sup_flash_cmd = 0.
	frm_start	[5]	0 – Transaction is a continuation of the previous packet. 1 – Packet is the first of the transaction or a standalone/single packet transaction. This field affects the behavior of the chip select signal. Applicable when sup_flash_cmd = 0.
	frm_end	[6]	0 – Transaction continues in the next packet. 1 – Packet is the last of the transaction or a standalone/single packet transaction. This field affects the behavior of the chip select signal. Applicable when sup_flash_cmd = 0.
	wait_ds or dummy_data	[7]	• wait_ds 0 – No data strobe. Read data is sampled after the specified dummy cycle or wait state. 1 – Read data arrives along with data strobe. The number of dummy cycles is irrelevant so the num_wait_sck field must be set to 0. The Octal SPI Controller IP continuously provides a clock until all the requested data are retrieved. Applicable when sup_flash_cmd = 0 and cmd_type = 0 (this is applicable in DTR when data strobe signal is present (spi_ds_i) and supported by the target device). Set this to 1 when sending an SPI read command (cmd_type = 0) and *JEDEC xSPI Support* is enabled in the IP GUI. • dummy_data 0 – Payload follows after the header. 1 – There is no payload to be transferred and the transfer length specifies the number of wait cycles in multiples of 8 (wait cycles = 8 × xfer_len). Applicable when sup_flash_cmd = 0 and cmd_type = 1.

Byte	Name	Field	Description
1	tgt_cs[4:0]	[4:0]	SPI target chip select Indicates the SPI target device to be selected. Applicable when sup_flash_cmd = 0.
	num_wait_sck[2:0]	[7:5]	Indicates the number of SPI clock cycles generated while waiting for the read data. This is used for number of dummy cycles that is less than 8. 0 – No wait cycle 1 – 1 clock wait cycle 2 – 2 clocks wait cycle ... 7 – 7 clocks wait cycle Applicable when sup_flash_cmd = 0.
2	xfer_len[7:0]	[7:0]	Indicates the number of data bytes to be transmitted or received. 0 – 64K bytes 1 – 1 byte 2 – 2 bytes … 65535 – 65535 bytes
3	xfer_len[15:8]	[7:0]	When sup_flash_cmd = 0 and dummy_data = 1, xfer_len[4:0] is used to indicate the number of dummy cycles in multiples of 8 (xfer_len[15:5] is ignored). 1 – 8 dummy cycles 2 – 16 dummy cycles … 31 – 248 dummy cycles
—	Payload Byte 1…n	[7:0]	The data bytes sent for write access.

Table 2.4. Packet Fields – Supported Flash Command Format

Byte	Name	Field	Description
0	sup_flash_cmd	[0]	1 – Supported flash command. See user_cmd_code for specific pattern of transaction.
	with_payload	[1]	0 – No payload in the packet. 1 – Payload is present in the packet. Applicable when sup_flash_cmd = 1.
	user_cmd_code[4:0]	[6:2]	• user_cmd_code[1:0] indicates the pattern of transaction: 0 – [OPCODE], [ADDRESS], [RD DATA] 1 – [OPCODE], [ADDRESS], [WR DATA] 2 – [WR ENABLE], [OPCODE], [ADDRESS], [WR DATA] 3 – [WR ENABLE], [OPCODE], [ADDRESS], [WR DATA], [CHECK STATUS] Applicable when sup_flash_cmd = 1. **Note:** Address and write data are optional. These are controlled by the flash_addr_width and with_payload fields, respectively. • user_cmd_code[2] is reserved. • user_cmd_code[3] (en_sr_addr) is reserved for transaction patterns 0, 1, and 2. For transaction pattern 3: 0 – No address included. 1 – Include 4-byte address 0x0000_0000 when reading the flash status registers. • user_cmd_code[4] is reserved for transaction patterns 0 and 1. For transaction patterns 2 and 3, this field determines if the same settings are used when sending write enable and when reading the flash status registers: 0 – Use the default transfer setting in the supported command configuration register. 1 – Use the setting in packet field (xfer_rate, lane_width, wait_data_strobe, en_sr_addr, num_wait_state). In transaction pattern 3, the target's read status register is read until the transaction is confirmed done. Additionally, the security register or flag status register is read to check the success/fail status of the program/erase command. The status is reflected in the interrupt status register (0x100).
	en_2byte_fcc	[7]	Enable 2-byte flash command code. 0 – Flash command code is 1 byte only. 1 – Flash command code is 2 bytes. Applicable when sup_flash_cmd = 1 and *Enable 2 bytes command code* == Checked in the IP GUI.
1	num_wait_state[7:0]	[7:0]	Indicates the number of SPI clock cycles generated while waiting for the read data. 0 – No wait cycle 1 – 1 clock wait cycle 2 – 2 clocks wait cycle ... 255 – 255 clocks wait cycle Applicable when sup_flash_cmd = 1.

Byte	Name	Field	Description
2	xfer_len[7:0]	[7:0]	Indicates the number of data bytes to be transmitted or received. 0 – 64K bytes 1 – 1 byte 2 – 2 bytes … 65535 – 65535 bytes
3	xfer_len[15:8]	[7:0]	When sup_flash_cmd = 0 and dummy_data = 1, xfer_len[4:0] is used to indicate the number of dummy cycles in multiples of 8 (xfer_len[15:5] is ignored). 1 – 8 dummy cycles 2 – 16 dummy cycles … 31 – 248 dummy cycles
4	lane_width[1:0]	[3:0]	SPI I/O data width Indicates the number of lanes to use in each phase (Command, Address, and Data). 4'h0 – (x1, x1, x1) 4'h1 – (x2, x2, x2) 4'h2 – (x4, x4, x4) 4'h3 – (x8, x8, x8) 4'h4 – (x1, x1, x2) 4'h5 – (x1, x2, x2) 4'h6–4'h7 – reserved 4'h8 – (x1, x1, x4) 4'h9 – (x1, x4, x4) 4'hA–4'hB – reserved 4'hC – (x1, x1, x8) 4'hD – (x1, x8, x8) 4'hE–4'hF – reserved Applicable when sup_flash_cmd = 1.
	xfer_rate[2:0]	[6:4]	Specifies the transfer rate to be used for each phase: Command (xfer_rate[2]), Address (xfer_rate[1]), and Data (xfer_rate[0]). DTR is currently only valid in x8 width. 0 – STR 1 – DTR Applicable when sup_flash_cmd = 1.
	wait_ds	[7]	Wait for data strobe 1 – Read data arrives along with data strobe. The number of dummy cycles is irrelevant so the num_wait_state field must be set to 0. The Octal SPI Controller IP continuously provides a clock until all the requested data are retrieved. Applicable when sup_flash_cmd = 1 and data strobe is enabled.

Byte	Name	Field	Description
5	tgt_cs[4:0]	[4:0]	SPI target chip select Indicates the SPI target device to be selected. Applicable when sup_flash_cmd = 1.
	flash_addr_width[2:0]	[7:5]	Flash address width 0 – no address 1 – 16b address 2 – 24b address 3 – 32b address … 7 – 64b address Values 4–7 are reserved if *Target Address up to 64b* = Unchecked in the IP GUI. Applicable when sup_flash_cmd = 1.
6	flash_command_code[7:0]	[7:0]	Vendor specific flash command code Applicable when sup_flash_cmd = 1.
7	flash_command_code[15:8]	[7:0]	Vendor specific flash command code Applicable when sup_flash_cmd = 1.
11–8	{flash_addr[7:0], flash_addr[15:8], flash_addr[23:16] flash_addr[31:24]}	[31:0]	Lower 32b flash address Applicable when sup_flash_cmd = 1 and flash_addr_width ≤ 3. The flash address byte order is reversed. Valid bytes start on the lowest byte index (byte 8). For example, in 32b address, if flash_addr_width = 3: Byte 8: flash_addr[31:24] Byte 9: flash_addr[23:16] Byte 10: flash_addr[15:8] Byte 11: flash_addr[7:0] If flash_addr_width < 3, then valid bytes of the flash_addr are shifted towards byte 8. For example, if flash_addr_width = 2, then the flash address is {8'h00, flash_addr[7:0], flash_addr[15:8], flash_addr[23:16]}.
	{flash_addr[39:32], flash_addr[47:40], flash_addr[55:48], flash_addr[63:56]}		Upper 32b flash address Applicable when sup_flash_cmd = 1 and 3 < flash_addr_width ≤ 7. The flash address byte order is reversed. Valid bytes start on the lowest byte index (byte 8). For example, in 64b address, if flash_addr_width = 7: Byte 8: flash_addr[63:56] Byte 9: flash_addr[55:48] Byte 10: flash_addr[47:40] Byte 11: flash_addr[39:32] If flash_addr_width < 7, then valid bytes of the flash_addr are shifted towards byte 8. For example, if flash_addr_width = 4, then the upper flash address is {flash_addr[15:8], flash_addr[23:16], flash_addr[31:24], flash_addr[39:32]}.
15–12	{flash_addr[7:0], flash_addr[15:8], flash_addr[23:16], flash_addr[31:24]}	[31:0]	Lower 32b flash address Applicable when sup_flash_cmd = 1 and 3 < flash_addr_width ≤ 7. The flash address byte order is reversed. For example, in 64b address, if flash_addr_width = 7: Byte 12: flash_addr[31:24] Byte 13: flash_addr[23:16] Byte 14: flash_addr[15:8] Byte 15: flash_addr[7:0] If flash_addr_width < 7, then valid bytes of the flash_addr are shifted towards byte 8. For example, if flash_addr_width = 4, then the lower flash address is {8'h00, 8'h00, 8'h00, flash_addr[7:0]}.

Byte	Name	Field	Description
—	Payload Byte 1…n	[7:0]	The data bytes sent for write access.

2.6. Generating SPI Transactions

2.6.1. SPI Transfer Modes

The SPI transfer mode can be configured by setting the clock polarity and clock phase in the IP GUI. If the respective programmable capabilities are enabled in the IP GUI (*Programmable CPOL*, *Programmable CPHA*), clock polarity and clock phase can be set through configuration 0 register (0x004).

The following figures show the clock idle state and how data is driven and sampled in each mode. Note that for DTR, the SPI transfer mode defaults to mode 0 and driving or sampling of data can occur on both clock edges.

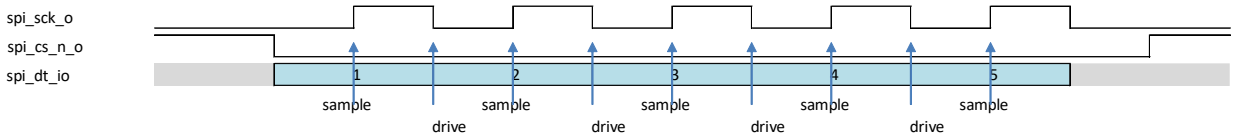


Figure 2.8. SPI Transfer Mode 0 (cpol = 0, cpha = 0)

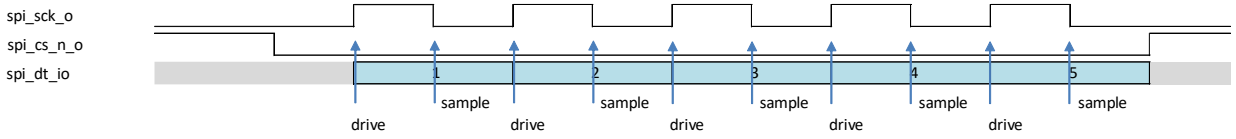


Figure 2.9. SPI Transfer Mode 1 (cpol = 0, cpha = 1)

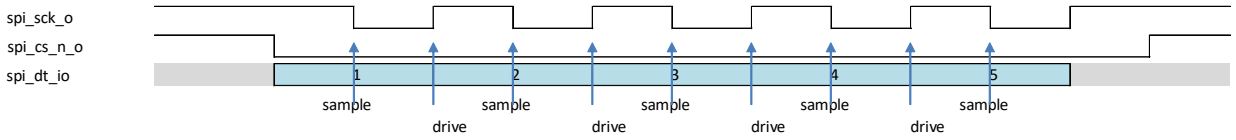


Figure 2.10. SPI Transfer Mode 2 (cpol = 1, cpha = 0)

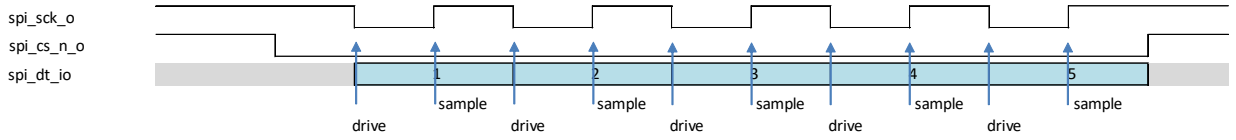


Figure 2.11. SPI Transfer Mode 3 (cpol = 1, cpha = 1)

2.6.2. Data I/O Width

The number of active SPI data I/O lines is dynamically controlled in the packet header. The following figure shows sample waveforms of a byte transfer across different I/O widths. This example uses SPI transfer mode 0 and the MSB first setting.

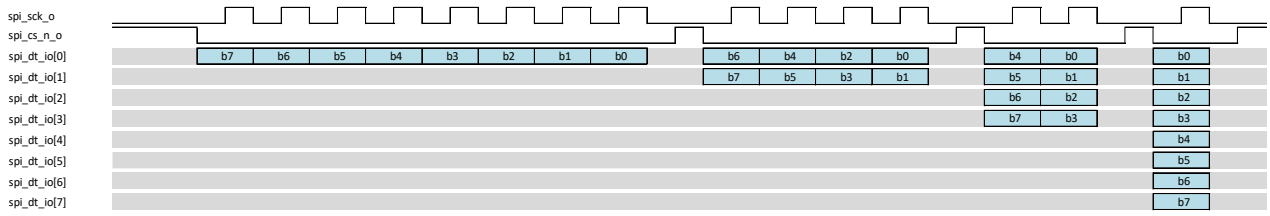


Figure 2.12. Byte Transfer across Different I/O Widths

2.6.3. DTR Mode

Double transfer rate (DTR) mode increases the performance of SPI transfer since it uses both the positive and negative edges of the clock for sampling and driving data. This mode is currently only applicable when using a data I/O width greater than four (octal SPI only). When using DTR, the clock phase and clock polarity are restricted to 0.

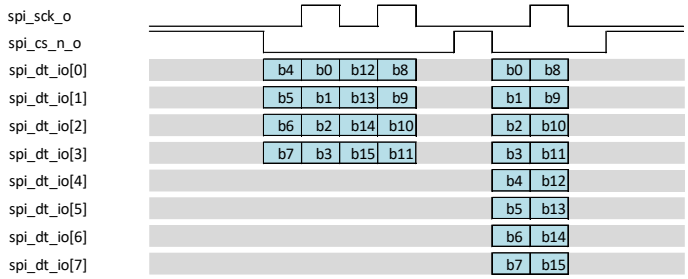


Figure 2.13. Byte Transfer in DTR Mode across x4 and x8 I/O Widths

2.6.4. Generic SPI Write/Read Transactions

In a generic SPI write transaction, the packet payload is transmitted or shifted out on the SPI data I/O line or lines. In a generic SPI read transaction, the received data from input pins is stored in the Rx FIFO or in the read data register. The length of transfer, number of valid SPI data pins, and data rate to use can be controlled in the packet header. Dummy cycles can be controlled and inserted as necessary. The context of a user transaction is not transparent to the IP core. A transaction is simply seen as either a write or read transfer. It is up to you to manipulate the packets in order to convert application-level transactions (such as flash command sequence) into SPI transfer.

There are instances when multiple packets or even transactions combining SPI write and SPI read need to be transferred continuously in a single assertion of chip select. In this case, the frm_start and frm_end packet header fields must be used to indicate that these packets belong to a single frame. This means that target chip select will not de-assert until it encounters a packet with frm_start or frm_end set to 1.

If a transfer can be completed with one packet, then the packet will have frm_start = 1 and frm_end = 1. If a transfer can only be completed with two packets, then the first packet will have frm_start = 1 and frm_end = 0, and the second packet will have frm_start = 0 and frm_end = 1. If a transfer can only be completed with 3 or more packets, then the first packet will have frm_start = 1 and frm_end = 0, the middle packet or packets will have frm_start = 0 and frm_end = 0, and the last packet will have frm_start = 0 and frm_end = 1.

The following figures show example waveform diagrams of generic SPI transactions.

Figure 2.14. SPI Write Transaction

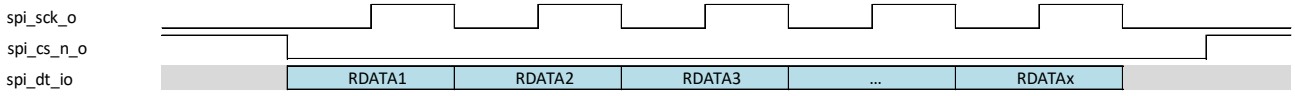


Figure 2.15. SPI Read Transaction

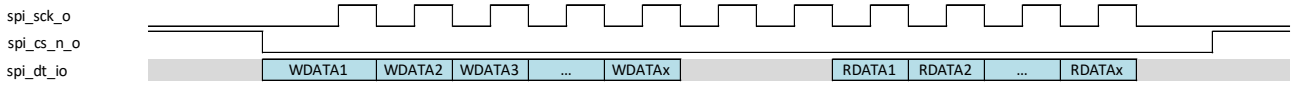


Figure 2.16. SPI Write and Read Transaction with Dummy Cycles

The following tables show the AXI4 bus transactions to perform SPI transfer.

Table 2.5. Generic SPI Transaction Example: Write 16 Bytes in x8 STR

AXI4 Address	Write Data	Description/Comment
Register Map: AXI4 Base Address + 0x200	0x0010_006E	[0] sup_flash_cmd = 0 (Generic)
		[1] cmd_type = 1 (Write)
		[3:2] lane_width = 3 (x8)
		[4] data_rate = 0 (STR)
		[5] frm_start = 1
		[6] frm_end = 1
		[7] dummy_data = 0
		[12:8] tgt_cs = 0
		[15:13] num_wait_sck = 0
		[31:16] xfer_len = 16 (bytes)
	0x1234_5678	Payload DW1
	0x9ABC_DEF0	Payload DW2
	0x1122_3344	Payload DW3
	0x5566_7788	Payload DW4
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction

Table 2.6. Generic SPI Transaction Example: Read 16 Bytes in x1 STR, with 12 Dummy Cycles

AXI4 Address	Write Data	Description/Comment
Register Map: AXI4 Base Address + 0x200	0x0001_00A0	[0] sup_flash_cmd = 0 (Generic)
		[1] cmd_type = 1 (dummy write)
		[3:2] lane_width = 0 (x1)
		[4] data_rate = 0 (STR)
		[5] frm_start = 1
		[6] frm_end = 0
		[7] dummy_data = 1 (dummy – no data)
		[12:8] tgt_cs = 0
		[15:13] num_wait_sck = 0
		[31:16] xfer_len = 1 (x8 dummy cycle)
	0x0010_8040	[0] sup_flash_cmd = 0 (Generic)
		[1] cmd_type = 0 (Read)
		[3:2] lane_width = 0 (x1)
		[4] data_rate = 0 (STR)
		[5] frm_start = 0
		[6] frm_end = 1
		[7] wait_ds = 0
		[12:8] tgt_cs = 0
		[15:13] num_wait_sck = 4 (dummy cycle)
		[31:16] xfer_len = 16 (bytes)
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction

After performing the transaction, check the interrupt status register rx_fifo_not_empty or rdata_available to determine if there is read data available.

Table 2.7. Generic SPI Transaction Example: Write 5 Bytes in x8 DTR, then Read 8 Bytes in x8 STR, with 2 Dummy Cycles

AXI4 Address	Write Data	Description/Comment
Register Map: AXI4 Base Address + 0x200	0x0005_0032	[0] sup_flash_cmd = 0 (Generic)
		[1] cmd_type = 1 (Write)
		[3:2] lane_width = 0 (x8)
		[4] data_rate = 1 (DTR)
		[5] frm_start = 1
		[6] frm_end = 0
		[7] dummy_data = 0
		[12:8] tgt_cs = 0
		[15:13] num_wait_sck = 0
		[31:16] xfer_len = 5 (bytes)
	0x55BB_44AA	Payload DW1
	0x0000_00CC	Payload DW2
	0x0008_4040	[0] sup_flash_cmd = 0 (Generic)
		[1] cmd_type = 0 (Read)
		[3:2] lane_width = 0 (x8)
		[4] data_rate = 0 (STR)
		[5] frm_start = 0
		[6] frm_end = 1
		[7] dummy_data = 0
		[12:8] tgt_cs = 0
		[15:13] num_wait_sck = 2 (dummy cycle)
		[31:16] xfer_len = 8 (bytes)
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction

2.6.5. Supported Flash Commands

Applicable only if *Enable Command Processing* is selected in the IP GUI.

For flash controller application, supported flash commands offer a simplified user packet to generate and complete commonly used flash commands (for example, page program, erase, and read). While not all command patterns are supported, these commands can still be generated using generic SPI write/read transactions.

The following table shows the command patterns supported by the IP core.

Table 2.8. Supported Command Patterns

Command Pattern	Description
Command Pattern 0	[OPCODE] + [OPTIONAL ADDRESS] + [READ DATA]
Command Pattern 1	[OPCODE] + [OPTIONAL ADDRESS] + [OPTIONAL WRITE DATA]
Command Pattern 2	[WRITE ENABLE] + [OPCODE] + [OPTIONAL ADDRESS] + [OPTIONAL WRITE DATA]
Command Pattern 3	[WRITE ENABLE] + [OPCODE] + [OPTIONAL ADDRESS] + [OPTIONAL WRITE DATA] + [CHECK STATUS]

The following figures show example commands with the various command patterns.

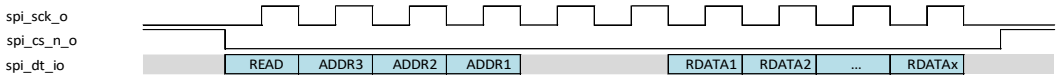


Figure 2.17. Example Command Pattern 0: Read Command (with 2 Dummy Cycles)

Figure 2.18. Example Command Pattern 2: Erase Command

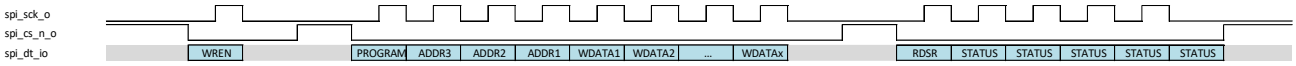


Figure 2.19. Example Command Pattern 3: Page Program Command

The following tables show examples of how to create the supported command transactions.

Table 2.9. Create Supported Flash Command Example: Read 256 Bytes in x1 STR, with 20 Dummy Cycles, Start Address = 0x00_0400

AXI4 Address	Write Data	Description/Comment	
Register Map: AXI4 Base Address + 0x200	0x0100_1401	[0] sup_flash_cmd	= 1 (Supported Command)
		[1] with_payload	= 0 (none)
		[3:2] user_cmd_code[1:0]	= 0 (command pattern 0)
		[6:4] user_cmd_code[4:2]	= 0 (don't care)
		[7] en_2byte_fcc	= 0 (1-byte command)
		[15:8] num_wait_state	= 20 (dummy cycles)
		[31:16] xfer_len	= 256 (bytes)
	0x000B_4000	[3:0] lane_width	= 0 (x1, x1, x1)
		[6:4] xfer_rate	= 0 (STR, STR, STR)
		[7] wait_data_strobe	= 0
		[12:8] tgt_cs	= 0
		[15:13] flash_addr_width	= 2 (24b)
		[31:16] flash_cmd_code	= 0x000B (opcode)
	0x0000_0400	[31:0] flash_addr	= 0x0000_0400 (address)
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction	

After performing the transaction, check the interrupt status register rx_fifo_not_empty or rdata_available to determine if there is read data available.

Table 2.10. Create Supported Flash Command Example: Sector Erase in x1 STR, Start Address = 0x00_0400, Confirm Done Status

AXI4 Address	Write Data	Description/Comment	
Register Map: AXI4 Base Address + 0x200	0x0100_000D	[0]	sup_flash_cmd = 1 (Supported Command)
		[1]	with_payload = 0 (none)
		[3:2]	user_cmd_code[1:0] = 3 (command pattern 3)
		[5:4]	user_cmd_code[3:2] = 0 (x1- status width)
		[6]	user_cmd_code[4] = 0 (STR – status rate)
		[7]	en_2byte_fcc = 0 (1-byte command)
		[15:8]	num_wait_state = 0 (no dummy cycles)
		[31:16]	xfer_len = 0 (don't care)
	0x0020_4000	[3:0]	lane_width = 0 (x1, x1, x1)
		[6:4]	xfer_rate = 0 (STR, STR, STR)
		[7]	wait_data_strobe = 0
		[12:8]	tgt_cs = 0
		[15:13]	flash_addr_width = 2 (24b)
		[31:16]	flash_cmd_code = 0x0020 (opcode)
	0x0000_0400	[31:0]	flash_addr = 0x0000_0400 (address)
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction	

Table 2.11. Create Supported Flash Command Example: Program 256 Bytes in x8 DTR, Start Address = 0x00_0400

AXI4 Address	Write Data	Comment	
Register Map: AXI4 Base Address + 0x200	0x0100_00FB	[0]	sup_flash_cmd = 1 (Supported Command)
		[1]	with_payload = 1
		[3:2]	user_cmd_code[1:0] = 2 (command pattern 2)
		[5:4]	user_cmd_code[3:2] = 3 (x8- status width)
		[6]	user_cmd_code[4] = 1 (DTR – status rate)
		[7]	en_2byte_fcc = 1 (2-byte command)
		[15:8]	num_wait_state = 0 (no dummy cycles)
		[31:16]	xfer_len = 256 (bytes)
	0xED12_40FF	[3:0]	lane_width = 3 (x8, x8, x8)
		[6:4]	xfer_rate = 7 (DTR, DTR, DTR)
		[7]	wait_data_strobe = 0
		[12:8]	tgt_cs = 0
		[15:13]	flash_addr_width = 2 (24b)
		[31:16]	flash_cmd_code = 0xED12 (opcode)
	0x0000_0400	[31:0]	flash_addr = 0x0000_0400 (address)
	0x7654_3210	[31:0]	payload 1
	0x89AB_CDEF	[31:0]	payload 2
	…	…	
	…	[31:0]	payload 256
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction	

2.7. Target Memory Map Access

Applicable only if *Enable Target Address Map* is selected in the IP GUI.

2.7.1. XiP Access

This feature can be enabled by selecting *Enable Target Address Map* and *Enable XiP Mode* in the IP GUI.

In flash controller application, XiP mode is a method to execute a code directly from the flash memory. Read operation in XiP mode can be optimized because it only requires an address (command code is not required) in succeeding read transactions.

The Octal SPI Controller IP provides an option to map the target flash device to the IP address space so that the flash can be accessed as if it is a local memory. The activation or deactivation of XiP mode in a flash device is vendor specific and can be different for each device. Refer to the flash device data sheet for the necessary steps or required sequence.

After sending the necessary command to enable or activate XiP mode in the flash device, set the enter_xip_mode register (0x220) to indicate that the IP will perform read transaction in XiP mode. This means that the AXI4 read access that hits the target memory map address space will be converted to XiP mode read transaction. The supported command configuration register (0x018) is used when performing XiP mode read transaction. Register fields (such as lane_width_r and xfer_rate_r) should be configured to the appropriate settings.

This following discussion is an example of XiP access. Before performing XiP access, the required conditions are as follows:

- A 128-MiB flash memory device with XiP mode is mapped to the local address space for CPU access.
- Read access uses the xSPI DTR mode with data strobe and 32b address.
- The flash device XiP mode is enabled by writing to a configuration register and activated on the next fast read command through the confirmation bit after the address.

Firstly, map the flash memory device by allocating the local address space and assigning a base address. The Octal SPI Controller IP uses 1 KiB address space for internal registers. The target base address must be aligned to the size of the address space allocation (128 MiB). If the first address space (0x0000_0000) is allocated to the IP core registers, the next address space available is 1 × 128 MiB or 0x0800_0000. Set this in the target AXI4 base address register.

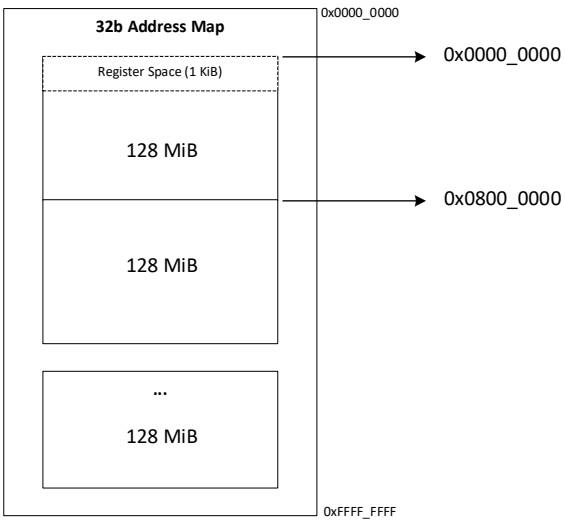


Figure 2.20. Example of Target Device Address Mapping

After address space allocation, configure the transaction settings in the supported command configuration register (lane_width_r = 4'h3, xfer_rate_r = 3'b111, wait_ds_r = 1'b1, addr_mode_r = 1'b1). Next, enable XiP mode in the flash device by sending a generic SPI write to the XiP enable register. Then, send a fast read command (using either the generic packet or supported command packet) with the confirmation bit after the address. Note that this is only applicable for this hypothetical flash device. Different vendors may have their own enable or activation methods which are specified in the respective data sheets.

Finally, assert the enter_xip_mode register (0x220). The CPU will be able to perform read access on the flash device by sending an AXI4 read transaction to the allocated address range of 0x0800_0000 to 0x0FFF_FFFF.

2.7.2. Non-XiP Memory Map Access

If XiP mode is not enabled, enter_xip_mode is not set to high (non-XiP mode), and an AXI4 read access to the target memory address space is performed, the IP core converts this access to a regular flash read command using the command code in the fast_read_b0 and fast_read_b1 registers (0x014). In non-XiP mode, both read and write direct access can be performed. AXI4 write access to the target memory address space is converted to the flash page program command using the command code in the page_program_b0 and page_program_b1 registers (0x014).

2.7.3. FIFO Burst Transfer for Indirect Flash Access

When the dual bus interface is selected and *FIFO Interface Mapping* is set to **Map to Data Interface**, the FIFO can be accessed using the target data interface with indirect SPI transactions. The Tx and Rx FIFO offsets are the same as with the CSR mapping while the upper address (base) can be anything outside the allocation of the target address space. For example, if *Target Map: AXI4 Base Address* is 0x8000_0000 and *Total Target Memory Map Size* is 64 KiB, then the Tx and Rx FIFO can be accessed at offsets 0x200 and 0x204, respectively, with a base address that does not fall within 0x8000_0000 to 0x8000_FFFF. This feature is useful when the CSR interface is using the AXI4-Lite or APB bus and you want to perform a burst to the FIFO instead of a single transfer. Note that this is different from the direct target data path access in that you will still need to create a generic SPI or supported flash command packet to perform flash access.

2.8. Flash Controller Sequence Examples

One of the applications of the Octal SPI Controller IP is to access a flash memory target device.

The typical commands used when accessing a flash target device are erase, program, and read. The examples in the following sections show how to generate these commands in the Octal SPI Controller IP using the generic SPI command packet format and the supported flash command packet format. Each sequence assumes that FIFO is enabled in the IP GUI.

If FIFO is not enabled, packet headers should be written to the packet header registers (0x208 – 0x214) and write data to the write data register (0x218). Read data should be read from the read data register (0x21C). In write access, before writing the next data (except for the first write), wait for wdata_done to assert. When wdata_done asserts, the data has already transferred, and the IP is ready to accept new data. Clear wdata_done before writing new data to the write data register. Alternatively, wdata_not_empty in the octal SPI debug information 0 register (0x10C) can be used to determine if the next data can be written to the write data register. Unlike wdata_done, wdata_not_empty does not need to be cleared. When wdata_not_empty is de-asserted (0), the next data can be written to the write data register. In read access, wait for rdata_available to assert before reading the read data register. Clear rdata_available after getting the read data from the read data register.

2.8.1. Flash Erase Command

2.8.1.1. Using Generic SPI Command Packet

The following table shows the AXI4 bus transactions to generate the flash erase command using generic SPI command packet.

Table 2.12. Using Generic SPI Command Packet: Sector Erase (128 KiB) at Address (0x0159_E5A8), x8 I/O width, DTR, 4-Byte Address Mode

AXI4 Address	Write Data	Description/Comment		
Register Map: AXI4 Base Address + 0x200	0x0002_007E	[0]	sup_flash_cmd	= 0 (Generic)
		[1]	cmd_type	= 1 (Write)
		[3:2]	lane_width	= 3 (x8)
		[4]	data_rate	= 1 (DTR)
		[5]	frm_start	= 1
		[6]	frm_end	= 1
		[7]	dummy_data	= 0
		[12:8]	tgt_cs	= 0
		[15:13]	num_wait_sck	= 0
		[31:16]	xfer_len	= 2 (bytes)
	0x0000_0606	WREN Command code		
	0x0002_003E	[0]	sup_flash_cmd	= 0 (Generic)
		[1]	cmd_type	= 1 (Write)
		[3:2]	lane_width	= 3 (x8)
		[4]	data_rate	= 1 (DTR)
		[5]	frm_start	= 1
		[6]	frm_end	= 0
		[7]	dummy_data	= 0
		[12:8]	tgt_cs	= 0
		[15:13]	num_wait_sck	= 0
		[31:16]	xfer_len	= 2 (bytes)
	0x0000_DCDC	Sector Erase Command Code		
	0x0004_005E	[0]	sup_flash_cmd	= 0 (Generic)
		[1]	cmd_type	= 1 (Write)
		[3:2]	lane_width	= 3 (x8)

AXI4 Address	Write Data	Description/Comment
		[4] data_rate = 1 (DTR)
		[5] frm_start = 0
		[6] frm_end = 1
		[7] dummy_data = 0
		[12:8] tgt_cs = 0
		[15:13] num_wait_sck = 0
		[31:16] xfer_len = 4 (bytes)
	0xA8E5_5901	{Address[7:0], Address[15:8], Address[23:16], Address[31:24]}
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction

Read the register 0x10C to determine if the erase command has been sent. When spi_has_started = 1 and spi_busy = 0, transfer is done. Once the erase command is sent, the flash status register needs to be checked to determine if erase is complete. This is done by sending a read status register command.

Table 2.13. Using Generic SPI Command Packet: Send Read Status Register Command, x8 I/O Width, DTR

AXI4 Address	Write Data	Description/Comment
Register Map: AXI4 Base Address + 0x200	0x0002_003E	[0] sup_flash_cmd = 0 (Generic)
		[1] cmd_type = 1 (Write)
		[3:2] lane_width = 3 (x8)
		[4] data_rate = 1 (DTR)
		[5] frm_start = 1
		[6] frm_end = 0
		[7] dummy_data = 0
		[12:8] tgt_cs = 0
		[15:13] num_wait_sck = 0
		[31:16] xfer_len = 2 (bytes)
	0x0000_0505	Read Status Command code
	0x0002_00DC	[0] sup_flash_cmd = 0 (Generic)
		[1] cmd_type = 0 (Read)
		[3:2] lane_width = 3 (x8)
		[4] data_rate = 1 (DTR)
		[5] frm_start = 0
		[6] frm_end = 1
		[7] wait_ds = 1 (wait for data strobe)
		[12:8] tgt_cs = 0
		[15:13] num_wait_sck = 0
		[31:16] xfer_len = 2 (bytes)
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction

Read the register 0x10C to determine if the read status command is done (spi_has_started = 1 and spi_busy = 0). The flash status data should now be available in the Rx FIFO (if FIFO is enabled in the IP GUI), which can be read at 0x204, or at 0x21C (if FIFO is not enabled).

The flash status busy bit (usually at bit 0) should indicate if the operation is still ongoing (busy = 1) or done (busy = 0). The flash program command may not proceed if the previous command (erase) is not done. The read status command should be repeatedly sent until the busy bit shows that the operation is done.

Some flash devices have another status register that indicates if the erase or program command has failed. You may opt to read this status register by sending another read command that targets the status register.

2.8.1.2. Using Supported Flash Command Packet

The following discusses how to perform flash erase using supported flash command packet.

Before sending any supported flash command packet, ensure that the supported command code 0 and 1 registers (0x00C and 0x010) have been configured either in the IP GUI or through programming (if *Programmable Command Code* is enabled). The supported command configuration register (0x018) must be set to the intended configuration if a different setting from that set in the supported flash command packet (for example, when user_cmd_code[4]==0) is to be used. The write enable and read status command sequences are automatically executed by the Octal SPI Controller IP when the supported flash command packet is used in the flash erase, program, or write command. The IP core relies on the pre-configured command codes and settings (such as I/O width and data rate when user_cmd_code[4]==0) to generate these command sequences.

Table 2.14. Using Supported Flash Command Packet: Sector Erase (128 KiB) at Address (0x0159_E5A8), x8 I/O Width, DTR, 4-Byte Address Mode

AXI4 Address	Write Data	Description/Comment		
Register Map: AXI4 Base Address + 0x200	0x0001_00CD	[0]	sup_flash_cmd	= 1 (Supported Command)
		[1]	with_payload	= 0 (none)
		[3:2]	user_cmd_code[1:0]	= 3 (command pattern 3)
		[5:4]	user_cmd_code[3:2]	= 0 (don't care)
		[6]	user_cmd_code[4]	= 1 use packet field settings
		[7]	en_2byte_fcc	= 1 (2 bytes command)
		[15:8]	num_wait_state	= 0 (dummy cycles)
		[31:16]	xfer_len	= 1 (no payload)
	0xDCDC_6073	[3:0]	lane_width	= 3 (x8, x8, x8)
		[6:4]	xfer_rate	= 7 (DTR, DTR, DTR)
		[7]	wait_data_strobe	= 0
		[12:8]	tgt_cs	= 0
		[15:13]	flash_addr_width	= 3 (32b)
		[31:16]	flash_cmd_code	= 0xDCDC (Sector Erase)
	0xA8E5_5901	{Address[7:0], Address[15:8], Address[23:16], Address[31:24]}		
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction		

Once this command is activated (tx_start = 1 for start transaction register), the Octal SPI Controller IP generates the write enable sequence, followed by the erase command sequence, and finally the read status and read status flag sequence.

The register 0x10C may be read to determine if the erase command is complete (spi_has_started = 1 and spi_busy = 0). For flash devices that support the status flag for erase fail or program fail, this status is reflected in the interrupt status register (0x100). It is important that the correct bit index be configured in the IP GUI (*Flash Status Register Erase Fail bit index* or *Flash Status Register Program Fail bit index*) together with the appropriate command code in register 0x010. Otherwise, this status is meaningless and should be ignored.

2.8.2. Flash Program Command

2.8.2.1. Using Generic SPI Command Packet

The following discusses how to perform flash program using generic SPI command packet.

Table 2.15. Using Generic SPI Command Packet: Page Program (256 Bytes) at Address (0x0159_E500), x8 I/O Width, DTR, 4-Byte Address Mode

AXI4 Address	Write Data	Description/Comment	
Register Map: AXI4 Base Address + 0x200	0x0002_007E	[0] sup_flash_cmd	= 0 (Generic)
		[1] cmd_type	= 1 (Write)
		[3:2] lane_width	= 3 (x8)
		[4] data_rate	= 1 (DTR)
		[5] frm_start	= 1
		[6] frm_end	= 1
		[7] dummy_data	= 0
		[12:8] tgt_cs	= 0
		[15:13] num_wait_sck	= 0
		[31:16] xfer_len	= 2 (bytes)
	0x0000_0606	WREN Command code	
	0x0002_003E	[0] sup_flash_cmd	= 0 (Generic)
		[1] cmd_type	= 1 (Write)
		[3:2] lane_width	= 3 (x8)
		[4] data_rate	= 1 (DTR)
		[5] frm_start	= 1
		[6] frm_end	= 0
		[7] dummy_data	= 0
		[12:8] tgt_cs	= 0
		[15:13] num_wait_sck	= 0
		[31:16] xfer_len	= 2 (bytes)
	0x0000_1212	Page Program Command Code	
	0x0004_001E	[0] sup_flash_cmd	= 0 (Generic)
		[1] cmd_type	= 1 (Write)
		[3:2] lane_width	= 3 (x8)
		[4] data_rate	= 1 (DTR)
		[5] frm_start	= 0
		[6] frm_end	= 0
		[7] dummy_data	= 0
		[12:8] tgt_cs	= 0
		[15:13] num_wait_sck	= 0
		[31:16] xfer_len	= 4 (bytes)
	0x00E5_5901	{Address[7:0], Address[15:8], Address[23:16], Address[31:24]}	
	0x0100_005E	[0] sup_flash_cmd	= 0 (Generic)
		[1] cmd_type	= 1 (Write)
		[3:2] lane_width	= 3 (x8)
		[4] data_rate	= 1 (DTR)
		[5] frm_start	= 0
		[6] frm_end	= 1
		[7] dummy_data	= 0
		[12:8] tgt_cs	= 0

AXI4 Address	Write Data	Description/Comment
		[15:13] num_wait_sck = 0
		[31:16] xfer_len = 256 (bytes)
	0x0403_0201	Payload_DW0[31:0]
	0x0807_0605	Payload_DW1[31:0]
	…	…
	…	…
	0x00FF_FEFD	Payload_DW63[31:0]
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction

Follow the same procedure for flash erase (refer to 2.8.1.1 Using Generic SPI Command Packet) to check the status until it is determined that the program operation is done.

2.8.2.2. Using Supported Flash Command Packet

The following discusses how to perform flash program using supported flash command packet.

Similar to the erase command, ensure that all necessary register settings have been configured prior to sending any supported flash command packet (refer to 2.8.1.2 Using Supported Flash Command Packet).

Table 2.16. Using Supported Flash Command Packet: Page Program (256 Bytes) at Address (0x0159_E500), x8 I/O Width, DTR, 4-Byte Address Mode

AXI4 Address	Write Data	Comment
Register Map: AXI4 Base Address + 0x200	0x0100_00CF	[0] sup_flash_cmd = 1 (Supported Command)
		[1] with_payload = 1
		[3:2] user_cmd_code[1:0] = 3 (command pattern 3)
		[5:4] user_cmd_code[3:2] = 0 (don't care)
		[6] user_cmd_code[4] = 1 use packet field settings
		[7] en_2byte_fcc = 1 (2 bytes command)
		[15:8] num_wait_state = 0 (dummy cycles)
		[31:16] xfer_len = 256 (bytes)
	0x1212_6073	[3:0] lane_width = 3 (x8, x8, x8)
		[6:4] xfer_rate = 7 (DTR, DTR, DTR)
		[7] wait_data_strobe = 0
		[12:8] tgt_cs = 0
		[15:13] flash_addr_width = 3 (32b)
		[31:16] flash_cmd_code = 0x1212 (Page Program)
	0x00E5_5901	{Address[7:0], Address[15:8], Address[23:16], Address[31:24]}
	0x0403_0201	Payload_DW0[31:0]
	0x0807_0605	Payload_DW1[31:0]
	…	…
	…	…
	0x00FF_FEFD	Payload_DW63[31:0]
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction

Once this command is activated (tx_start = 1 for start transaction register), the Octal SPI Controller IP generates the write enable sequence, followed by the program command sequence, and finally the read status and read status flag sequence.

The register 0x10C may be read to determine if the program command is complete (spi_has_started = 1 and spi_busy = 0). For flash devices that support the status flag for erase fail or program fail, this status is reflected in the interrupt status register (0x100). It is important that the correct bit index be configured in the IP GUI (*Flash Status Register Erase*

Fail bit index or *Flash Status Register Program Fail bit index*) together with the appropriate command code in register 0x010. Otherwise, this status is meaningless and should be ignored.

2.8.3. Flash Read Command

2.8.3.1. Using Generic SPI Command Packet

The following discusses how to perform flash read using generic SPI command packet.

Table 2.17. Using Generic SPI Command Packet: Fast Read (256 Bytes) at Address (0x0159_E500), x8 I/O Width, DTR, 4-Byte Address Mode

AXI4 Address	Write Data	Comment	
Register Map: AXI4 Base Address + 0x200	0x0002_003E	[0] sup_flash_cmd	= 0 (Generic)
		[1] cmd_type	= 1 (Write)
		[3:2] lane_width	= 3 (x8)
		[4] data_rate	= 1 (DTR)
		[5] frm_start	= 1
		[6] frm_end	= 0
		[7] dummy_data	= 0
		[12:8] tgt_cs	= 0
		[15:13] num_wait_sck	= 0
		[31:16] xfer_len	= 2 (bytes)
	0x0000_0C0C	Fast Read Command code	
	0x0004_001E	[0] sup_flash_cmd	= 0 (Generic)
		[1] cmd_type	= 1 (Write)
		[3:2] lane_width	= 3 (x8)
		[4] data_rate	= 1 (DTR)
		[5] frm_start	= 0
		[6] frm_end	= 0
		[7] dummy_data	= 0
		[12:8] tgt_cs	= 0
		[15:13] num_wait_sck	= 0
		[31:16] xfer_len	= 4 (bytes)
	0x00E5_5901	{Address[7:0], Address[15:8], Address[23:16], Address[31:24]}	
	0x0100_00DC	[0] sup_flash_cmd	= 0 (Generic)
		[1] cmd_type	= 0 (Read)
		[3:2] lane_width	= 3 (x8)
		[4] data_rate	= 1 (DTR)
		[5] frm_start	= 0
		[6] frm_end	= 1
		[7] wait_ds	= 1 (wait for data strobe)
		[12:8] tgt_cs	= 0
		[15:13] num_wait_sck	= 0
		[31:16] xfer_len	= 256 (bytes)
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction	

Read the register 0x10C to determine if the fast read command is done (spi_has_started = 1 and spi_busy=0). Once it is done, all the expected read data should be in the Rx FIFO and can be read at 0x204. It is also possible to get the read data even if fast read is not yet done. Monitor or poll the register 0x100 to check the rx_fifo_not_empty status before reading 0x204.

If FIFO is not enabled, continuously check the interrupt status (0x100) rdata_available. Once the read data is available, read the data and then clear the interrupt status. Otherwise, the read command will not proceed further after the first four bytes since there is no space to store the data.

2.8.3.2. Using Supported Flash Command Packet

The following discusses how to perform flash read using supported flash command packet.

Table 2.18. Using Supported Flash Command Packet: Fast Read (256 Bytes) at Address (0x0159_E500), x8 I/O Width, DTR, 4-Byte Address Mode

AXI4 Address	Write Data	Description/Comment	
Register Map: AXI4 Base Address + 0x200	0x0100_0081	[0] sup_flash_cmd	= 1 (Supported Command)
		[1] with_payload	= 0 (none)
		[3:2] user_cmd_code[1:0]	= 0 (command pattern 3)
		[5:4] user_cmd_code[3:2]	= 0 (don't care)
		[6] user_cmd_code[4]	= 0 (don't care)
		[7] en_2byte_fcc	= 1 (2 bytes command)
		[15:8] num_wait_state	= 0 (dummy cycles)
		[31:16] xfer_len	= 256 (bytes)
	0x0C0C_60F3	[3:0] lane_width	= 3 (x8, x8, x8)
		[6:4] xfer_rate	= 7 (DTR, DTR, DTR)
		[7] wait_data_strobe	= 1
		[12:8] tgt_cs	= 0
		[15:13] flash_addr_width	= 3 (32b)
		[31:16] flash_cmd_code	= 0x0C0C (Fast Read)
	0x00E5_5901	{Address[7:0], Address[15:8], Address[23:16], Address[31:24]}	
Register Map: AXI4 Base Address + 0x220	0x0000_0001	Start Transaction	

Read the register 0x10C to determine if the fast read command is done (spi_has_started = 1 and spi_busy = 0). Once it is done, all the expected read data should be in the Rx FIFO and can be read at 0x204. It is also possible to get the read data even if fast read is not yet done. Monitor or poll the register 0x100 to check the rx_fifo_not_empty status before reading 0x204.

If FIFO is not enabled, continuously check the interrupt status (0x100) rdata_available. Once the read data is available, read the data and then clear the interrupt status. Otherwise, the read command will not proceed further after the first four bytes since there is no space to store the data.

3. IP Parameter Description

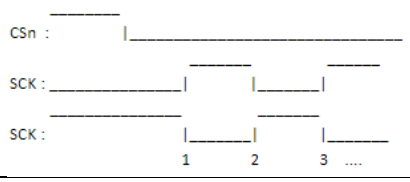
The configurable attributes of the Octal SPI Controller IP are shown in the following tables. You can configure the IP by setting the attributes accordingly in the IP Catalog Module/IP wizard of the Lattice Radiant software.

Wherever applicable, default values are in bold.

3.1. General

Table 3.1. General Attributes

Attribute	Selectable Values	Description
Configuration Preset		
SPI Protocol	Custom, **JEDEC xSPI**	Preconfigures the required settings for a particular protocol. **Note:** The JEDEC xSPI mode uses the UMXSPI primitive which is only available for the LAV-AT-X70 device.
Clock Configuration		
System Clock Frequency (MHz)	1 – **200**	Sets the system clock (clk_i) frequency.
Internal Clock Frequency (MHz)	**200**	Informational only.
SPI Clock (SCK) Pulse width	0 – 31, **1**	Sets the SCK pulse width in terms of number of cycles of system clock (clk_i). Setting this to 0 means that SCK will have the same frequency as the system clock (clk_i). When *JEDEC xSPI Support* = Unchecked, the minimum SPI clock pulse width is 1. The range of selectable values vary with the supported maximum SCK frequency, which depends on the FPGA device selected and settings such as whether DTR is enabled. For example, Nexus devices support a maximum SCK frequency of 100 MHz in STR mode and only 50 MHz in DTR mode.
SPI Clock (SCK) Frequency (MHz)	**100**	Informational only.
SPI Clock (SCK) Period (ns)	**10**	Informational only.
User Interface		
Interface	**AXI4**, AXI-Lite, APB, Data: AXI4; CSR: AXI4, Data: AXI4; CSR: AIX4-Lite, Data: AXI4; CSR: APB	Selects the bus interface. If *Enable Target Address Map* == Checked, selects single bus interface or dual bus interface. In single bus interface, CSR access and direct target data path access are on the same bus. In dual bus interface, CSR access and direct target data path access are on separate buses. The option for FIFO mapping is also available through the *FIFO Interface Mapping* attribute.
Number of outstanding request	**1** – 16	Specifies the AXI4 request queue size.
Enable AXI4-Lite ID	Checked, **Unchecked**	Enables the optional transaction ID in AXI4-Lite. Applicable when the AXI4-Lite bus is selected.
AXI4 Bus ID Width	**1** – 12	Specifies the AXI4 bus ID width.
FIFO Interface Mapping	**Map to CSR Interface**, Map to Data Interface	Selects the bus interface used to access the FIFO. Applicable when the dual bus interface is selected.
FIFO depth	64, 128, **256**, 512	Specifies the depth of transmit and receive FIFO. Applicable when *Include FIFO* == Checked.
FIFO Implementation	**HARD IP**, EBR, LUT	Specifies the FIFO implementation type. Applicable when *Include FIFO* == Checked.

Attribute	Selectable Values	Description
SPI Configuration		
Default LSBF value	**0**, 1	Default value of transmit/receive LSB first setting. 0 – Transmit/Receive MSB first 1 – Transmit/Receive LSB first
Default CPOL value	**0**, 1	Default value of clock polarity setting. 0 – Clock is low in idle state 1 – Clock is high in idle state When *Enable Double Transfer Rate* == Checked, CPOL is fixed to 0.
Default CPHA value	**0**, 1	Default value of clock phase setting. 0 – Sample data at odd number clock edge 1 – Sample data at even number clock edge When *Enable Double Transfer Rate* == Checked, CPHA is fixed to 0. 
Default Endianness value	**0**, 1	Default value of endianness. 0 – Little endian 1 – Big endian
Default Use DS value	0, **1**	Default value of use data strobe setting. 0 – Do not use data strobe 1 – Use data strobe to sample read data in double transfer rate When *SPI Protocol* == JEDEC xSPI, DS is required so this setting is fixed to 1. When *SPI Protocol* = Custom, this setting is applicable only when *Enable Double Transfer Rate* == Checked.
Default Non-blocking Tx FIFO	**0**, 1	Default value of non-blocking Tx FIFO setting. 0 – Wait for Tx FIFO not full status during write access; block write access if full **Note:** This may prevent any write access (including register access) until Tx FIFO has free space. 1 – Do not wait for Tx FIFO not full status; discard data when full
Default Non-blocking Rx FIFO	**0**, 1	Default value of non-blocking Rx FIFO setting. 0 – Wait for Rx FIFO not empty status during read access; block read access if empty **Note:** This may prevent any read access (including register access) until Rx FIFO has available data. 1 – Do not wait for Rx FIFO not empty status; return any data (garbage) if empty
Minimum Timing (number of SPI clock): CS assert to SCK	0 – 15, **1**	Sets the minimum timing from chip select (CS) assertion to the first SPI clock edge in terms of SPI clock cycles. For example, if SPI clock is running at 25 MHz, a value of 1 means at least 40 ns.
Minimum Timing (number of SPI clock): SCK to CS deassert	**0** – 15	Sets the minimum timing from the last SPI clock edge to CS de-assertion in terms of SPI clock cycles. For example, if SPI clock is running at 25 MHz, a value of 1 means at least 40 ns.
Minimum Timing (number of SPI clock): CS deassert to CS assert	0 – 15, **1**	Sets the minimum timing from CS de-assertion to the next CS assertion in terms of SPI clock cycles. For example, if SPI clock is running at 25 MHz, a value of 1 means at least 40 ns.

Attribute	Selectable Values	Description
Supported Capabilities		
Generic SPI Controller	**Checked**	Informational only.
JEDEC xSPI Support	**Checked**, Unchecked	Informational only. Enables support for JEDEC xSPI standard. Automatically sets when selected in the *SPI Protocol* preset.
Enable Target Address Map	Checked, **Unchecked**	Enables support for mapping the target address space to AXI4.
Enable XiP Mode	Checked, **Unchecked**	Enables support for XiP mode. This feature is only applicable if the flash device used is either Micron™ or Macronix™ or has a similar method of activating and keeping the XiP mode. Applicable when *Enable Target Address Map* == Checked.
Enable Command Processing	Checked, **Unchecked**	Enables capability to implement a sequence of transactions in order to complete certain flash commands.
Enable Double Transfer Rate	**Checked**, Unchecked	Enables capability to use both the positive and negative edges of the clock to transfer data. Fixed to DTR if *JEDEC xSPI Support* == Checked.
Max number of data lanes	**X8**, X4, X2, X1	Sets the maximum number of data I/O lines. X1 (SPI) – 1 data pin for transmit and 1 data pin for receive X2 (Dual SPI) – 2 bidirectional data pins for transmit and receive X4 (Quad SPI) – 4 bidirectional data pins for transmit and receive X8 (Octal SPI) – 8 bidirectional data pins for transmit and receive Fixed to X8 if *JEDEC xSPI Support* == Checked.
Max number of SPI Target	**1** – 32	Sets the maximum number of SPI targets supported. This determines the number of target chip select. Fixed to 1 if *JEDEC xSPI Support* == Checked.
Include FIFO	**Checked**, Unchecked	Adds FIFO for transmit and receive data.
Programmable SCK Divider	**Checked**, Unchecked	Enables programmable SCK divider (sck_rate). Otherwise, the corresponding field is non-programmable (cannot be modified).
Programmable LSBF	Checked, **Unchecked**	Enables programmable transmit/receive LSB first setting. Otherwise, the corresponding field is non-programmable (cannot be modified).
Programmable CPOL	Checked, **Unchecked**	Enables programmable clock polarity setting. Otherwise, the corresponding field is non-programmable (cannot be modified). Not applicable when *Enable Double Transfer Rate* == Checked.
Programmable CPHA	Checked, **Unchecked**	Enables programmable clock phase setting. Otherwise, the corresponding field is non-programmable (cannot be modified). Not applicable when *Enable Double Transfer Rate* == Checked.
Programmable Use DS	Checked, **Unchecked**	Enables programmable use data strobe setting. Otherwise, the corresponding field is non-programmable (cannot be modified). Not applicable when *Enable Double Transfer Rate* = Unchecked or *JEDEC xSPI Support* == Checked.
Programmable Command Code	**Checked**, Unchecked	Enables programmable supported command code. Otherwise, the corresponding fields are non-programmable (cannot be modified). Applicable only when *Enable Target Address Map* == Checked or *Enable Command Processing* == Checked.
Programmable Memory Map	Checked, **Unchecked**	Enables programmable target memory map size and base address. Otherwise, the corresponding field is non-programmable (cannot be modified). Applicable only when *Enable Target Address Map* == Checked.
Programmable Non-blocking FIFO setting	Checked, **Unchecked**	Enables programmable setting for non-blocking/blocking FIFO access.
Target Address up to 64b	Checked, **Unchecked**	Enables support for command modifier/address up to 64 bits.
Enable Target Reset Pin	Checked, **Unchecked**	Adds an optional target reset output pin.

Attribute	Selectable Values	Description
IO Primitive		
Include 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. Fixed to Checked when *JEDEC xSPI Support* == Checked.
Use xSPI IO Primitive	**Checked**, Unchecked	Uses the dedicated I/O primitive for xSPI (UMXSPI primitive). This is only applicable when enabling xSPI in the Avant device. Fixed to Checked when *JEDEC xSPI Support* == Checked.
Memory Map Configuration		
Register Space Size (KiB)	—	Informational only.
Enable Full CSR Address Decoding	Checked, **Unchecked**	If enabled, the register address is decoded using the full 32b address. If disabled, only the lower 10 bits of the address are decoded. Disabling full decode means that the bus controller will ensure the correct routing of requests to individual targets. This is only applicable for register space. This setting must be disabled when used in Propel-based designs where the address space is automatically allocated.
Register Map: AXI4 Base Address (32b Hex)	**0x0000_0000** – 0xFFFF_F400	This is the starting address of the register space. Applicable when *Enable Full CSR Address Decoding* == Checked.
Target Memory Map Size (KiB)	**1** – 2,097,152 (power of 2)	This is the size of the address space to be mapped per target. Applicable when *Enable Target Address Map* == Checked.
Target Actual Start Address (32b Hex)	**0x0000_0000** – 0xFFFF_F400	This is the actual starting address of the memory space in the target. The start address must be aligned to the target memory map size. Applicable when *Enable Target Address Map* == Checked.
Total Target Memory Map Size (KiB)	**1** – 2,097,152 (power of 2)	This is the total size of all target address space mapping (target memory map size × number of targets). Applicable when *Enable Target Address Map* == Checked and *Max number of SPI Target* > 1.
Target Map: AXI4 Base Address (32b Hex)	**0x0000_0400** – 0xFFFF_F400	This is the starting address of the target memory map space. The upper bits of the base address must be set to 0 when used in Propel-based designs where the address space is automatically allocated. For example, if the total memory map size is 64 KiB, then the upper 15 bits must be 0 and the base address is 0x0001_0000. Applicable when *Enable Target Address Map* == Checked.
Optional Debug Registers		
Enable Tx FIFO free space count	Checked, **Unchecked**	Adds a register that can be read to check the number of free space in the Tx FIFO. At reset, the number of free space is equal to the FIFO depth. The count is decremented each time data is written to the FIFO and incremented each time data is read from the FIFO.
Enable Rx FIFO available data count	Checked, **Unchecked**	Adds a register that can be read to check the number of available Rx FIFO data. The count is incremented each time data is written to the FIFO and decremented each time data is read from the FIFO.
Enable Generic Packet Counter	Checked, **Unchecked**	Adds a register showing the number of generic packets done.
Enable Supported Command Packet Counter	Checked, **Unchecked**	Adds a register showing the number of supported command packets done. Applicable when *Enable Command Processing* == Checked.
Packet Counter max count	**16** - 65535	Maximum count determines the bit width of the packet counters (generic and supported command packet counters). Applicable when *Enable Generic Packet Counter* == Checked.

3.2. Flash Device Configuration

This is applicable when *Enable Target Address Map* == Checked or *Enable Command Processing* == Checked. The default values shown in the following table are for Micron flash devices. The attribute values or settings may be different for flash devices from different vendors. Refer to the specific flash device data sheet before configuring these parameters.

Table 3.2. Flash Device Configuration Attributes

Attribute	Selectable Values	Description
Vendor Specific Flash Command Code		
Enable 2 bytes command code	**Checked**, Unchecked	Uses 2-byte command codes. This is usually applicable in double transfer rate (DTR) mode. Fixed to Checked when *Enable Double Transfer Rate* == Checked.
Write Enable (byte 1)	0x00–0xFF, **0x06**	Byte 1 of write enable command code.
Write Enable (byte 2)	0x00–0xFF, **0xF9**	Byte 2 of write enable command code. Applicable when *Enable 2 bytes command code* == Checked.
Read Status Register (byte 1)	0x00–0xFF, **0x05**	Byte 1 of read status register command code.
Read Status Register (byte 2)	0x00–0xFF, **0xFA**	Byte 2 of read status register command code. Applicable when *Enable 2 bytes command code* == Checked.
Read Security Register (byte 1)	0x00–0xFF, **0x70**	Byte 1 of read security register command code. This is another status register. Some flash devices may support this or may have an equivalent register. If it is not available or not supported, set this to the same value as the read status register (byte 1).
Read Security Register (byte 2)	0x00–0xFF, **0x8F**	Byte 2 of read security register command code. This is another status register. Some flash devices may support this or may have an equivalent register. If it is not available or not supported, set this to the same value as the read status register (byte 2). Applicable when *Enable 2 bytes command code* == Checked.
Page Program (byte 1)	0x00–0xFF, **0x02**	Byte 1 of page program command code.
Page Program (byte 2)	0x00–0xFF, **0xFD**	Byte 2 of page program command code. Applicable when *Enable 2 bytes command code* == Checked.
Fast Read (byte 1)	0x00–0xFF, **0x0B**	Byte 1 of fast read command code.
Fast Read (byte 2)	0x00–0xFF, **0xF4**	Byte 2 of fast read command code. Applicable when *Enable 2 bytes command code* == Checked.
Dummy Cycle: Read Status Register	**0** – 31	Specifies the number of dummy cycles for read status command.
Dummy Cycle: Read Security Register	**0** – 31	Specifies the number of dummy cycles for read security command.
Dummy Cycle: Fast Read	**0** – 31	Specifies the number of dummy cycles for fast read command.
Flash Status Flags		
Flash Status Busy Value	0 – **1**	Specifies the value of the flash register status bit when it is busy during write, program, or erase operation.
Flash Status Register Busy bit index	**0** – 7	Specifies the bit location index of the flash busy status.
Flash Status Register Program Fail bit index	0 – 7, **4**	Specifies the bit location index of the flash program fail status. Some flash devices may support this status flag. If this is not supported, set to 0.
Flash Status Register Erase Fail bit index	0 – 7, **5**	Specifies the bit location index of the flash erase fail status. Some flash devices may support this status flag. If this is not supported, set to 0.

Attribute	Selectable Values	Description
XiP Mode		
XiP Mode Address Padding	0x00–0xFF, **0x5A**	In Flash controller applications, this byte is inserted after the address to remain in XiP mode. Different flash device vendors may require different patterns to achieve this. Refer to the flash vendor data sheet to determine the required pattern. Applicable when *Enable XiP Mode* == Checked.

4. Signal Description

This section describes the Octal SPI Controller IP ports.

At reset, all output signals are driven low (0) except for specific SPI signals as described in Table 4.1.

Table 4.1. Octal SPI Controller Ports

Port	Type	Description
System Clock and Reset		
clk_i	Input	System clock Maximum frequency is 200 MHz.
rst_n_i	Input	Asynchronous active low reset Minimum reset pulse width is one clock cycle (clk_i). The reset goes through an internal reset synchronizer and may take several clock cycles to propagate. After de-asserting the reset, wait for at least four clock cycles before starting any transaction.
SPI Interface		
spi_sck_o	Output	SPI output clock This clock is generated by the Octal SPI Controller IP. SPI data transfer is synchronous to this clock. The idle/default value is based on the SPI clock polarity attribute and is configurable/programmable.
spi_tgt_rst_n_o	Output	Asynchronous active low reset This is an optional pin to reset the SPI target and is available if selected in the IP GUI. The default state of this port is high (1).
spi_cs_n_o[x:0]	Output	SPI chip select Active low signal for selecting the SPI target. This signal goes low when there is an active transaction or data transfer. The number of chip select signals is based on the number of supported SPI targets (*Max number of SPI Target*) set in the IP GUI. The default state of this port is high (1). **Note:** x = *Max number of SPI Target* – 1
SPI Interface - Internal I/O Primitive[1]		
spi_ds_io	Input/Output	Data strobe This signal is an input to the controller during read data transfers and is driven by the target. This signal is not driven (tri-stated) by the IP core and only serves as a data strobe or clock input for the received xSPI data.
spi_dt_io[7:0]	Input/Output	Bidirectional SPI data signals Controller and target drivers are operating in push-pull mode. The number of data pins depends on the selected maximum number of data lanes in the IP GUI. x8 – Transmit and receive data is 8 bits x4 – Transmit and receive data is 4 bits x2 – Transmit and receive data is 2 bits x1 – Transmit data on data bit[0]; receive data on data bit[1] 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).

Port	Type	Description
SPI Interface - External I/O Primitive[2]		
spi_ds_i	Input	Data strobe input
spi_ds_o	Output	Data strobe output Tied to low.
spi_ds_oe_o	Output	Active high data strobe output enable Tied to low.
spi_dt_i[7:0]	Input	SPI data input
spi_dt_o[7:0]	Output	SPI data output The default state of this port is low (0).
spi_dt_oe_o[7:0]	Output	Active high SPI data output enable The default state of this port is low (0).
Interrupt Interface		
int_o	Output	Level sensitive active high interrupt signal This signal asserts when any of the enabled interrupt status bits is asserted.
AXI4 or AXI4-Lite Interface (Data or CSR)[3, 4]		
Write Address Channel		
s_axi4_awready_o	Output	Ready indicator
s_axi4_awvalid_i	Input	Valid indicator
s_axi4_awid_i[x:0]	Input	Transaction identifier for write channel The width depends on the configured *AXI4 Bus ID Width.* **Note:** x = *AXI4 Bus ID Width* – 1 This is optional if the bus is AXI4-Lite.
s_axi4_awaddr_i[31:0]	Input	Transaction address When accessing the registers or FIFO, the address must be DWORD aligned (bit[1:0] = 0).
s_axi4_awlen_i[7:0]	Input	Transaction length The total number of transactions is encoded as Length = s_axi4_awlen_i + 1 When accessing the registers, s_axi4_awlen_i must be set to 0 (single transfer). Variable length can be used when accessing the FIFO or target address space. This is not available if the bus is AXI4-Lite and length is fixed to 1.
s_axi4_awsize_i[2:0]	Input	Transaction size 2 – 4 bytes per transfer Other values are not supported/reserved. This is not available if the bus is AXI4-Lite.
s_axi4_awburst_i[1:0]	Input	Burst type 0 – Fixed burst 1 – Incrementing burst Other values are not supported/reserved. When accessing the registers or FIFO, s_axi4_awburst_i must be set to 0 (fixed burst transfer). s_axi4_awburst_i = 1 can be used when accessing the target address space. This is not available if the bus is AXI4-Lite.
Write Data Channel		
s_axi4_wready_o	Output	Ready indicator
s_axi4_wvalid_i	Input	Valid indicator
s_axi4_wdata_i[31:0]	Input	Write data
s_axi4_wstrb_i[3:0]	Input	Write data strobes
s_axi4_wlast_i	Input	Last write data This is not available if the bus is AXI4-Lite.

Port	Type	Description
Write Response Channel		
s_axi4_bready_i	Input	Ready indicator
s_axi4_bvalid_o	Output	Valid indicator
s_axi4_bid_o[x:0]	Output	Transaction identifier for write channel **Note:** x = *AXI4 Bus ID Width* – 1 This is optional if the bus is AXI4-Lite.
s_axi4_bresp_o[1:0]	Output	Write response
Read Address Channel		
s_axi4_arready_o	Output	Ready indicator
s_axi4_arvalid_i	Input	Valid indicator
s_axi4_arid_i[x:0]	Input	Transaction identifier for read channel The width depends on the configured *AXI4 Bus ID Width*. **Note:** x = *AXI4 Bus ID Width* – 1 This is optional if the bus is AXI4-Lite.
s_axi4_araddr_i[31:0]	Input	Transaction address When accessing the registers or FIFO, the address must be DWORD aligned (bit[1:0] = 0).
s_axi4_arlen_i[7:0]	Input	Transaction length The total number of transactions is encoded as Length = s_axi4_arlen_i + 1 When accessing the registers, s_axi4_arlen_i must be set to 0 (single transfer). Variable length can be used when accessing the FIFO or target address space. This is not available if the bus is AXI4-Lite and length is fixed to 1.
s_axi4_arsize_i[2:0]	Input	Transaction size 2 – 4 bytes per transfer Other values are not supported/reserved. This is not available if the bus is AXI4-Lite.
s_axi4_arburst_i[1:0]	Input	Burst type 0 – Fixed burst 1 – Incrementing burst Other values are not supported/reserved. When accessing the registers or FIFO, s_axi4_arburst_i must be set to 0 (fixed burst transfer). s_axi4_arburst_i = 1 can be used when accessing the target address space. This is not available if the bus is AXI4-Lite.
Read Data Channel		
s_axi4_rready_i	Input	Ready indicator
s_axi4_rvalid_o	Output	Valid indicator
s_axi4_rid_o[x:0]	Output	Transaction identifier for read channel **Note:** x = *AXI4 Bus ID Width* – 1 This is optional if the bus is AXI4-Lite.
s_axi4_rdata_o[31:0]	Output	Read data
s_axi4_rresp_o[1:0]	Output	Read response
s_axi4_rlast_o	Output	Last read data

Port	Type	Description
AXI4-Lite Interface (CSR) [3] – Applicable only if dual bus interface is selected[5]		
Write Address Channel		
s_axi4_c_awready_o	Output	Ready indicator
s_axi4_c_awvalid_i	Input	Valid indicator
s_axi4_c_awid_i[x:0]	Input	Transaction identifier for write channel The width depends on the configured *AXI4 Bus ID Width*. **Note:** x = *AXI4 Bus ID Width* – 1 Available only if *Enable AXI4-Lite ID* is selected in IP GUI.
s_axi4_c_awaddr_i[31:0]	Input	Transaction address When accessing the registers or FIFO, the address must be DWORD aligned (bit[1:0] = 0).
Write Data Channel		
s_axi4_c_wready_o	Output	Ready indicator
s_axi4_c_wvalid_i	Input	Valid indicator
s_axi4_ c_wdata_i[31:0]	Input	Write data
s_axi4_ c_wstrb_i[3:0]	Input	Write data strobes
Write Response Channel		
s_axi4_ c_bready_i	Input	Ready indicator
s_axi4_ c_bvalid_o	Output	Valid indicator
s_axi4_ c_bid_o[x:0]	Output	Transaction identifier for write channel **Note:** x = *AXI4 Bus ID Width* – 1
s_axi4_ c_bresp_o[1:0]	Output	Write response
Read Address Channel		
s_axi4_ c_arready_o	Output	Ready indicator
s_axi4_ c_arvalid_i	Input	Valid indicator
s_axi4_ c_arid_i[x:0]	Input	Transaction identifier for read channel The width depends on the configured *AXI4 Bus ID Width*. **Note:** x = *AXI4 Bus ID Width* – 1 Available only if *Enable AXI4-Lite ID* is selected in IP GUI.
s_axi4_ c_araddr_i[31:0]	Input	Transaction address When accessing the registers or FIFO, the address must be DWORD aligned (bit[1:0] = 0).
Read Data Channel		
s_axi4_ c_rready_i	Input	Ready indicator
s_axi4_ c_rvalid_o	Output	Valid indicator
s_axi4_ c_rid_o[x:0]	Output	Transaction identifier for read channel **Note:** x = *AXI4 Bus ID Width* – 1 Available only if *Enable AXI4-Lite ID* is selected in IP GUI.
s_axi4_ c_rdata_o[31:0]	Output	Read data
s_axi4_ c_rresp_o[1:0]	Output	Read response

Port	Type	Description
AXI4 Interface (CSR)[3] – Applicable only if dual bus interface is selected[5]		
Write Address Channel		
s_axi4_c_awready_o	Output	Ready indicator
s_axi4_c_awvalid_i	Input	Valid indicator
s_axi4_c_awid_i[x:0]	Input	Transaction identifier for write channel The width depends on the configured *AXI4 Bus ID Width*. **Note:** x = *AXI4 Bus ID Width* – 1
s_axi4_c_awaddr_i[31:0]	Input	Transaction address When accessing the registers or FIFO, the address must be DWORD aligned (bit[1:0] = 0).
s_axi4_c_awlen_i[7:0]	Input	Transaction length The total number of transactions is encoded as Length = s_axi4_awlen_i + 1. When accessing the registers, s_axi4_awlen_i must be set to 0 (single transfer). Variable length can be used when accessing the FIFO.
s_axi4_c_awsize_i[2:0]	Input	Transaction size 2 to 4 bytes per transfer. Other values are reserved or not supported.
s_axi4_c_awburst_i[1:0]	Input	Burst type 0 – Fixed burst 1 – Incrementing burst Other values are reserved or not supported. When accessing the registers or FIFO, s_axi4_awburst_i must be set to 0 (fixed burst transfer).
Write Data Channel		
s_axi4_c_wready_o	Output	Ready indicator
s_axi4_c_wvalid_i	Input	Valid indicator
s_axi4_c_wdata_i[31:0]	Input	Write data
s_axi4_c_wstrb_i[3:0]	Input	Write data strobes
s_axi4_c_wlast_i	Input	Last write data
Write Response Channel		
s_axi4_c_bready_i	Input	Ready indicator
s_axi4_c_bvalid_o	Output	Valid indicator
s_axi4_c_bid_o[x:0]	Output	Transaction identifier for write channel **Note:** x = *AXI4 Bus ID Width* – 1
s_axi4_c_bresp_o[1:0]	Output	Write response
Read Address Channel		
s_axi4_c_arready_o	Output	Ready indicator
s_axi4_c_arvalid_i	Input	Valid indicator
s_axi4_c_arid_i[x:0]	Input	Transaction identifier for read channel The width depends on the configured *AXI4 Bus ID Width*. **Note:** x = *AXI4 Bus ID Width* – 1
s_axi4_c_araddr_i[31:0]	Input	Transaction address When accessing the registers or FIFO, the address must be DWORD aligned (bit[1:0] = 0).
s_axi4_c_arlen_i[7:0]	Input	Transaction length The total number of transactions is encoded as Length = s_axi4_arlen_i + 1. When accessing the registers, s_axi4_arlen_i must be set to 0 (single transfer). Variable length can be used when accessing the FIFO.
s_axi4_c_arsize_i[2:0]	Input	Transaction size 2 to 4 bytes per transfer. Other values are reserved or not supported.

Port	Type	Description
s_axi4_c_arburst_i[1:0]	Input	Burst type 0 – Fixed burst 1 – Incrementing burst Other values are reserved or not supported. When accessing the registers or FIFO, s_axi4_arburst_i must be set to 0 (fixed burst transfer).
Read Data Channel		
s_axi4_c_rready_i	Input	Ready indicator
s_axi4_c_rvalid_o	Output	Valid indicator
s_axi4_c_rid_o[x:0]	Output	Transaction identifier for read channel **Note:** x = *AXI4 Bus ID Width* – 1
s_axi4_c_rdata_o[31:0]	Output	Read data
s_axi4_c_rresp_o[1:0]	Output	Read response
s_axi4_c_rlast_o	Output	Last read data
APB Interface[3] (Data or CSR)		
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.
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 0 – Read 1 – Write
s_apb_pwdata_i [31:0]	Input	APB write data signal
s_apb_prdata_o [31:0]	Output	APB read data signal
s_apb_pready_o	Output	APB ready signal Indicates transfer completion. Completer uses this signal to extend an APB transfer.
s_apb_pslverr_o	Output	APB error signal

Notes:

1. Bidirectional SPI interface is only available when internal I/O primitives are enabled in the IP GUI.
2. External I/O SPI interface is only available when internal I/O primitives are disabled in the IP GUI.
3. Availability of signals depends on the selected interface in the IP GUI.
4. For the single bus interface, AXI4 and AXI4-Lite signals use the same signal names.
5. For the dual bus interface, AXI4 and AXI4-Lite signals for access through the CSR interface are denoted with _c (for example, s_axi4_c_rready_i).

5. Register Description

5.1. Overview

This section defines the configuration, status, and control registers of the Octal SPI Controller IP. The Octal SPI Controller 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 (AXI4, AXI4-Lite, APB).

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.

Table 5.2. Summary of Octal SPI Controller IP Core Registers

Offset Address[1]	Register Name	Description
Configuration Registers		
0x000	Reserved	Reserved address for configuration register
0x004	Octal SPI Configuration 0 Register	Configures SPI settings
0x008	Octal SPI Configuration 1 Register	Configures transaction counters
0x00C	Supported Command Code 0 Register	Vendor specific flash command code
0x010	Supported Command Code 1 Register	Vendor specific flash command code
0x014	Supported Command Code 2 Register	Vendor specific flash command code
0x018	Supported Command Configuration Register	Configures SPI transaction settings
0x02C	Minimum Target Address Alignment Register	Configures address alignment per target
0x030	Target Start Address Offset Register	Configures address offset per target
0x034	Total Target Address Map Alignment Register	Configures total target address alignment
0x038	Target AXI4 Base Address Register	Configures base address for total target address space
0x03C	Interrupt Enable Register	Enables interrupt output
Status Registers		
0x100	Interrupt Status Register	Interrupt event information
0x104	Generic SPI Command Packet Counter Register	Packet counter information
0x108	Supported Flash Command Packet Counter Register	Packet counter information
0x10C	Octal SPI Debug Information 0 Register	Miscellaneous status and information
0x110	Octal SPI Debug Information 1 Register	FIFO data count information
Control Registers		
0x200	Tx FIFO Register	Transmit FIFO mapping
0x204	Rx FIFO Register	Receive FIFO mapping
0x208	Packet Header 0 Register	User packet structure for generic SPI commands
0x20C	Packet Header 1 Register	User packet structure for support flash commands
0x210	Packet Header 2 Register	User packet structure for support flash commands
0x214	Packet Header 3 Register	User packet structure for support flash commands
0x218	Write Data Register	Payload data register
0x21C	Read Data Register	Return data register
0x220	Start Transaction Register	Start IP operation to generate SPI transaction

Offset Address[1]	Register Name	Description
0x224	Interrupt Set Register	Triggers the interrupt status
0x228	Test Mode Register	Control for loopback test
0x22C	Soft Reset Register	Programmable reset control
0x230	IO Delay Control Register	Controls the SPI IO delay

Note:

1. AXI4 starting offset (s_axi4_awaddr_i[9:0] or s_axi4_araddr_i[9:0]).

5.2. Configuration Registers

5.2.1. Octal SPI Configuration 0 Register

Table 5.3. Octal SPI Configuration 0 Register

Field	Name	Description	Access	Default
[31:24]	*reserved*	Reserved	RO	0x0
[23]	non_blocking_rxfifo	When Rx FIFO is empty and there is a read access: 0 – Wait until Rx FIFO is not empty. This may block the bus from accepting a new request. 1 – Do not wait for Rx FIFO status and return any (garbage) data. Applicable only when FIFO is enabled in the IP GUI.	RW	*Default Non-blocking Rx FIFO*
[22]	non_blocking_txfifo	When Tx FIFO is full and there is a write access: 0 – Wait until Tx FIFO is not full. This may block the bus from accepting a new request. 1 – Do not wait for Tx FIFO status and discard the write data. Applicable only when FIFO is enabled in the IP GUI.	RW	*Default Non-blocking Tx FIFO*
[21]	use_ds_in_ddrmode	When set, the read data is captured using the input data strobe generated by the target device. 0 – Use SCK to sample read data 1 – Use data strobe signal from target device to sample read data Applicable only when *Programmable Use DS* == Checked.	RW	*Default Use DS value*
[20]	auto_clr_soft_rst	0 – Do not auto-clear the soft reset. 1 – Auto-clear the soft reset. The soft reset register automatically returns to 0 after write of 1.	RW	0x1
[19]	auto_clr_tx_start	Auto clear the tx_start register 0 – Do not auto-clear tx_start after command is done. 1 – Auto-clear tx_start after a command is done. This is useful when debugging to allow execution of commands one at a time. Applicable only when FIFO is enabled in the IP GUI.	RW	0x0
[18]	*reserved*	Reserved	RO	0x0
[17]	en_addr_space_map	Enables mapping of target device in the bus. When set, the target device must have a valid address space allocation in the bus. 0 – No address mapping for target device 1 – Target device is mapped in the bus address space Applicable only when *Enable Target Address Map* == Checked and *Programmable Memory Map* == Checked.	RW	*Programmable Memory Map*

Field	Name	Description	Access	Default
[16]	endianness	Data endianness format in the bus 0 – Little endian 1 – Big endian	RO	*Default Endianness*
[15:13]	*reserved*	Reserved	RO	0x0
[12:8]	sck_rate	Clock divider value that is used to generate SCK clock (spi_sck_o) 0 – Divide by 1 (SCK is the same as clk_i) 1 – Divide by 2 of clk_i 2 – Divide by 4 of clk_i 3 – Divide by 6 of clk_i … SCK Frequency = clk_i / (2 × sck_rate) When *JEDEC xSPI Support* = Unchecked, the minimum value that can be set on this register is 1 (clock divide by 2).	RW	*SPI Clock (SCK) Pulse width*
[7:3]	*reserved*	Reserved	RO	0x0
[2]	cpol	Clock polarity 0 – Active high clock. Clock is low in idle state. 1 – Active low clock. Clock is high in idle state. Applicable only when *Programmable CPOL* == Checked.	RW	*Default CPOL value*
[1]	cpha	Clock phase setting changes the transmission format in SPI. 0 – Sampling of data occurs at odd edges (1, 3, 5, …, 15) of the SCK clock. 1 – Sampling of data occurs at even edges (2, 4, 6, …, 16) of the SCLK clock. Applicable only when *Programmable CPHA* == Checked.	RW	*Default CPHA value*

Field	Name	Description	Access	Default
[0]	lsbf	LSB first setting configures the way the data is transmitted or received on the SPI interface. 0 – Transmit/Receive MSB first 1 – Transmit/Receive LSB first Applicable only when *Programmable LSBF* == Checked. x1 MSB first in spi_dt_io[0] (receive in spi_dt_io [1]) spi_dt_io [0] -> 7,6,5,4,3,2,1,0 spi_dt_io [1] <- 7,6,5,4,3,2,1,0 x1 LSB first in SPI spi_dt_io [0] (receive in spi_dt_io [1]) spi_dt_io [0] -> 0,1,2,3,4,5,6,7 spi_dt_io [1] <- 0,1,2,3,4,5,6,7 x2 MSB first in SPI spi_dt_io [1:0] spi_dt_io [0] -> 6,4,2,0 spi_dt_io [1] -> 7,5,3,1 x2 LSB first in SPI spi_dt_io [1:0] spi_dt_io [0] -> 0,2,4,6 spi_dt_io [1] -> 1,3,5,7 x4 MSB first in SPI spi_dt_io [3:0] spi_dt_io [0] -> 4,0 spi_dt_io [1] -> 5,1 spi_dt_io [2] -> 6,2 spi_dt_io [3] -> 7,3 x4 LSB first in SPI spi_dt_io [3:0] spi_dt_io [0] -> 0,4 spi_dt_io [1] -> 1,5 spi_dt_io [2] -> 2,6 spi_dt_io [3] -> 3,7 x8 spi_dt_io [7:0] -> data[7:0]	RW	*Default LSBF*

5.2.2.Octal SPI Configuration 1 Register

Applicable only when *Enable Generic Packet Counter* == Checked or *Enable Supported Command Packet Counter* == Checked in the IP GUI.

Table 5.4. Octal SPI Configuration 1 Register

Field	Name	Description	Access	Default
[31:16]	rd_trans_int_thresh	Configures the read transaction count threshold for interrupt. When the read transaction counter is equal to or greater than this number, interrupt status asserts. The actual register bit width depends on the *Packet Counter max count* setting. Unused bits are read-only. For example, if *Packet Counter max count* is set to 16 in the IP GUI, then only bits 20 down to 16 are valid and the upper bits are read-only.	RW	0x1
[15:0]	wr_trans_int_thresh	Configures the write transaction count threshold for interrupt. When the write transaction counter is equal to or greater than this number, interrupt status asserts. The actual register bit width depends on the *Packet Counter max count* setting. Unused bits are read-only. For example, if *Packet Counter max count* is set to 16 in the IP GUI, then only bits 4 down to 0 are valid and the upper bits are read-only.	RW	0x1

5.2.3.Supported Command Code 0 Register

Available only when *Programmable Command Code* == Checked in the IP GUI.

Table 5.5. Supported Command Code 0 Register

Field	Name	Description	Access	Default
[31:16]	*reserved*	Reserved	RO	0x0
[15:8]	wren_b1	The second byte of the write enable command code. Applicable only when *Enable 2 bytes command code* == Checked.	RW	*Write Enable (byte 2)*
[7:0]	wren_b0	Write enable command code	RW	*Write Enable (byte 1)*

5.2.4.Supported Command Code 1 Register

Available only when *Programmable Command Code* == Checked in the IP GUI.

Table 5.6. Supported Command Code 1 Register

Field	Name	Description	Access	Default
[31:24]	rdscur_b1	The second byte of the read security or read flag status register command code. Applicable only when *Enable 2 bytes command code* == Checked in the IP GUI.	RW	*Read Security Register (byte 2)*
[23:16]	rdscur_b0	Read security or read flag status register command code.	RW	*Read Security Register (byte 1)*
[15:8]	rdsr_b1	The second byte of the read status register command code. Applicable only when *Enable 2 bytes command code* == Checked in the IP GUI.	RW	*Read Status Register (byte 2)*
[7:0]	rdsr_b0	Read status register command code.	RW	*Read Status Register (byte 1)*

5.2.5.Supported Command Code 2 Register

Available only when *Programmable Command Code* == Checked in the IP GUI.

Table 5.7. Supported Command Code 2 Register

Field	Name	Description	Access	Default
[31:24]	fast_read_b1	The second byte of the fast read command code. Applicable only when *Enable 2 bytes command code* == Checked in the IP GUI.	RW	Fast *Read (byte 2)*
[23:16]	fast_read_b0	Fast read command code	RW	Fast *Read (byte 1)*
[15:8]	page_program_b1	The second byte of the page program command code. Applicable only when *Enable 2 bytes command code* == Checked in the IP GUI.	RW	*Page Program (byte 2)*
[7:0]	page_program_b0	Page program command code	RW	*Page Program (byte 1)*

5.2.6.Supported Command Configuration Register

This register specifies default transfer settings and dummy cycles. Available only when *Programmable Command Code* == Checked in the IP GUI. These settings are used when the IP core is performing transactions from supported flash command packet or target memory map access (XiP or non-XiP mode).

Table 5.8. Supported Command Configuration Register

Field	Name	Description	Access	Default
[31]	addr_mode_r	Addressing mode to be used when performing XiP or non-XiP target memory mapped access. 0 – 24b address 1 – 32b address Applicable only when *Enable Target Address Map* == Checked and *Programmable Memory Map* == Checked.	RW	0x1
[30]	en_2byte_fcc	Indicates the number of bytes of command code to be sent when in STR mode. In DTR mode, the command is 2 bytes by default and is not affected by this setting. 0 – 1-byte command 1 – 2-byte command Applicable only when *Enable 2 bytes command code* == Checked.	RW	0x0
[29]	en_sr_addr	Indicates if an address is necessary when sending read status command or read security command. 0 – No address 1 – Include 4-byte address 0x0000_0000	RW	0x0
[28:24]	fast_read_dummy	Default number of dummy cycles required for fast read command.	RW	0x0
[23:21]	*reserved*	Reserved	RO	0x0
[20:16]	rdscur_dummy	Default number of dummy cycles required for read security register (or read flag status register) command.	RW	0x0
[15:13]	*reserved*	Reserved	RO	0x0
[12:8]	rdsr_dummy	Default number of dummy cycles required for read status register command.	RW	0x0

Field	Name	Description	Access	Default
[7]	wait_ds_r	Applicable only in DTR when there is a data strobe. 0 – No data strobe. Read data will be sampled after the specified dummy cycle or wait state. 1 – Read data arrives along with data strobe. The number of dummy cycles is irrelevant. The Octal SPI Controller IP continuously provides a clock until all the requested data are retrieved.	RW	0x0
[6:4]	xfer_rate_r	Transfer rate setting specifies the transfer rate to be used for each phase: Command (xfer_rate[2]), Address (xfer_rate[1]), and Data (xfer_rate[0]). DTR is currently only valid in x8 width. 0 – STR 1 – DTR	RW	0x0
[3:0]	lane_width_r	SPI I/O data width setting indicates the number of lanes to use in each phase (Command, Address, and Data). 4'h0 – (x1, x1, x1) 4'h1 – (x2, x2, x2) 4'h2 – (x4, x4, x4) 4'h3 – (x8, x8, x8) 4'h4 – (x1, x1, x2) 4'h5 – (x1, x2, x2) 4'h6 – 4'h7 – reserved 4'h8 – (x1, x1, x4) 4'h9 – (x1, x4, x4) 4'hA – 4'hB – reserved 4'hC – (x1, x1, x8) 4'hD – (x1, x8, x8) 4'hE – 4'hF – reserved	RW	0x0

5.2.7. Minimum Target Address Alignment Register

Applicable only when *Enable Target Address Map* == Checked and *Programmable Memory Map* == Checked in the IP GUI.

Table 5.9. Minimum Target Address Alignment Register

Field	Name	Description	Access	Default
[31:10]	min_tgt_addr_align	This setting identifies the target address alignment based on the size of the target memory map. The lower bits that are within the memory map size value are set to 0 while the upper bits are set to 1. The minimum size should be 1 KiB. For example, if the memory map size is 1 KiB (0x400), then the lower 10 bits ([9:0]) correspond to the offset and the upper bits [31:10] are the base address. Thus, the min_tgt_addr_align value is 0xFFFF_FC00.	RW	Calculated based on the attribute value *Target Memory Map Size*.
[9:0]	min_tgt_addr_align		RO	0x0

5.2.8.Target Start Address Offset Register

Applicable only when *Enable Target Address Map* == Checked and *Programmable Memory Map* == Checked in the IP GUI.

Table 5.10. Target Start Address Offset Register

Field	Name	Description	Access	Default
[31:10]	tgt_start_addr	This is the actual address offset of the target memory map. The minimum address offset alignment should be 1 KiB.	RW	*Target Actual Start Address*
[9:0]	tgt_start_addr		RO	0x0

5.2.9.Total Target Address Map Alignment Register

Applicable only when *Enable Target Address Map* == Checked and *Programmable Memory Map* == Checked in the IP GUI.

Table 5.11. Total Target Address Map Alignment Register

Field	Name	Description	Access	Default
[31:10]	total_addr_map_align	This setting identifies the target address alignment in the AXI4 bus based on the total size of the memory map. For single target only, it has the same value as the minimum target address alignment. The lower bits that are within the memory map size value are set to 0 while the upper bits are set to 1. The minimum size should be 1 KiB. For example, if the total memory map size is 64 KiB (0x10000), then the lower 16 bits ([15:0]) correspond to the offset and the upper bits [31:16] are the base address. Thus, the total_addr_map_align value is 0xFFFF_0000.	RW	Calculated based on the attribute value *Total Target Memory Map Size*.
[9:0]	total_addr_map_align		RO	0x0

5.2.10. Target AXI4 Base Address Register

Applicable only when *Enable Target Address Map* == Checked and *Programmable Memory Map* == Checked in the IP GUI.

Table 5.12. Target AXI4 Base Address Register

Field	Name	Description	Access	Default
[31:10]	tgt_axi_bar	This is the AXI4 base address of the target memory map. The minimum address offset alignment should be 1 KiB.	RW	*Target Map: AXI4 Base Address*
[9:0]	tgt_axi_bar		RO	0x0

5.2.11. Interrupt Enable Register

Each interrupt enable bit is applicable only when the corresponding interrupt status is applicable for the device configuration.

Table 5.13. Interrupt Enable Register

Field	Name	Description	Access	Default
[31:18]	reserved	Reserved	RO	0x0
[17]	flash_program_fail	See description for bit[13:0].	RW	0x0
[16]	flash_erase_fail		RW	0x0
[15:14]	reserved	Reserved	RO	0x0
[13]	rd_on_empty_error	When set to high, enables the corresponding interrupt	RW	0x0

Field	Name	Description	Access	Default
[12]	wr_on_full_error	status signal to cause the assertion of the interrupt port signal (int_o). Refer to the Interrupt Status Register section for more information.	RW	0x0
[11]	bus_access_error		RW	0x0
[10]	user_pkt_decode_error		RW	0x0
[9]	sup_rd_trans_cnt_hit		RW	0x0
[8]	sup_wr_trans_cnt_hit		RW	0x0
[7]	gen_rd_trans_cnt_hit		RW	0x0
[6]	gen_wr_trans_cnt_hit		RW	0x0
[5]	rdata_available		RW	0x0
[4]	wdata_done		RW	0x0
[3]	rx_fifo_not_empty		RW	0x0
[2]	rx_fifo_full		RW	0x0
[1]	tx_fifo_empty		RW	0x0
[0]	tx_fifo_full		RW	0x0

5.3. Status Registers

5.3.1. Interrupt Status Register

Each interrupt status bit, when set to high, causes the assertion of the interrupt port (int_o) if the corresponding interrupt enable bit is high.

Table 5.14. Interrupt Status Register

Field	Name	Description	Access	Default
[31:18]	reserved	Reserved	RO	0x0
[17]	flash_program_fail	When set to high, indicates that the last flash program operation failed. Applicable only when *Enable Command Processing* == Checked in the IP GUI.	RW1C	0x0
[16]	flash_erase_fail	When set to high, indicates that the last flash erase operation failed. Applicable only when *Enable Command Processing* == Checked in the IP GUI.	RW1C	0x0
[15:14]	reserved	Reserved	RO	0x0
[13]	rd_on_empty_error	• When FIFO is enabled in the IP GUI: When set to high, indicates that a read access was done on the Rx FIFO while it is empty. The return data is invalid or garbage. • When FIFO is not enabled in the IP GUI: When set to high, indicates that a read access was done on the read data register while it is empty (rdata_available = 0). The return data may be invalid or garbage.	RW1C	0x0
[12]	wr_on_full_error	• When FIFO is enabled in the IP GUI: When set to high, indicates that a write access was done on the Tx FIFO while it is full. This may cause a loss of data and an unexpected result if the packet is broken. • When FIFO is not enabled in IP GUI: When set to high, indicates that a write access was done on the write data register while it is not empty (wdata_not_empty = 1). This may cause	RW1C	0x0

Field	Name	Description	Access	Default
		data corruption during SPI transfer.		
[11]	bus_access_error	When set to high, indicates that a request was received that hits a reserved or unallocated address region.	RW1C	0x0
[10]	user_pkt_decode_error	When set to high, indicates that an error was encountered while decoding the user packet. This could be a result of inconsistent bit setting or unsupported command in the packet header. This may cause the IP core to stop or an unexpected result. To recover, clean up the FIFOs or issue a soft reset.	RW1C	0x0
[9]	sup_rd_trans_cnt_hit	When set to high, indicates that rd_xfer_done_count value of supported flash command packet counter register is greater than or equal to the set threshold rd_trans_int_thresh. Applicable only when both *Enable Command Processing* == Checked and *Enable Supported Command Packet Counter* == Checked in the IP GUI.	RW1C	0x0
[8]	sup_wr_trans_cnt_hit	When set to high, indicates that wr_xfer_done_count value of supported flash command packet counter register is greater than or equal to the set threshold wr_trans_int_thresh. Applicable only when both *Enable Command Processing* == Checked and *Enable Supported Command Packet Counter* == Checked in the IP GUI.	RW1C	0x0
[7]	gen_rd_trans_cnt_hit	When set to high, indicates that rd_xfer_done_count value of generic SPI command packet counter register is greater than or equal to the set threshold rd_trans_int_thresh. Applicable only when *Enable Generic Packet Counter* == Checked in the IP GUI.	RW1C	0x0
[6]	gen_wr_trans_cnt_hit	When set to high, indicates that wr_xfer_done_count value of generic SPI command packet counter register is greater than or equal to the set threshold wr_trans_int_thresh. Applicable only when *Enable Generic Packet Counter* == Checked in the IP GUI.	RW1C	0x0
[5]	rdata_available	When set to high, indicates that read data register has a valid data. Applicable only when FIFO is not enabled in the IP GUI.	RW1C	0x0
[4]	wdata_done	When set to high, indicates that the write data register has been taken to be transmitted and is now available for write. Applicable only when FIFO is not enabled in the IP GUI.	RW1C	0x0
[3]	rx_fifo_not_empty	When set to high, indicates that the Rx FIFO is not empty. Applicable only when FIFO is enabled in the IP GUI.	RW1C	0x0
[2]	rx_fifo_full	When set to high, indicates that the Rx FIFO is full. Applicable only when FIFO is enabled in the IP GUI.	RW1C	0x0
[1]	tx_fifo_empty	When set to high, indicates that the last data has been taken from the Tx FIFO and it is now empty. Note that this only asserts when the Tx FIFO status transitions from “not empty” to “empty”. Applicable only when FIFO is enabled in the IP GUI.	RW1C	0x0
[0]	tx_fifo_full	When set to high, indicates that the Tx FIFO is full. Applicable only when FIFO is enabled in the IP GUI.	RW1C	0x0

5.3.2. Generic SPI Command Packet Counter Register

Applicable only when *Enable Generic Packet Counter* == Checked in the IP GUI. The actual counter width depends on the *Packet Counter max count* setting.

Table 5.15. Generic SPI Command Packet Counter Register

Field	Name		Access	Default
[31:16]	rd_xfer_done_count	This field shows the number of completed generic packet for read access. Writing 1 to bit[16] resets this register to 0; writing to other bits is inconsequential.	RW	0x0
[15:0]	wr_xfer_done_count	This field shows the number of completed generic packet for write access. Writing 1 to bit[0] resets this register to 0; writing to other bits is inconsequential.	RW	0x0

5.3.3.Supported Flash Command Packet Counter Register

Applicable only when both *Enable Command Processing* == Checked and *Enable Supported Command Packet Counter* == Checked in the IP GUI. The actual counter width depends on the *Packet Counter max count* setting.

Table 5.16. Supported Flash Command Packet Counter Register

Field	Name	Description	Access	Default
[31:16]	rd_xfer_done_count	This field shows the number of completed supported flash packet for read access. Writing 1 to bit[16] resets this register to 0; writing to other bits is inconsequential.	RW	0x0
[15:0]	wr_xfer_done_count	This field shows the number of completed supported flash packet for write access. Writing 1 to bit[0] resets this register to 0; writing to other bits is inconsequential.	RW	0x0

5.3.4. Octal SPI Debug Information 0 Register

Table 5.17. Octal SPI Debug Information 0 Register

Field	Name	Description	Access	Default
[31:4]	reserved	Reserved	RO	0x0
[3]	spi_has_started	SPI transaction has started since the last read on this register. This register asserts when SPI chip select has been asserted. Reading this register automatically clears the value but re-triggers if chip select is still asserted. This register can be used along with spi_busy to check if a recent transaction is done. 0 – SPI chip select not asserted 1 – SPI chip select was asserted	RO	0x0
[2]	wdata_not_empty	Status of write data register When high, indicates that user has written something to the write data register and it has not been taken yet. Applicable only when *Include FIFO* is not selected in the IP GUI.	RO	0x0
[1]	spi_on_hold	SPI on-hold transfer status 0 – Normal operation 1 – When there is an ongoing transfer (spi_busy = 1) and SCK has to be stopped temporarily while waiting for user action or waiting for data to be available in the FIFO.	RO	0x0

Field	Name	Description	Access	Default
[0]	spi_busy	SPI busy status 0 – Idle state 1 – There is an ongoing transaction	RO	0x0

5.3.5. Octal SPI Debug Information 1 Register

Table 5.18. Octal SPI Debug Information 1 Register

Field	Name	Description	Access	Default
[31:16]	rx_fifo_avail_count	Rx FIFO data available counter This field shows the number of data that is currently available in the Rx FIFO. Applicable only when *Enable Rx FIFO available data count* == Checked in the IP GUI.	RO	0x0
[15:0]	tx_fifo_free_count	Tx FIFO free space counter This field shows the number of remaining space available in the Tx FIFO. Applicable only when *Enable Tx FIFO free space count* == Checked in the IP GUI.	RO	*FIFO depth*

5.4. Control Registers

5.4.1. Tx FIFO Register

This is a dummy register used to fill data to the Tx FIFO. Applicable only when FIFO is enabled in the IP GUI.

Note: When the dual bus interface is selected and *FIFO Interface Mapping* is set to **Map to Data Interface**, this register is accessible through the target data interface and not the CSR interface.

Table 5.19. Tx FIFO Register

Field	Name	Description	Access	Default
[31:0]	tx_fifo_data	Use this dummy register to send packet frames to the Tx FIFO through repeated write access to this register.	WO	0x0

5.4.2. Rx FIFO Register

This is a dummy register used to get read data from the Rx FIFO. Read data from the response to read commands is stored in the Rx FIFO. Applicable only when FIFO is enabled in the IP GUI.

Note: When the dual bus interface is selected and *FIFO Interface Mapping* is set to **Map to Data Interface**, this register is accessible through the target data interface and not the CSR interface.

Table 5.20. Rx FIFO Register

Field	Name	Description	Access	Default
[31:0]	rx_fifo_data	Use this dummy register to get the received data from the Rx FIFO through repeated read access to this register. Check the interrupt status register (bit[3]) to determine if Rx FIFO is not empty. When reading data that is not multiples of 4 bytes, the invalid bytes should be ignored (for example, 3-byte data: bit[23:0] – valid bytes; bit[31:24] – invalid byte).	RO	0x0

5.4.3. Packet Header 0 – 3 Registers

Applicable only when FIFO is not enabled in the IP GUI.

Table 5.21. Packet Header 0 – 3 Registers

Field	Name	Description	Access	Default
[31:0]	pkt_header_0	Refer to the User Packets section for the packet formats and field definitions. If *Target Address up to 64b* = Unchecked in the IP GUI, then pkt_header_3 is reserved. If *Enable Command Processing* = Unchecked in the IP GUI, then only pkt_header_0 is available and the others (pkt_header_1 to pkt_header_3) are reserved.	RW	0x0
[31:0]	pkt_header_1		RW	0x0
[31:0]	pkt_header_2		RW	0x0
[31:0]	pkt_header_3		RW	0x0

5.4.4. Write Data Register

Applicable only when FIFO is not enabled in the IP GUI.

Table 5.22. Write Data Register

Field	Name	Description	Access	Default
[31:0]	write_data	This is the write data or payload of the packet. For commands with more than 1 DWORD payload, check the interrupt status wdata_done or wdata_not_empty status before writing again to this register.	RW	0x0

5.4.5. Read Data Register

Applicable only when FIFO is not enabled in the IP GUI.

Table 5.23. Read Data Register

Field	Name	Description	Access	Default
[31:0]	read_data	This is the read data from the response to the read command. Check the interrupt status rdata_available before reading this register to get a valid data. When reading data that is not multiples of 4 bytes, the invalid bytes should be ignored (for example, 3-byte data: bits[23:0] – valid bytes; [31:24] – invalid byte).	RO	0x0

5.4.6. Start Transaction Register

Table 5.24. Start Transaction Register

Field	Name	Description	Access	Default
[31:2]	reserved	Reserved	RO	0x0
[1]	enter_xip_mode	This is applicable in flash controller application and if the target device supports XiP mode. When set to high, indicates target address space access using XiP mode. Before setting this to high, ensure that the target device has already been enabled for XiP mode (a command has been issued previously to enable XiP mode). Otherwise, data may be invalid. When set to low, indicates exit XiP mode sequence is done. Available only when *Enable XiP Mode* == Checked in the IP GUI.	RW	0x0

Field	Name	Description	Access	Default
[0]	tx_start	When set to high, triggers the IP to start processing the commands from the Tx FIFO or packet header register (if FIFO is not enabled). The spi_busy register (0x10C) asserts when transaction starts. • Without FIFO: This register auto-clears when a command packet is transmitted. • With FIFO: This register auto-clears when a command is done and the configuration register auto_clr_tx_start is set to 1. **Note:** If a command is done and the Tx FIFO becomes empty, tx_start auto-clears even if auto_clr_tx_start is set to 0. This means that when a new command is sent to the Tx FIFO, 1 must be written to this register to trigger the start of transaction again. This is useful when debugging to halt the process after each command.	RW	0x0

5.4.7. Interrupt Set Register

This is a dummy register used to test the assertion of interrupt status. Each interrupt set bit is applicable only when the corresponding interrupt status is applicable for the device configuration

Table 5.25. Interrupt Set Register

Field	Name	Description	Access	Default
[31:18]	reserved	Reserved	RO	0x0
[17]	flash_program_fail	See description for bit[13:0].	WO	0x0
[16]	flash_erase_fail		WO	0x0
[15:14]	reserved	Reserved	RO	0x0
[13]	rd_on_empty_error	When set to high, triggers the corresponding interrupt status signal. Refer to the Interrupt Status Register section for more information.	WO	0x0
[12]	wr_on_full_error		WO	0x0
[11]	bus_access_error		WO	0x0
[10]	user_pkt_decode_error		WO	0x0
[9]	sup_rd_trans_cnt_hit		WO	0x0
[8]	sup_wr_trans_cnt_hit		WO	0x0
[7]	gen_rd_trans_cnt_hit		WO	0x0
[6]	gen_wr_trans_cnt_hit		WO	0x0
[5]	rdata_available		WO	0x0
[4]	wdata_done		WO	0x0
[3]	rx_fifo_not_empty		WO	0x0
[2]	rx_fifo_full		WO	0x0
[1]	tx_fifo_empty		WO	0x0
[0]	tx_fifo_full		WO	0x0

5.4.8.Test Mode Register

Table 5.26. Test Mode Register

Field	Name	Description	Access	Default
[31:1]	reserved	Reserved	RO	0x0
[0]	en_loopback	This is used for debugging or testing. 0 – Loopback is disabled. 1 – Loopback is enabled. Any data transmitted or shifted out returns as received data.	RW	0x0

5.4.9.Soft Reset Register

The behavior of soft reset is affected by the configuration register auto_clr_soft_rst. When auto_clr_soft_rst is high, soft reset automatically returns to 0 after 1 is written. When auto_clr_soft_rst is low, the soft reset register retains the previous value unless a new value is written. The ip_core_rst and ip_csr_rst bits are always auto cleared to avoid deadlock. After performing reset, wait at least four clock cycles before performing any other transactions.

Table 5.27. Soft Reset Register

Field	Name	Description	Access	Deafult
[31:5]	reserved	Reserved	RO	0x0
[4]	spi_tgt_rst	When set to high, resets the target by asserting the output reset pin (spi_tgt_rst_n_o). Applicable only when *Target Reset Pin* == Checked in the IP GUI.	RW[1] RW1C[2]	0x0
[3]	rx_fifo_rst	When set to high, resets the Rx FIFO. Applicable only when FIFO is enabled in the IP GUI.	RW[1] RW1C[2]	0x0
[2]	tx_fifo_rst	When set to high, resets the Tx FIFO. Applicable only when FIFO is enabled in the IP GUI.	RW[1] RW1C[2]	0x0
[1]	ip_csr_rst	When set to high, resets the Octal SPI IP core register block.	RW1C	0x0
[0]	ip_core_rst	When set to high, resets the Octal SPI IP core excluding the registers and FIFO.	RW1C	0x0

Notes:

1. When auto_clr_soft_rst = 0.
2. When auto_clr_soft_rst = 1.

5.4.10. IO Delay Control Register

This register is recommended for advanced users. Applicable only when *Include IO Primitive* == Checked and *Enable Double Transfer Rate* == Checked in the IP GUI. When *Use xSPI IO Primitive* == Checked, this is used to adjust the timing of the following SPI I/O signals: spi_ds_io, spi_dt_io, and spi_sck_o. When *Use xSPI IO Primitive* = Unchecked, this is only used for spi_ds_io delay adjustment.

Table 5.28. IO Delay Control Register

Field	Name	Description	Access	Default
[31:22]	reserved	Reserved	RO	0x0
[21]	delay_control_type	Indicates the operation to perform which is either to toggle the load signal or move signal of the delay controller. 0 – Load; update the delay element in tandem 1 – Move; increments the delay one step at a time	RW	0x0
[20]	delay_control_direction	Indicates the direction of delay adjustment. Applicable only if the type is “move”. 0 – Increment the delay by delay_control_step 1 – Decrement the delay by delay_control_step	RW	0x0
[19:17]	delay_control_select	When set to high, enables a particular delay control element. It is possible to enable all delay control elements at the same time to adjust them with the same parameters. [0] – spi_dt_io delay [1] – spi_ds_io delay [2] – spi_sck_o delay	RW	0x0
[16]	delay_control_request	When set to high, the delay control updates or adjusts the delay according to the parameters (such as step and direction) provided in this register. This field automatically clears once the request has been done.	RW	0x0
[15]	reserved	Reserved	RO	0x0
[14:12]	overflow_flag	When set to high, this flag indicates overflow of the delay counter. Overflow depends on the direction. For example, if the delay_control_direction is 0 (increment), overflow means beyond the maximum delay count. Otherwise, if the delay_control_direction is 1 (decrement), overflow means below the minimum delay count. [0] – overflow flag for spi_dt_io delay [1] – overflow flag for spi_ds_io delay [2] – overflow flag for spi_sck_o delay	RO	0x0
[11:9]	reserved	Reserved	RO	0x0
[8:0]	delay_control_step	Indicates the number of delay steps. One delay step may range from 10 ps to 18 ps depending on PVT (process, voltage, temperature).	RW	0x0

6. Example Design

This section shows an example of a system design used to validate the Octal SPI Controller IP. The system design is implemented in the Lattice Avant-X Versa board with the LAV-AT-X70ES device.

The Octal SPI Controller example design allows you to compile, simulate, and test the Octal SPI Controller IP on the following Lattice evaluation boards:

- Avant-X Versa Board
- Avant-E Evaluation Board
- CertusPro-NX Versa Board

6.1. Example Design Configuration

Table 6.1. Octal SPI Controller IP Configuration Supported by the Example Design

Octal SPI Controller IP GUI Attribute	Octal SPI Controller IP Configuration
Configuration Preset	
SPI Protocol	Custom
Clock Configuration	
System Clock Frequency (MHz)	100
SPI Clock (SCK) Pulse width	1
User Interface	
Interface	AXI4
Number of outstanding request	2
AXI4 Bus ID Width	1
FIFO depth	256
FIFO Implementation	HARD IP
SPI Configuration	
Default LSBF value	0
Default CPOL value	0
Default CPHA value	0
Default Endianness value	0
Default Use DS value	1
Default Non-blocking Tx FIFO	0
Default Non-blocking Rx FIFO	0
Minimum Timing (number of SPI clock): CS assert to SCK	2
Minimum Timing (number of SPI clock): SCK to CS deassert	2
Minimum Timing (number of SPI clock): CS deassert to CS assert	2
Supported Capabilities	
Generic SPI Controller	Checked
Enable Target Address Map	Checked
Enable XiP Mode	Checked
Enable Command Processing	Checked
Enable Double Transfer Rate	Checked
Max number of data lanes	X8
Max number of SPI Target	1
Include FIFO	Checked
Programmable Command Code	Checked
Programmable Memory Map	Checked
Programmable Non-blocking FIFO setting	Checked

Octal SPI Controller IP GUI Attribute	Octal SPI Controller IP Configuration
IO Primitive	
Include IO Primitive	Checked
Memory Map Configuration	
Register Space Size (KiB)	1
Enable Full CSR Address Decoding	Unchecked
Register Map: AXI4 Base Address (32b Hex)	00000000
Target Memory Map Size (KiB)	65536
Target Actual Start Address (32b Hex)	00000000
Total Target Memory Map Size (KiB)	65536
Target Map: AXI4 Base Address (32b Hex)	04000000
Optional Debug Registers	
Enable Tx FIFO free space count	Checked
Enable Rx FIFO available data count	Checked
Enable Generic Packet Counter	Checked
Enable Supported Command Packet Counter	Checked
Packet Counter max count	16
Vendor Specific Flash Command Code	
Enable 2 bytes command code	Checked
Write Enable (byte 1)	0x06
Write Enable (byte 2)	0xF9
Read Status Register (byte 1)	0x05
Read Status Register (byte 2)	0xFA
Read Security Register (byte 1)	0x70
Read Security Register (byte 2)	0x8F
Page Program (byte 1)	0x02
Page Program (byte 2)	0xFD
Fast Read (byte 1)	0x0B
Fast Read (byte 2)	0xF4
Dummy Cycle: Read Status Register	0
Dummy Cycle: Read Security Register	0
Dummy Cycle: Fast Read	0
Flash Status Flags	
Flash Status Busy Value	1
Flash Status Register Busy bit index	0
Flash Status Register Program Fail bit index	4
Flash Status Register Erase Fail bit index	5

6.2. Example Design Components

This example design utilizes the “Hello World” design template that is available in the Lattice Propel™ Builder.

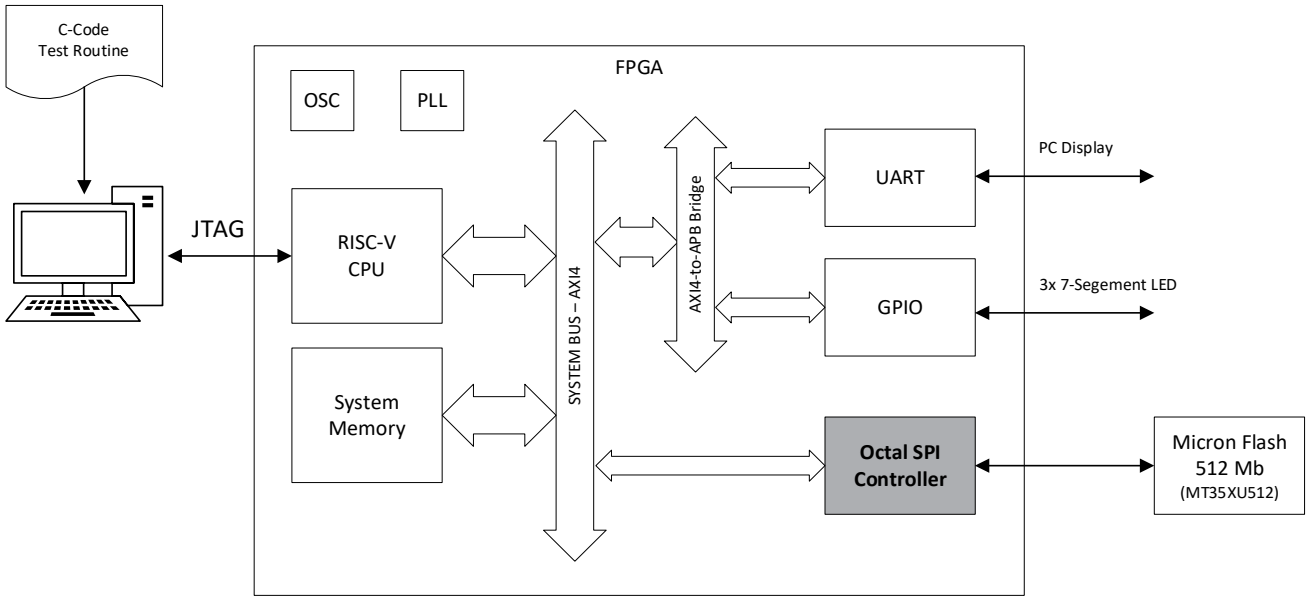


Figure 6.1. Octal SPI Controller Example Design Block Diagram

The Octal SPI Controller example design includes the following blocks:

- RISC-V RX CPU – Embedded processor
- System Memory and TCM – A shared memory for instruction and data
- OSC and PLL – Provides the system clock (100 MHz)
- AXI4 System Bus – Bus interconnect
- AXI4 to APB Bridge – Converts AXI4 to APB interface
- GPIO – Used to drive the 7-segment LED display
- UART – Used for printing debug messages
- Octal SPI Controller – Instance of the design under test (DUT)
- Micron Flash – Non-volatile SPI flash memory that is included in the Avant-X Versa board

6.3. Generating the Example Design

The Lattice Propel Builder and Propel SDK can be used to create an embedded system which consists of a system-on-chip (SoC) design with an embedded processor and system software. The following sections describe the procedures for generating an example design for the Octal SPI Controller IP. A workspace is generated on Propel SDK. The SoC project is then created in Propel Builder and imported to the Lattice Radiant software to be synthesized. For complete step-by-step instructions, refer to A Step-By-Step Approach to Lattice Propel (FPGA-AN-02052) or the Lattice Propel Builder User Guide (FPGA-UG-02212) and Lattice Propel SDK User Guide (FPGA-UG-02211).

6.3.1. Creating the Octal SPI Controller SoC Project

1. Launch Propel SDK. The **Lattice Propel Launcher** opens as shown in Figure 6.2. Browse to a directory for the **Workspace** field and click **Launch**.

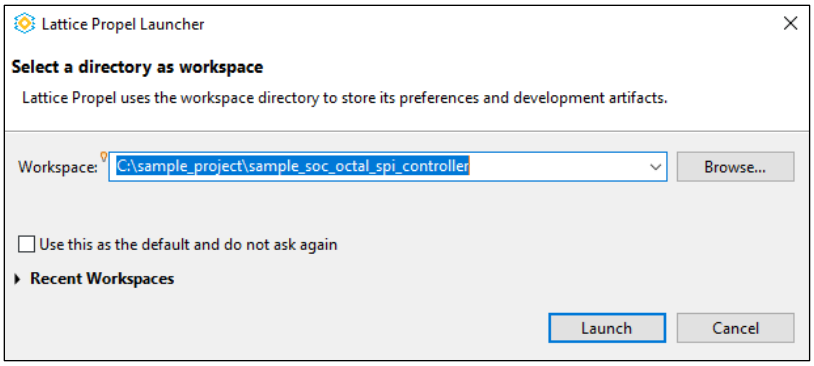


Figure 6.2. Lattice Propel Launcher

2. Create a new SoC design project by selecting **File > New > Lattice SoC Design Project**. The **Create SoC Project** window opens as shown in Figure 6.3.

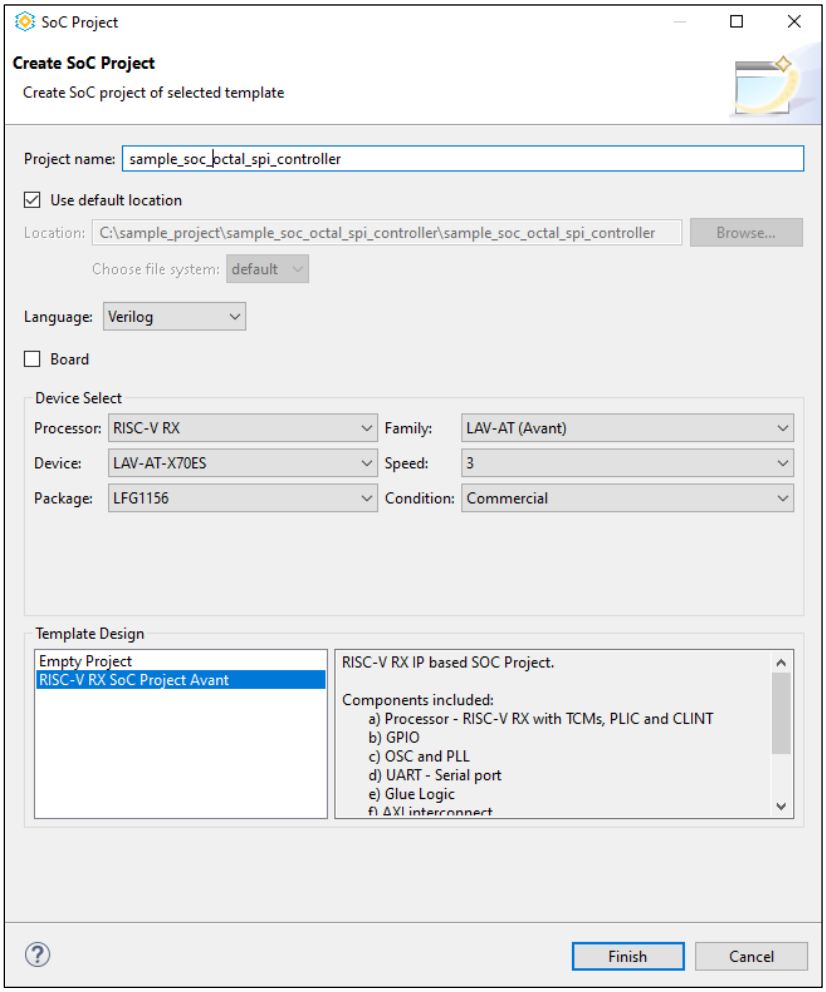


Figure 6.3. Create SoC Project

3. Enter a name for the SoC project in the **Project name** field. Customize the options under **Device Select** using drop-down menus or configure them through the **Board** option. Then, select the template design from the **Template Design** list. For this example design, RISC-V RX processor, LAV-AT-X70ES device, LFG1156 package, and RISC-V RX SoC Project Avant are used.

4. Click **Finish**.
5. Select the generated project, then run Propel Builder by clicking the icon or select **LatticeTools > Open Design in Propel Builder.** Propel Builder opens and loads the design template.
6. In the **IP Catalog** tab in the Propel Builder window, double-click on the Octal SPI Controller IP and generate the IP configuration as shown in Table 6.1. This process is similar to that covered in the Generating and Instantiating the IP section.
7. After generating the IP, the **Define Instance** window opens as shown in Figure 6.4. Modify the instance name if needed, then click **OK**.

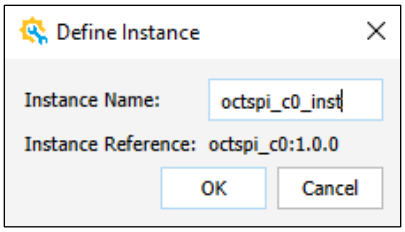


Figure 6.4. Define Instance

8. Connect the instantiated IP to the system. Modify the system component settings from default as follows:
   - RISC-V RX CPU Number of User Interrupt Requests to 3.
   - Update the AXI4 interconnect and AXI4 to APB bridge instances. If an error message is flagged during regeneration, select the instances, then delete and replace these instances with the latest IP versions available on the IP server.
   - Modify the AXI4 interconnect to have three AXI4 subordinates.
   - Connect the AXI4 interface of the Octal SPI Controller to one of the AXI4 interconnect interface. Connect the system clock and the reset port. Connect the interrupt port to the RISC-V RX CPU. Right-click on the SPI interface and select **Export** to export the interface to FPGA pins.
9. View the memory space of the project by selecting the **Address** tab as shown in Figure 6.5. Enable **Lock** for the address space settings so that addresses cannot be manually or automatically updated.

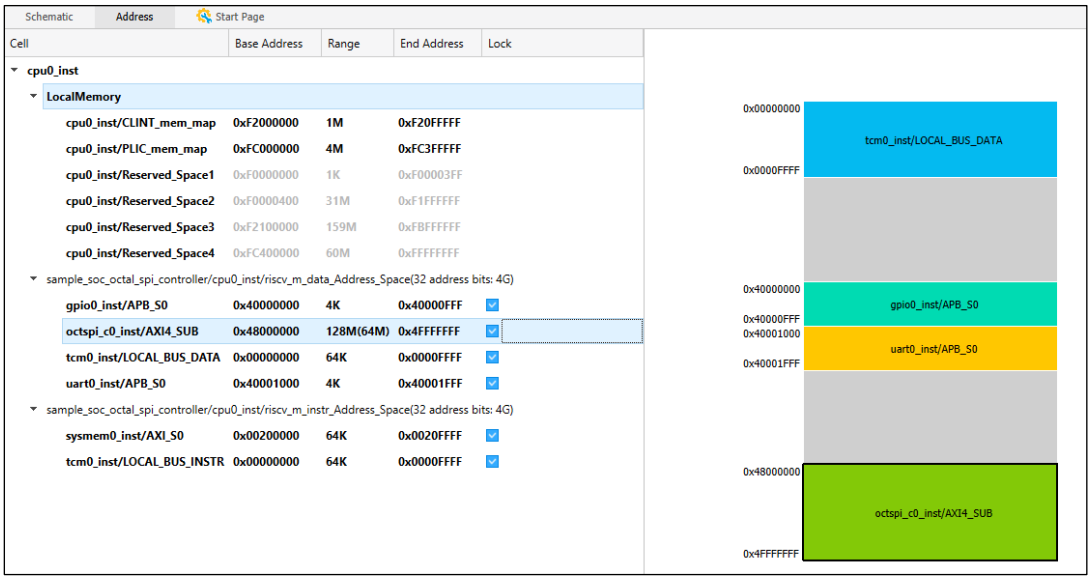


Figure 6.5. Address Tab

10. Click the icon to perform design rule checking (DRC).
11. Save the SoC project.

6.3.2. Importing the Design to Radiant Software

1. Click the icon or select **Design > Run Radiant** to launch the Lattice Radiant Software.
2. Create the IP constraint file. On the pdc file, connect the SPI ports of the IP to the on-board flash pins. Set sysCONFIG MASTER_SPI_PORT to DISABLE to use the on-board flash as the target device.
3. Run the Synthesis, Map, and Place and Route Flow to generate the bitstream programming file.

6.3.3. Creating a C/C++ Project

1. In the Lattice Propel software, build your SoC project to generate the system environment needed for the embedded C/C++ project by selecting the generated SoC project followed by **Project > Build Project**.
2. Check the build result from the **Console** view as shown in Figure 6.6.

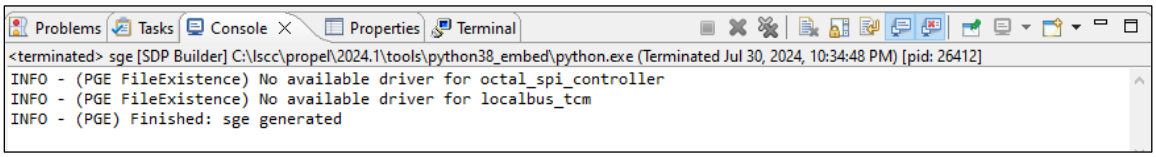


Figure 6.6. Build SoC Project Result

3. Generate a new Lattice C/C++ project by selecting **File > New > Lattice C/C++ Project**. The **C/C++ Project Load System and BSP** window opens as shown in Figure 6.7. For this example design, select the Hello World Project template.

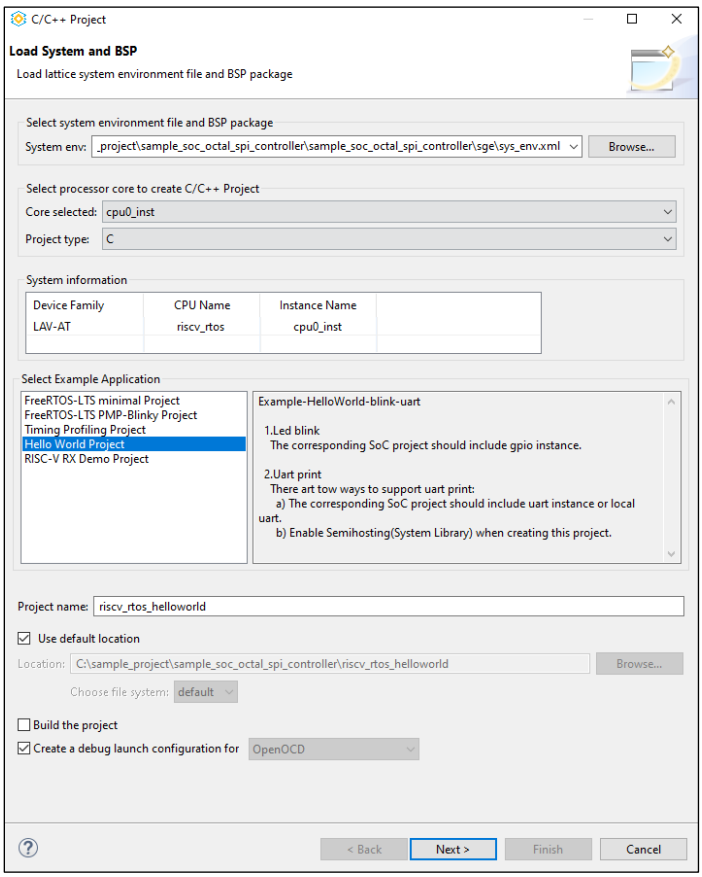


Figure 6.7. C/C++ Project Load System and BSP

4. Update the **Project name** field, then click **Next** followed by **Finish**. The C/C++ Hello World project template is as shown in Figure 6.8.

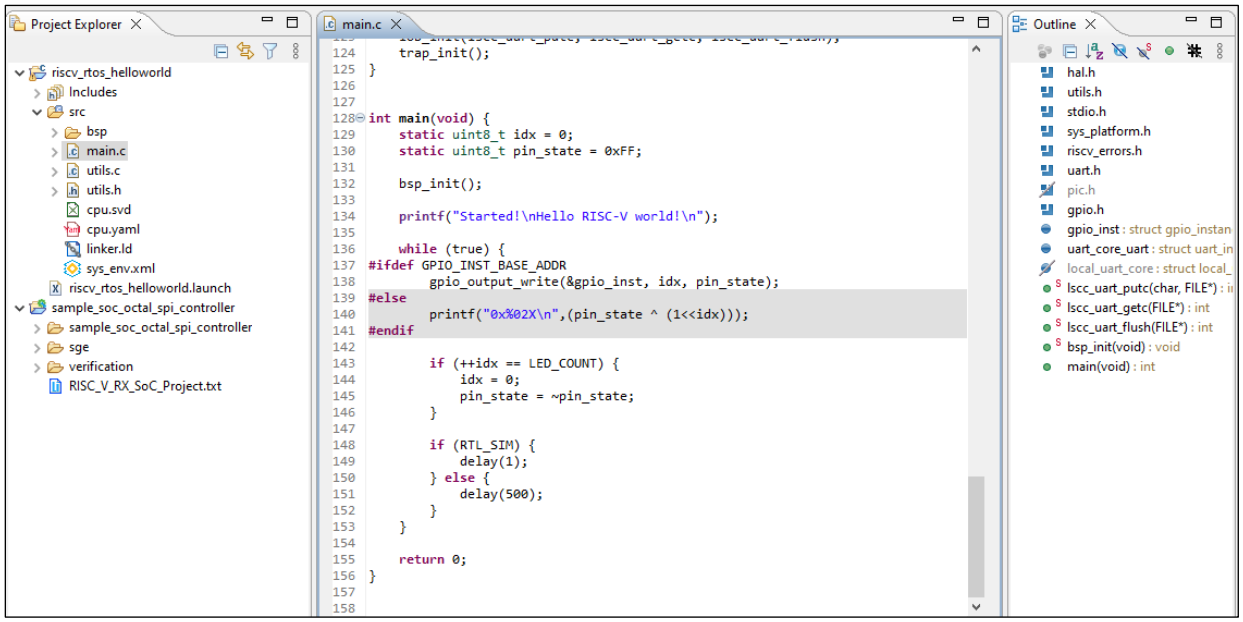


Figure 6.8. Build C/C++ Project Result

Use the generated C code file to further develop the drivers and test sequences. Once the C code is ready, compile and load the code to run on the FPGA board. Refer to the On-Chip Debugging section under Propel SDK Flow in A Step-By-Step Approach to Lattice Propel (FPGA-AN-02052) for details on how to run the design on the FPGA board.

6.4. Hardware Testing

The Octal SPI Controller is validated using the Avant-X Versa board with the LAV-AT-X70ES device. The following block diagram shows the system design setup created in Propel Builder.

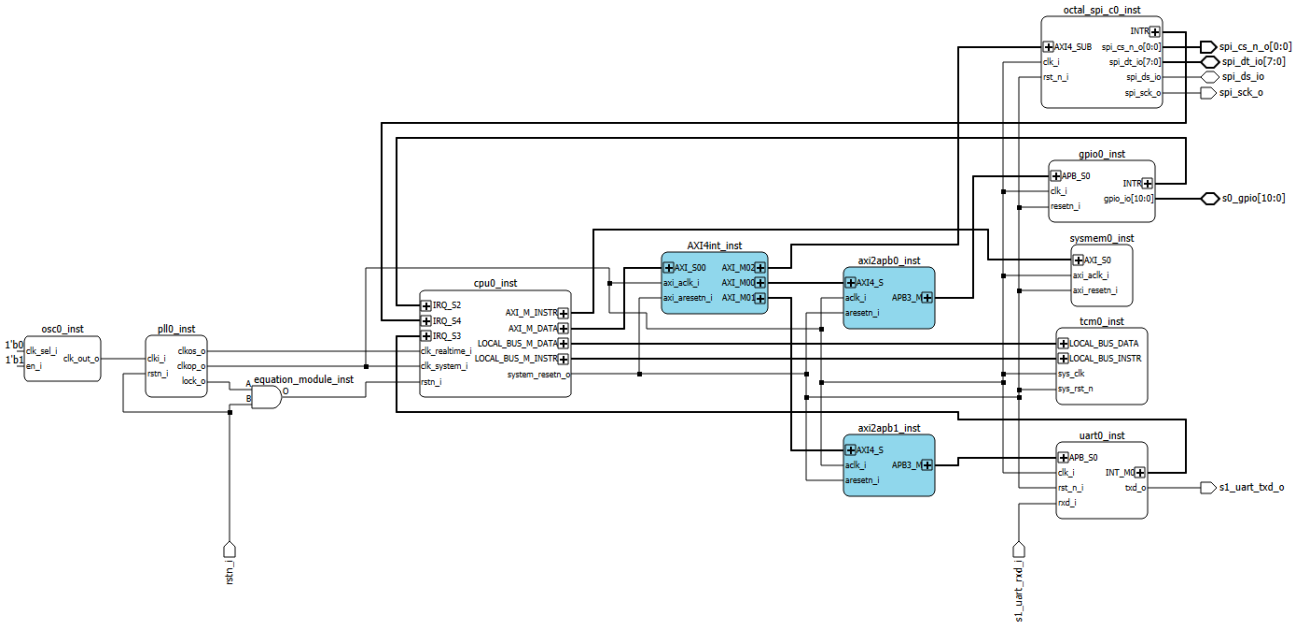


Figure 6.9. Octal SPI Controller SoC Design Block Diagram

The system clock frequency used in this setup is 100 MHz while the SPI clock frequency is 50 MHz. The system design is imported into the Radiant software to be synthesized. The bitstream is generated after running the Synthesis, Map, and Place and Route Flow in the Radiant software. This bitstream is then loaded into the FPGA using the Radiant Programmer tool. The test sequence and drivers (C code) are developed and run using Propel SDK.

The following table lists the test items, procedures, and results.

Table 6.2. Octal SPI Controller IP Test Items and Procedures

No.	Test Item / Feature	Test Method / Procedure	Result
1	Read ID, read and write to flash register in x1 mode, generic command	1. Initialize Octal SPI Controller: • SCK divider = 1 (50 MHZ). • Enable interrupt for bus access error, flash program fail, and flash erase fail. • Flash device starts at x1 mode. 2. Initiate read ID (x1 mode) on SPI flash using generic command packet. 3. Enable xSPI DTR mode by writing to non-volatile register using generic command packet. 4. Repeat step 2 in x8 mode. 5. Read the non-volatile register to get the I/O mode. Compare to expected value (x8). 6. Disable xSPI DTR mode by writing to non-volatile register using generic command packet. 7. Read the non-volatile register to get the I/O mode. Compare to expected value (x1). 8. Compare the returned data of read ID in step 2 and step 4 to the known ID value of Micron flash.	Pass
2	Read ID, read and write to flash register in x8 mode, generic command	Same as Test #1.	Pass
3	Random erase 4K, page program, fast read in x1 mode, generic command	1. Initialize Octal SPI Controller: • SCK divider = 1 (50 MHZ). • Enable interrupt for bus access error, flash program fail, and flash erase fail • Flash device starts at x1 mode. 2. Randomize a 4-KiB aligned starting flash address that is beyond the size of the bitstream (avoid overwrite). 3. Initiate a 4-KiB erase using generic command packet. Once the erase command is sent, wait until transaction is done then send a command to read the status register to determine if erase is done in the flash. Once confirmed done, send a command to read the status flag register to check if the erase command is successful. 4. Randomize an offset within a page (256 bytes) and a random length such that offset + length does not exceed 256 bytes. Store the offset and length in an array and repeat this 16 times (incrementing the page offset each time). 5. With the list of page offset and length, iterate through these items and initiate a page program command with random data. After a page program command is sent, wait until the transaction is done then send a command to read the status register to determine if program is done in the flash. Once confirmed done, send a command to read the status flag register to check if the program command is successful. 6. Compare all the program data and read data.	Pass

No.	Test Item / Feature	Test Method / Procedure	Result
4	Random erase 4K, page program, fast read in x8 mode, generic command	Same as Test #3. After step 1, enable xSPI DTR mode by writing to non-volatile register using generic command packet. After step 5, disable xSPI DTR mode by writing to non-volatile register using generic command packet.	Pass
5	Random erase 4K, page program, fast read in x1 mode, supported command	Same as Test #3 but using supported command packet format.	Pass
6	Random erase 4K, page program, fast read in x8 mode, supported command	Same as Test #4 but using supported command packet format.	Pass
7	Random direct bus write and read flash access in x1 mode	1. Initialize Octal SPI Controller: • SCK divider = 1 (50 MHZ). • Enable interrupt for bus access error, flash program fail, and flash erase fail. • Flash device starts at x1 mode. 2. Randomize a 4-KiB aligned starting flash address that is beyond the size of the bitstream (avoid overwrite). 3. Initiate a 4-KiB erase using generic command packet. Once the erase command is sent, wait until transaction is done then send a command to read the status register to determine if erase is done in the flash. Once confirmed done, send a command to read the status flag register to check if the erase command is successful. 4. Select a program command and a read command to be used: • Setup the target base address and the size of address map. • Configure the supported flash commands. • Set the supported command configuration register based on the program command and read command to be used. 5. Initiate a random bus write access on the target address space. This is expected to trigger a write access to the flash device. 6. Initiate bus read access on the same address in step 5. This is expected to trigger a read access to the flash device. 7. Compare all written and read data.	Pass
8	Random direct bus write and read flash access in x8 mode	Same as Test #7. After step 1, enable xSPI DTR mode by writing to non-volatile register using generic command packet. After step 6, disable xSPI DTR mode by writing to non-volatile register using generic command packet.	Pass

No.	Test Item / Feature	Test Method / Procedure	Result
9	Random XiP mode access in x1 mode	1. Initialize Octal SPI Controller: • SCK divider = 1 (50 MHZ). • Enable interrupt for bus access error, flash program fail, and flash erase fail. • Flash device starts at x1 mode. 2. Randomize a 4-KiB aligned starting flash address that is beyond the size of the bitstream (avoid overwrite). 3. Initiate a 4-KiB erase using generic command packet. Once the erase command is sent, wait until transaction is done then send a command to read the status register to determine if erase is done in the flash. Once confirmed done, send a command to read the status flag register to check if the erase command is successful. Once erase is done, fill up the 4-KiB block by sending 16 x 256 bytes page program command with random data. 4. Select a read command to be used: • Setup the target base address and the size of address map. • Configure the supported flash commands. • Set the supported command configuration register based on the read command to be used. 5. Enable XiP mode by writing to the non-volatile register. 6. Activate the XiP transaction by sending the initial read (at address 0) using generic command packet. This read transaction has the read command code sent as this is the first read. After this, the command need not be sent and the XiP confirmation bit (first bit of the dummy cycle after the address) just needs to be 0 to keep the XiP mode active. 7. Initiate a bus read access and sweep the whole 4-KiB block. 8. Terminate the XiP mode by sending a read (at address 0) with confirmation bit = 1 using generic command packet. Then, write to the non-volatile register to disable XiP mode. 9. Compare all written and read data.	Pass
10	Random XiP mode access in x8 mode	Same as Test #9. After step 1, enable xSPI DTR mode by writing to non-volatile register using generic command packet. After step 8, disable xSPI DTR mode by writing to non-volatile register using generic command packet.	Pass

The Octal SPI Controller IP has also been validated on the following hardware and configurations using a similar test method and SoC design:

- Avant-X Versa Board: IP configured as Octal Flash Controller (JEDEC xSPI enabled), 200 MHz system clock, 200 MHz SPI clock.
- Avant-E Evaluation Board: IP configured as QSPI Flash Controller, 100 MHz system clock, 100 MHz SPI clock.
- CertusPro-NX Versa Board: IP configured as QSPI Flash Controller, 100 MHz system clock, 50 MHz SPI clock.

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.

Note: 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.

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 following steps describe how to generate the Octal SPI Controller IP in the Lattice Radiant software.

To generate the Octal SPI Controller 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 **Octal SPI Controller** 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.

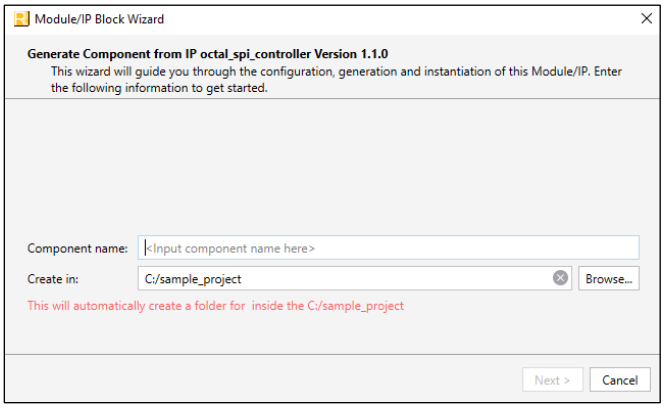


Figure 7.1. Module/IP Block Wizard

4. Enter values in the **Component name** and the **Create in** fields, then click **Next**.
5. Customize the selected Octal SPI Controller IP using drop-down lists and check boxes. Figure 7.2 through Figure 7.5 show an example configuration of the Octal SPI Controller IP. For details on the configuration options, refer to the IP Parameter Description section.

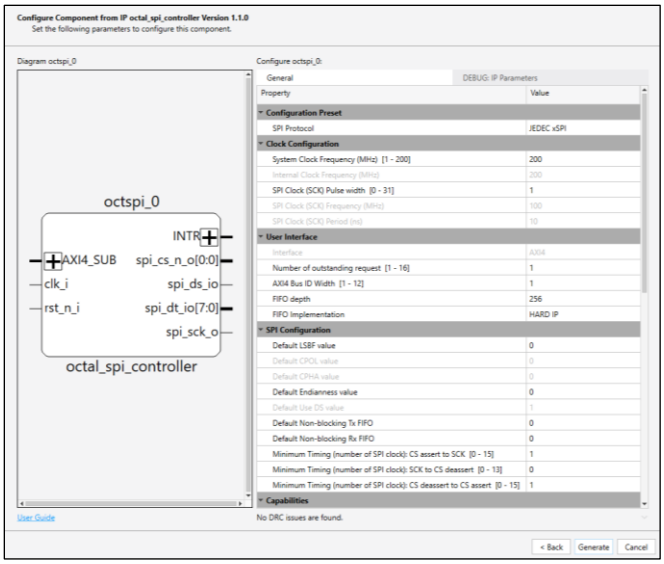


Figure 7.2. IP Configuration (View 1)

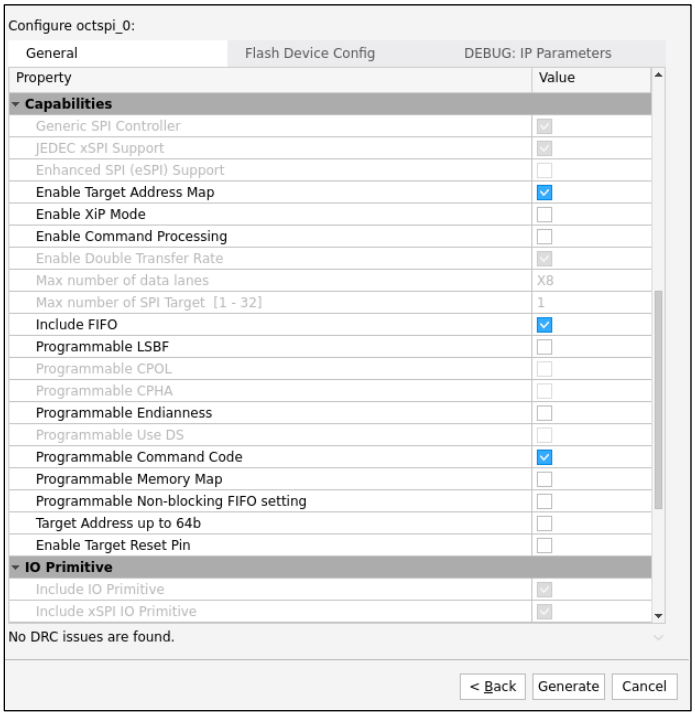


Figure 7.3. IP Configuration (View 2)

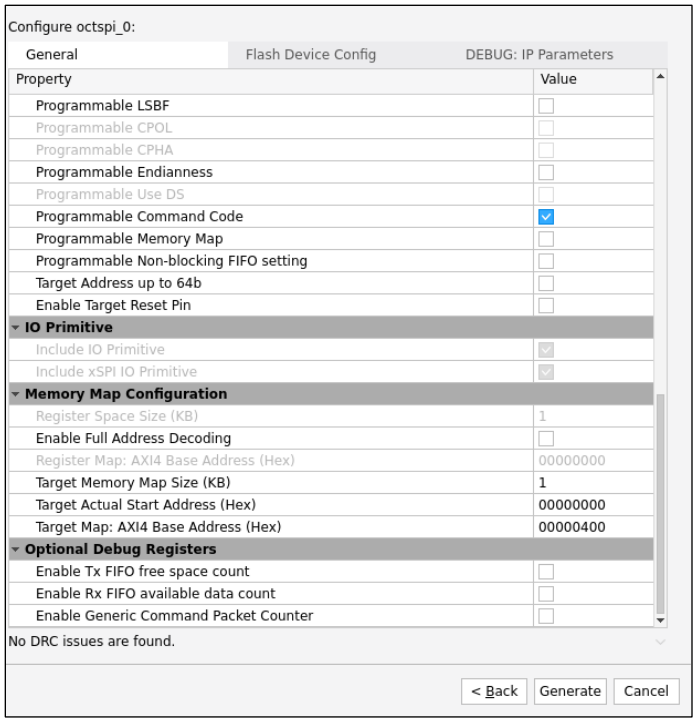


Figure 7.4. IP Configuration (View 3)

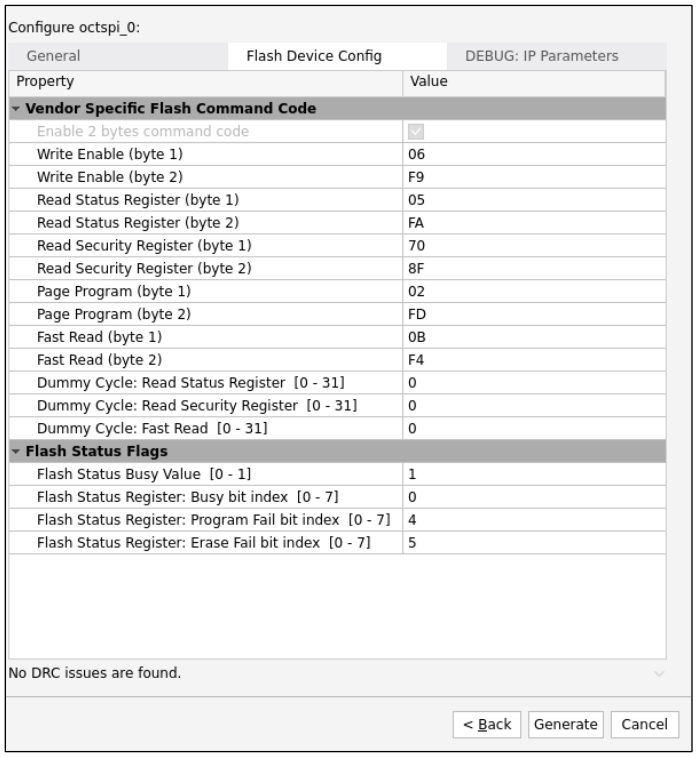


Figure 7.5. IP Configuration (View 4)

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

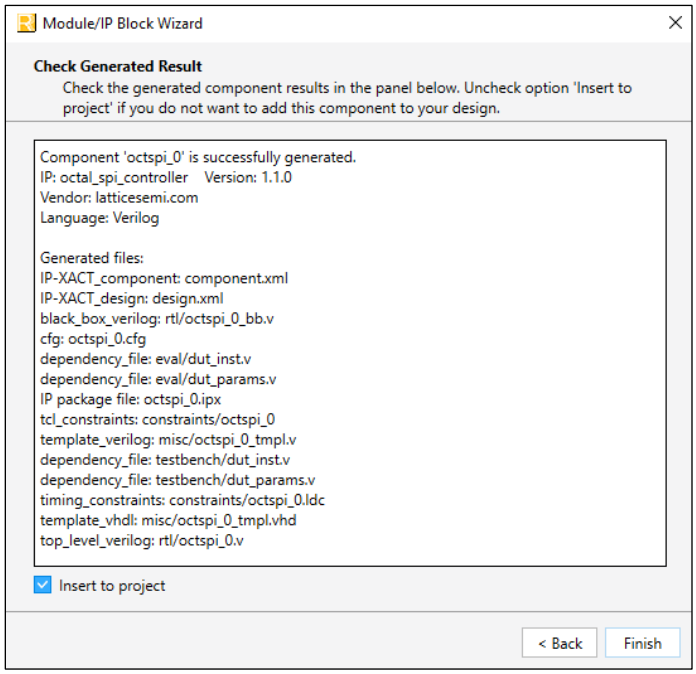


Figure 7.6. Check Generated Result

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 Octal SPI Controller 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 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. The timing constraints for the IP are defined in relevant constraint files. Refer to *constraints/<instance_name>.sdc* for the recommended constraints for this IP.

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. In the File List view, right-click on **Input Files**, select **Add > Existing Simulation File** followed by *testbench/tb_top.v*.

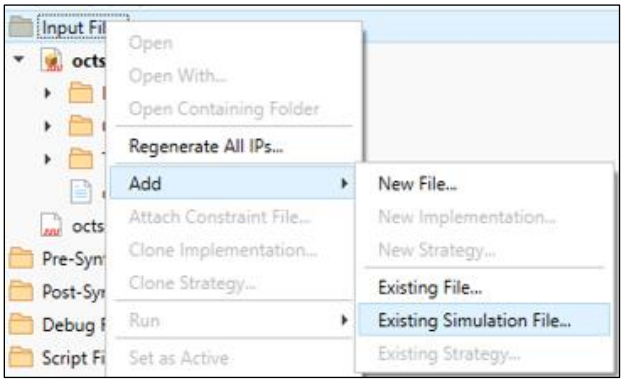


Figure 7.7. Adding Existing Simulation File (View 1)

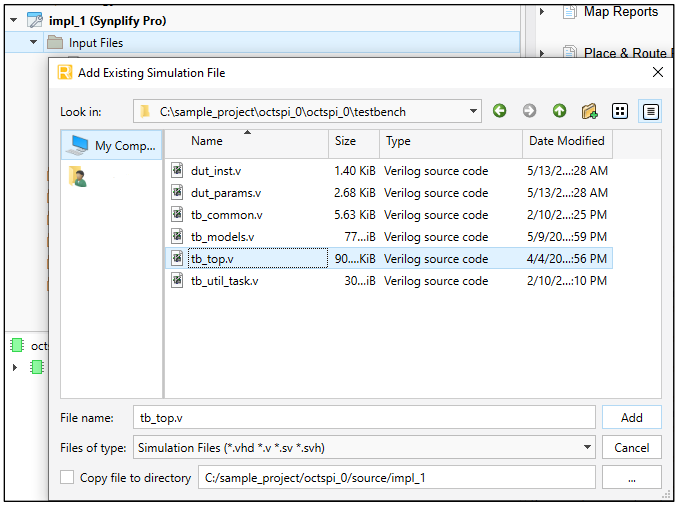


Figure 7.8. Adding Existing Simulation File (View 2)

2. Check to make sure the **Copy file to directory** option is unchecked, then click **Add**.

3. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in Figure 7.9.

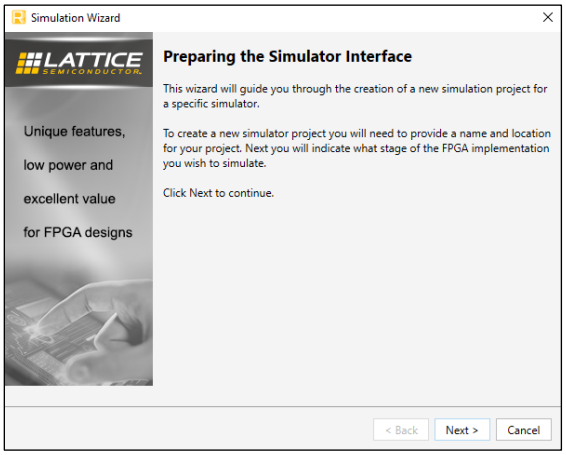


Figure 7.9. Simulation Wizard (View 1)

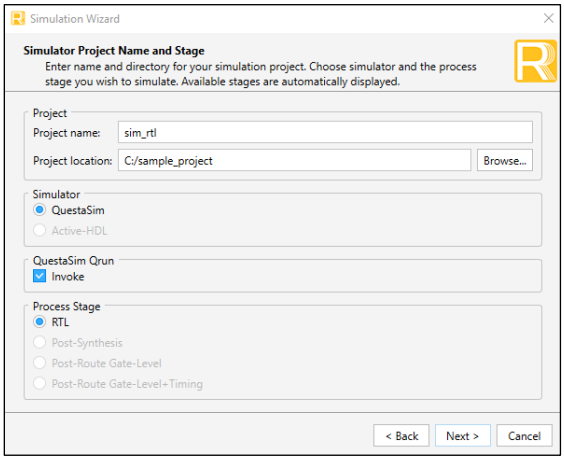


Figure 7.10. Simulation Wizard (View 2)

4. Click **Next** to open the **Add and Reorder Source** window as shown in Figure 7.11. Note that only two files are listed, *tb_top.v* and the generated IP wrapper file. If there are other files, they must be removed. Otherwise, this may cause a failure in the compilation and simulation.

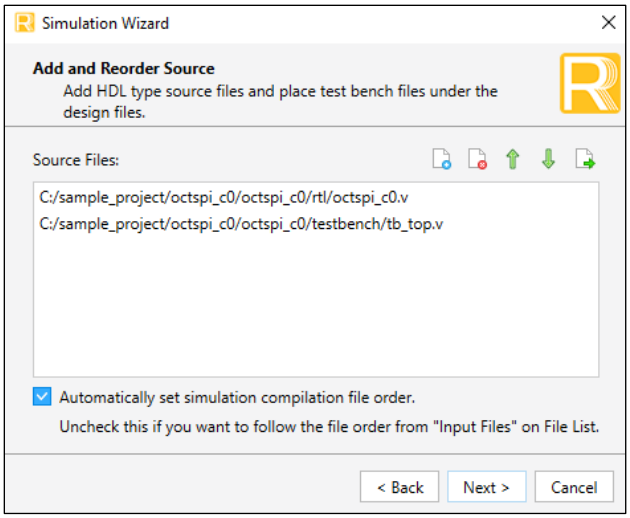


Figure 7.11. Add and Reorder Source

5. Click **Next**. The **Summary** window is shown.
6. Click **Finish** to run the simulation.

The waveform in Figure 7.12 shows an example simulation result.

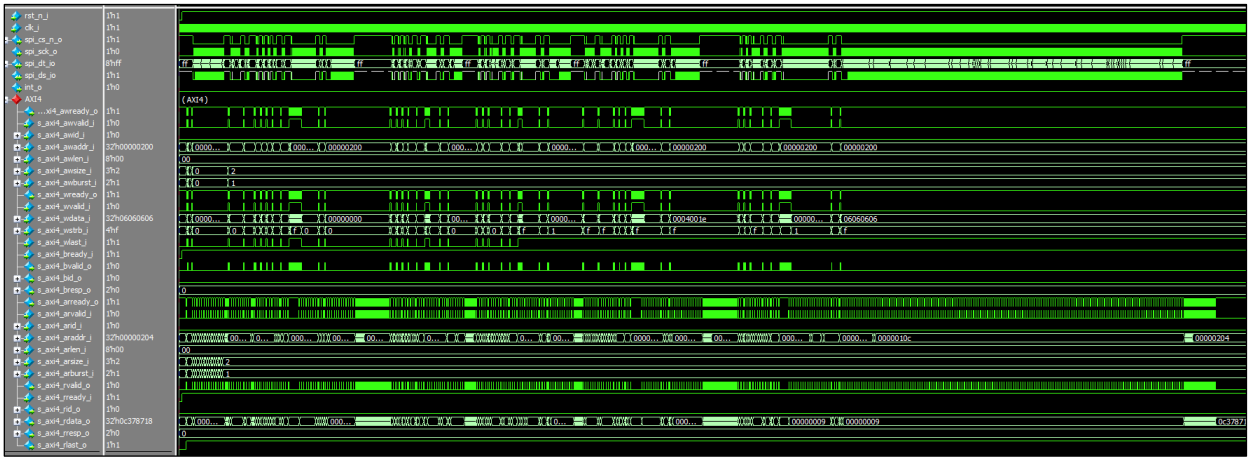


Figure 7.12. Simulation Waveform

7.4.1. Simulation Results

The simulation generates random flash access (write/read) and compares the written and read data. Figure 7.13 shows an example log after the simulation is done.

```
# [    57494000]:[Info   ]--- [DATA MATCH] [DWORD 51] Expected = 32'h65d2_73cb : Observed = 32'h65d2_73cb---[tb_top.random_octal_flash_access]
# [    57524000]:[Info   ]--- [DATA MATCH] [DWORD 52] Expected = 32'ha934_be52 : Observed = 32'ha934_be52---[tb_top.random_octal_flash_access]
# [    57554000]:[Info   ]--- [DATA MATCH] [DWORD 53] Expected = 32'heaad_74d5 : Observed = 32'heaad_74d5---[tb_top.random_octal_flash_access]
# [    57584000]:[Info   ]--- [DATA MATCH] [DWORD 54] Expected = 32'h2fc0_695f : Observed = 32'h2fc0_695f---[tb_top.random_octal_flash_access]
# [    57614000]:[Info   ]--- [DATA MATCH] [DWORD 55] Expected = 32'h6a24_7dd4 : Observed = 32'h6a24_7dd4---[tb_top.random_octal_flash_access]
# [    57644000]:[Info   ]--- [DATA MATCH] [DWORD 56] Expected = 32'h53c9_b7a7 : Observed = 32'h53c9_b7a7---[tb_top.random_octal_flash_access]
# [    57674000]:[Info   ]--- [DATA MATCH] [DWORD 57] Expected = 32'h0c37_8718 : Observed = 32'h0c37_8718---[tb_top.random_octal_flash_access]
# [    60173000]:Ending Simulation...
# ************************
# ***Message Counters*****
#      Info    : 845
#      Warning: 0
#      Debug   : 0
#      Error   : 0
# ************************
# ***SIMULATION PASSED***
```

Figure 7.13. Simulation Results

8. Design Considerations

8.1. Design Considerations in Propel-Based Designs

- When *Enable Target Address Map* is selected, the total address space allocation of the Octal SPI Controller IP is twice the size of the selected *Total Target Memory Map Size*. This is because the register space allocation of 1 KiB is added to the target memory map size and the nearest power of 2 is taken as the total size.
- *Enable Full CSR Address Decoding* must not be selected. The address routing is handled by the bus interconnect thus only the lower address bits are necessary. The base address is automatically assigned by the Propel Builder software.
- *Target Map: AXI4 Base Address* must be set based on the *Total Target Memory Map Size* + base address offset assigned by Propel to the Octal SPI Controller. For example, if base address offset assigned = 0x8000_0000 and *Total Target Memory Map Size* = 64 KiB, then *Target Map: AXI4 Base Address* = 0x8001_0000.

Appendix A. Resource Utilization

Table A.1 shows a sample resource utilization of the Octal SPI Controller IP core using the LAV-AT-X70 device.

Table A.1. Resource Utilization

Configuration	Slices	Registers	LUTs	EBRs
Default	1289	1289	1377	2
SPI Protocol: Custom Max number of data lanes: X8 Enable Double Transfer Rate	1294	1294	1617	2
SPI Protocol: Custom Max number of data lanes: X4	1050	1050	1306	2
SPI Protocol: Custom Max number of data lanes: X1	1061	1061	1294	2
SPI Protocol: Custom Max number of data lanes: X8 Enable Double Transfer Rate Enable Target Address Map Enable XiP Mode Enable Command Processing	1739	1739	2753	2
SPI Protocol: Custom Max number of data lanes: X4 Enable Target Address Map Enable XiP Mode Enable Command Processing	1402	1402	2094	2
SPI Protocol: Custom Max number of data lanes: X1 Number of outstanding requests: 2	1287	1287	1527	2

References

- Octal SPI Controller IP Release Notes (FPGA-RN-02011)
- Octal SPI Controller Driver API Reference (FPGA-TN-02400)
- A Step-By-Step Approach to Lattice Propel (FPGA-AN-02052)
- Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059)
- Avant-E web page
- Avant-G web page
- Avant-X web page
- Certus-NX web page
- CertusPro-NX web page
- CrossLink-NX web page
- MachXO5-NX web page
- Avant-X Versa Board web page
- Avant-E Evaluation Board web page
- CertusPro-NX Versa Board web page
- Octal SPI Controller IP Core 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 1.4, IP v1.4.0, December 2025

Section	Change Summary
All	• Added a note on the IP version in the Quick Facts and Revision History sections. • Made minor editorial fixes.

Introduction	• In Table 1.1. Summary of the Octal SPI Controller IP:
	  • Updated IP core version and software version in *Lattice Implementation*.
	  • Added *Driver Support* row.
	• Removed the Ordering Part Number section.

Register Description	In Table 5.3. Octal SPI Configuration 0 Register: • Updated sck_src_i to clk_i in the description of sck_rate.

Example Design	• Removed note on check and cross symbols in the Example Design Configuration section.
	• In Table 6.1. Octal SPI Controller IP Configuration Supported by the Example Design:
	  • Replaced check symbol with *Checked* and cross symbol with *Unchecked*.
	  • Updated attributes names to *Enable Full CSR Address Decoding*, *Register Map: AXI4 Base Address (32b Hex)*, *Target Actual Start Address (32b Hex)*, and *Target Map: AXI4 Base Address (32b Hex)*.

Designing with the IP	• Added note on IP version in GUI.
	• In the Generating and Instantiating the IP section:
	  • Split Step 2 into Steps 2, 3, and 4.
	  • Updated Steps 2 through 7.
	• In the Generated Files and File Structure section:
	  • Updated statement on using example top-level reference as starting template.
	• Added statement referencing Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059) in the Timing Constraints section.

Design Considerations	Updated attribute name to *Enable Full CSR Address Decoding* in the Design Considerations in Propel-Based Designs section.

Revision 1.3, IP v1.3.0, June 2025

Section	Change Summary
Abbreviations in This Document	Updated definition of *APB*.

Introduction	• In Table 1.1. Summary of the Octal SPI Controller IP:
	  • Renamed *Supported FPGA Family* to *Supported Devices* and incorporated *Targeted Devices* information into row.
	  • Removed *Targeted Devices* row.
	  • Added AXI4-Lite and APB to *Supported User Interface*.
	  • Added IP core version to *Lattice Implementation*.
	• In Table 1.2. Octal SPI Controller IP Support Readiness:
	  • Added x2 I/O width row for Avant device and x2/x8 I/O width row for CertusPro-NX device.
	  • Made minor editorial changes to Notes 1 and 2.
	• Added *dual bus interface option for separate CSR and target data path interfaces* in the Features section.

Functional Description	• Updated Figure 2.1. Octal SPI Controller IP Core Functional Diagram.
	• In the User Interfaces section:
	  • Added description of single bus and dual bus interfaces.
	  • In Table 2.1. User Interfaces and Supported Protocols:

Section	Change Summary
	• Removed Advanced eXtensible Interface 4. • Added CSR Interface and Target Data Interface. • Added the FIFO Burst Transfer for Indirect Flash Access section. • In Table 2.18. Using Supported Flash Command Packet: Fast Read (256 Bytes) at Address (0x0159_E500), x8 I/O Width, DTR, 4-Byte Address Mode: • Updated [31:16] flash_cmd_code description to *Fast Read*.
IP Parameter Description	In Table 3.1. General Attributes: • Updated description for SPI Clock (SCK) Pulse width under Clock Configuration. • Updated selectable values and description for Interface under User Interface. • Added Enable AXI4-Lite ID and FIFO Interface Mapping attributes under User Interface. • Added Programmable SCK Divider under Supported Capabilities.
Signal Description	• In Table 4.1. Octal SPI Controller Ports: • Updated section title to *AXI4 and AXI4-Lite Interface (Data and CSR)*. • In AXI4 or AXI4-Lite Interface (Data or CSR) section: • Updated descriptions for s_axi4_awid_i[x:0], s_axi4_awlen_i[7:0], s_axi4_awsize_i[2:0], and s_axi4_awburst_i[1:0] under Write Address Channel. • Updated description for s_axi4_wlast_i under Write Data Channel. • Updated description for s_axi4_bid_o[x:0] under Write Response Channel. • Updated descriptions for s_axi4_arid_i[x:0], s_axi4_arlen_i[7:0], s_axi4_arsize_i[2:0], and s_axi4_arburst_i[1:0] under Read Address Channel. • Updated description for s_axi4_rid_o[x:0] under Read Data Channel. • Added the AXI4-Lite Interface (CSR) section. • Added the AXI4 Interface (CSR) section. • Added the APB Interface (Data or CSR) section. • Added notes on signal availability and signal naming based on single bus or dual bus interface selection.
Register Description	• Updated description to reference the CSR interface and buses in the Overview section. • In Table 5.2. Summary of Octal SPI Controller IP Core Registers: • Corrected offset addresses for Packet Header 1 Register and Packet Header 2 Register. • Added note on the applicable interface in the Tx FIFO Register section. • Added note on the applicable interface in the Rx FIFO Register section.
Designing with the IP	• Updated Figure 7.8. Adding Existing Simulation File (View 2). • Added new Step 2 regarding ensuring *Copy file to directory* option is unchecked in the Running Functional Simulation section.
References	• Updated listing name to Lattice Radiant Software web page. • Added Octal SPI Controller IP Core web page.

Revision 1.2, IP v1.2.0, February 2025

Section	Change Summary
Introduction	• Added IP core version to Lattice Implementation in Table 1.1. Summary of the Octal SPI Controller IP. • Added notes to Table 1.2. Octal SPI Controller IP Support Readiness.
References	Added Octal SPI Controller Driver API Reference.

Revision 1.1, IP v1.1.0, December 2024

Section	Change Summary
All	Made minor editorial changes.
Cover	Added IP version.

Section	Change Summary

Introduction	• In Table 1.1. Summary of the Octal SPI Controller IP:
	  • Added Certus-NX, CertusPro-NX, CrossLink-NX, and MachXO5-NX to *Supported FPGA Family*.
	  • Added IP Changes row.
	  • Removed IP Version row.
	  • Added LAV-AT-E70, LAV-AT-G70, LFD2NX-17, LFD2NX-40, LFCPNX-50, LFCPNX-100, LIFCL-17, LIFCL-33, LIFCL-40, LFMXO5-25, LFMXO5-55T, and LFMXO5-100T to *Targeted Devices*.
	  • Added Resources row.
	  • Added IP core version to *Lattice Implementation*.
	• Added the IP Support Summary section.
	• Removed the IP Validation Summary section.
	• Added the Hardware Support section.

Functional Description	• Updated description in the Data Byte Order section.
	• In Table 2.3. Packet Fields – Generic SPI Command Format:
	  • Updated title.
	  • Updated description for sup_flash_cmd and wait_ds or dummy_data.
	  • Removed with_payload, user_cmd_code[4:0], en_2byte_fcc, and num_wait_state[7:0].
	  • Removed bytes 4 through 15.
	• Added Table 2.4. Packet Fields – Supported Flash Command Format.

IP Parameter Description	• In Table 3.1. General Attributes:
	  • Added *AXI4 Bus ID Width* and *FIFO Implementation* attributes under User Interface.
	  • Removed *Programmable Endianness* attribute.
	  • Updated description for *SPI Protocol*, *Default CPOL value*, *Default CPHA value*, *Default Use DS value*, *Enable XiP Mode*, *Include IO Primitive*, *Use xSPI IO Primitive*, *Register Map: AXI4 Base Address (32b Hex)*, *Target Memory Map Size (KiB)*, *Target Actual Start Address (32b Hex)*, *Total Target Memory Map Size (KiB)*, *Target Map: AXI4 Base Address (32b Hex)*, *Packet Counter max count*, and *Enable Supported Command Packet Counter*.
	  • Updated attribute name to *Enable Full CSR Address Decoding*, *Register Map: AXI4 Base Address (32b Hex)*, *Target Actual Start Address (32b Hex)*, and *Target Map: AXI4 Base Address (32b Hex)*.
	• In Table 3.2. Flash Device Configuration:
	  • Added *XiP Mode Address Padding* attribute under XiP Mode.
	  • Updated description for *Enable 2 bytes command code*, *Write Enable (byte 2)*, *Read Status Register (byte 2)*, *Read Security Register (byte 2)*, *Page Program (byte 2)*, and *Fast Read (byte 2).*

Signal Description	In Table 4.1. Octal SPI Controller Ports: • Updated description for s_axi4_awid_i[x:0], s_axi4_bid_o[x:0], s_axi4_arid_i[x:0], s_axi4_arlen_i[7:0], and s_axi4_rid_o[x:0].
Register Description	Updated description for endianness in Table 5.3. Octal SPI Configuration 0 Register.
Example Design	• Added introductory paragraph to list the evaluation boards used for the example design. • Added *AXI Bus ID Width* and *FIFO Implementation* under User Interface in Table 6.1. Octal SPI Controller IP Configuration Supported by the Example Design. • Added information about validation using other configurations and boards in the Hardware Testing section.
Designing with the IP	Updated Figure 7.1. Module/IP Block Wizard, Figure 7.2. IP Configuration (View 1), and Figure 7.6. Check Generated Result.
Known Issues or Limitations	Removed section.
References	• Added Avant-G, Avant-X, Certus-NX, CertusPro-NX, CrossLink-NX, and MachXO5-NX web pages. • Removed Lattice Radiant Software User Guide, Lattice Propel Builder User Guide, and

Section	Change Summary
	Lattice Propel SDK User Guide. • Added Versa/Evaluation Board and Lattice Solutions Reference Designs web pages.

Revision 1.0, September 2024

Section	Change Summary
All	Initial release.

www.latticesemi.com