



SPI Controller IP

IP Version: v2.4.0

User Guide

FPGA-IPUG-02069-2.3

July 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

Contents	3
Abbreviations in This Document.....	7
1. Introduction	8
1.1. Overview of the IP	8
1.2. Quick Facts	8
1.3. IP Support Summary	9
1.4. Features	10
1.5. Licensing and Ordering Information	10
1.6. Hardware Support	10
1.7. Conventions	11
1.7.1. Nomenclature.....	11
1.7.2. Signal Names	11
1.7.3. Attribute.....	11
2. Functional Description.....	12
2.1. IP Architecture Overview	12
2.2. Clocking	13
2.2.1. Clocking Overview	13
2.3. Reset	13
2.3.1. Reset Overview	13
2.4. User Interfaces	14
2.5. Operation Details	14
2.5.1. Clock Generation	14
2.5.2. Data Transmission	14
2.5.3. Clocking Modes: Clock Polarity and Clock Phase	15
2.5.4. Chip Select Pulse Mode	16
2.5.5. SPI Enable	16
2.6. Independent Target Configuration	17
2.7. Daisy Chain Configuration	17
2.8. Programming Flow	18
2.8.1. Initialization.....	18
2.8.2. Transmit/Receive Operation Interrupt Mode	18
2.8.3. Transmit/Receive Operation Polling Mode	19
3. IP Parameter Description.....	20
3.1. General.....	20
4. Signal Description	22
5. Register Description	24
5.1. Overview	24
5.2. Write Data Register (WR_DATA_REG)	25
5.3. Read Data Register (RD_DATA_REG)	25
5.4. Chip Select Register (CHP_SEL_REG).....	26
5.5. Configuration Register (CFG_REG)	27
5.6. Clock Prescaler Registers (CLK_PRESCAL_REG, CLK_PRESCAL_REG)	28
5.7. Interrupt Status Register (INT_STATUS_REG)	29
5.8. Interrupt Enable Register (INT_ENABLE_REG)	30
5.9. Interrupt Set Register (INT_SET_REG).....	31
5.10. Word Count Register (WORD_CNT_REG).....	32
5.11. Word Count Reset Register (WORD_CNT_RST_REG).....	32
5.12. Target Word Count Register (TGT_WORD_CNT_REG)	32
5.13. FIFO Reset Register (FIFO_RST_REG)	32
5.14. Chip Select Polarity Register (CHP_SEL_POL_REG)	33
5.15. FIFO Status Register (FIFO_STATUS_REG).....	34
5.16. SPI Enable Register (SPI_ENABLE_REG)	34

6.	Example Design.....	35
6.1.	Example Design Supported Configuration	35
6.2.	Overview of the Example Design and Features	35
6.3.	Design Components Example.....	37
6.4.	Generating the Example Design	38
6.5.	Hardware Testing	41
6.5.1.	Hardware Testing Setup	41
6.5.2.	Expected Output	41
7.	Designing with the IP	42
7.1.	Generating and Instantiating the IP	42
7.1.1.	Generated Files and File Structure	44
7.2.	Design Implementation	45
7.3.	Running Functional Simulation	45
7.3.1.	Simulation Results	47
7.4.	Constraining the IP	47
	Appendix A. Resource Utilization	48
	References	51
	Technical Support Assistance	52
	Revision History	53

Figures

Figure 2.1. SPI Controller IP Core Block Diagram	12
Figure 2.2. SPI Controller IP Clock Domain Block Diagram	13
Figure 2.3. Clocking Mode 0 (CPOL=0, CPHA=0)	15
Figure 2.4. Clocking Mode 1 (CPOL=0, CPHA=1)	15
Figure 2.5. Clocking Mode 2 (CPOL=1, CPHA=0)	15
Figure 2.6. Clocking Mode 3 (CPOL=1, CPHA=1)	16
Figure 2.7. Timing Diagram for a Transfer with SSNP Set to 1	16
Figure 2.8. Timing Diagram for a Transfer with SSNP Set to 0	16
Figure 2.9. SPI Enable Register Enable and Disable	16
Figure 2.10. SPI Enable after Writing Data to Transmit FIFO	16
Figure 2.11. Long Gap between Writing Data to Transmit FIFO	17
Figure 2.12. Independent Target Configuration	17
Figure 2.13. Daisy Chain Configuration	18
Figure 6.1. SPI Controller IP Example Design in Propel SoC Project	36
Figure 6.2. Sample C-Code Test Routine	36
Figure 6.3. SPI Controller Example Design Block Diagram	37
Figure 6.4. Create SoC Project	38
Figure 6.5. Define Instance	39
Figure 6.6. Build SoC Project Result	39
Figure 6.7. Lattice C/C++ Design Project	40
Figure 6.8. Build C/C++ Project Result	40
Figure 6.9 Sample SPI Transaction between Controller and Target	41
Figure 7.1. Module/IP Block Wizard	42
Figure 7.2. IP Configuration	43
Figure 7.3. Check Generated Result	44
Figure 7.4. Simulation Wizard	45
Figure 7.5. Add and Reorder Source	46
Figure 7.6. Simulation Waveform	46
Figure 7.7. AHB-Lite Write Operation	47
Figure 7.8. AHB-Lite Read Operation	47
Figure 7.9. SPI Controller Transaction	47
Figure 7.10. SPI Controller Simulation Completed Successfully	47

Tables

Table 1.1. Summary of the SPI Controller IP	8
Table 1.2. SPI Controller IP Support Readiness	9
Table 2.1. User Interfaces and Supported Protocols	14
Table 2.2. Clocking Modes (CPOL and CPHA)	15
Table 3.1. General Attributes	20
Table 4.1. SPI Controller IP Signal Description.....	22
Table 5.1. Summary of SPI Controller IP Core Registers	24
Table 5.2. Register Access Types	24
Table 5.3. Write Data Register	25
Table 5.4. Read Data Register	25
Table 5.5. Chip Select Register	26
Table 5.6. Configuration Register	27
Table 5.7. Clock Prescaler Lower Register	28
Table 5.8. Clock Prescaler Higher Register	28
Table 5.9. Interrupt Status Register.....	29
Table 5.10. Interrupt Enable Register	30
Table 5.11. Interrupt Set Register.....	31
Table 5.12. Word Count Register.....	32
Table 5.13. Word Count Reset Register.....	32
Table 5.14. Target Word Count Register	32
Table 5.15. FIFO Reset Register	32
Table 5.16. Chip Select Polarity Register	33
Table 5.17. FIFO Status Register	34
Table 5.18. FIFO Reset Register	34
Table 6.1. SPI Controller IP Configuration Used on Example Design	35
Table 7.1. Generated File List	44
Table A.1. Resource Utilization for LAV-AT-70E-3LFG1156I	48
Table A.2. Resource Utilization for LFMXO5-25-9BBG400I	48
Table A.3. Resource Utilization for LFCPNX-100-9BBG484I.....	49
Table A.4. Resource Utilization for LFD2NX-40-9BG256I.....	49
Table A.5. Resource Utilization for LIFCL-40-9BG400I.....	50
Table A.6. Resource Utilization for LN2-CT-20-1CBG484C	50

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AHB	Advanced High-performance Bus
AHB-Lite	Advanced High-performance Bus Lite
APB	Advanced Peripheral Bus
AMBA	Advanced Microcontroller Bus Architecture
CPHA	Clock Phase
CPOL	Clock Polarity
DUT	Device Under Test
EBR	Embedded Block RAM
FIFO	First Input First Output
FPGA	Field Programmable Gate Array
GPIO	General Purpose I/O
IP	Intellectual Property
LSB	Least Significant Bit
LUT	Look-Up Table
MC	Microcontroller
MSB	Most Significant Bit
PIC	Programmable Interrupt Controller
RTL	Register Transfer Level
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
UART	Universal Asynchronous Receiver/Transmitter

1. Introduction

1.1. Overview of the IP

The Serial Peripheral Interface (SPI) is a high-speed synchronous, serial, and full-duplex interface that allows a serial bitstream of configured length, 8, 16, 24, or 32 bits to be shifted into and out of the device at a programmed bit-transfer rate. The Lattice SPI Controller IP Core is normally used to communicate with external SPI target devices such as display drivers, SPI EPROMS, and analog-to-digital converters.

1.2. Quick Facts

Table 1.1. Summary of the SPI Controller IP

IP Requirements	Supported Devices	CrossLink™-NX, Certus™-NX, Certus-NX-RT, CertusPro™-NX, CertusPro-NX-RT, MachXO5™-NX, Lattice Avant™, Certus-N2
	IP Changes	Refer to the SPI Controller IP Release Notes (FPGA-RN-02015) .
Resource Utilization	Supported User Interface	Lattice Memory Mapped Interface (LMMI), Advanced High-performance Bus Lite (AHB-Lite), Advanced Peripheral Bus (APB)
	Resources	Refer to Appendix A. Resource Utilization .
Design Tool Support	Lattice Implementation	IP Core v2.3.0 – Lattice Radiant™ Software 2024.2 IP Core v2.4.0 – Lattice Radiant Software 2025.1 and Lattice Propel™ Builder Software 2025.1
	Synthesis	Synopsys® Synplify Pro for Lattice, Lattice Synthesis Engine
	Simulation	Refer to the Lattice Radiant Software User Guide for the list of supported simulators.

1.3. IP Support Summary

Table 1.2. SPI Controller IP Support Readiness

Device Family	IP	System Clock Frequency (MHz)	Chip Select	SPI Clock	Features Supported	Radiant Timing Model	Hardware Validated
Avant	SPI Controller	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Advance	Yes
MachXO5-NX	SPI Controller	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
			Pulse Mode, Low Polarity	Low Polarity, With Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
			Pulse Mode, Low Polarity	High Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
			Pulse Mode, Low Polarity	High Polarity, With Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
		10	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
			Pulse Mode, Low Polarity	Low Polarity, With Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
			Pulse Mode, Low Polarity	High Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
			Pulse Mode, Low Polarity	High Polarity, With Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
		200	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
			Pulse Mode, Low Polarity	Low Polarity, With Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
			Pulse Mode, Low Polarity	High Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
			Pulse Mode, Low Polarity	High Polarity, With Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	Yes
Certus-NX	SPI Controller	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	No
CertusPro-NX	SPI Controller	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	No

Device Family	IP	System Clock Frequency (MHz)	Chip Select	SPI Clock	Features Supported	Radiant Timing Model	Hardware Validated
CrossLink-NX	SPI Controller	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	No
Certus-NX-RT	SPI Controller	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	No
CertusPro-NX-RT	SPI Controller	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	No
Certus-N2	SPI Controller	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	No

1.4. Features

Key features of the SPI Controller IP include:

- Four-wire SPI interface (SCLK, SS, MOSI, MISO)
- Configurable SPI data width (8, 16, 24, or 32 bits wide)
- Transmit FIFO and Receive FIFO with configurable depth
- Configurable number of SPI Chip Select lines (1 to 8)
- Programmable polarity for each Chip Select line
- Support for all SPI clocking modes (combination of clock polarity and clock phase)
- Programmable serial clock frequency
- Programmable clock polarity
- Independent target configuration and daisy chain configuration
- Interrupt capability
- Selectable memory-mapped interface (AHB-Lite, APB, or LMMI)

1.5. Licensing and Ordering Information

The SPI Controller IP is provided at no additional cost with the Lattice Radiant software. The IP can be fully evaluated in hardware without requiring an IP license string.

1.6. Hardware Support

Refer to the [Example Design](#) section for more information on the boards used.

1.7. 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, asserted when value is logic 0

`_i` are input signals

`_o` are output signals

`_io` are bi-directional input/output signals

1.7.3. Attribute

Names of attributes in this document are formatted in title case and italicized (*Attribute Name*).

2. Functional Description

2.1. IP Architecture Overview

The SPI Controller IP Core allows the host inside the FPGA to communicate with multiple external SPI Target devices. The data size of the SPI transaction can be configured to 8, 16, 24, or 32 bits. This IP is designed to use an internal FIFO of configurable depth to minimize the host intervention during data transfer. The SPI Controller IP Core supports all SPI clocking modes — combinations of Clock Polarity (CPOL) and Clock Phase (CPHA) to match the settings of external devices.

The SPI Controller IP provides a bridge between LMMI/AHB-Lite/APB and standard external SPI bus interfaces. The functional diagram is shown in Figure 2.1. On the external, off-chip side, the SPI Controller IP has a standard SPI bus interface. On the internal, on-chip side, the SPI Controller IP has LMMI/AHB-Lite/APB interfaces depending on the *Interface* attribute settings.

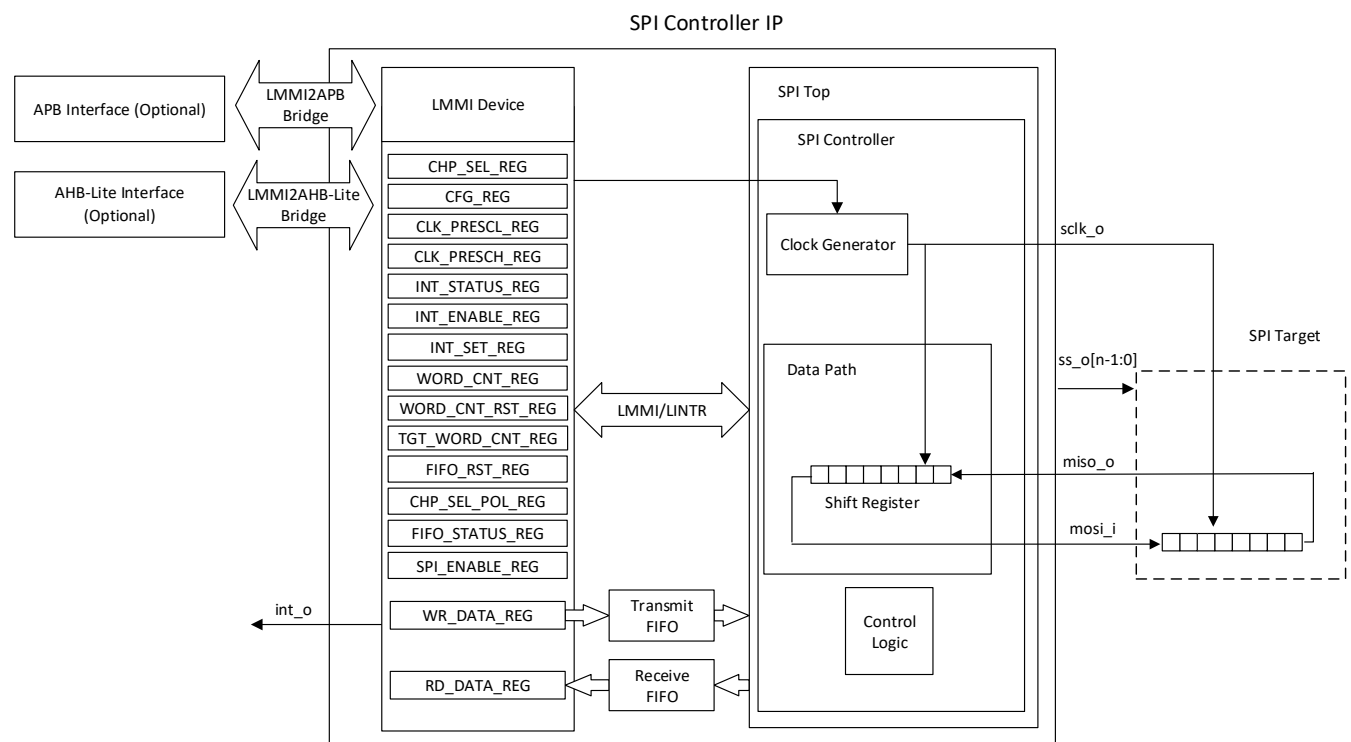


Figure 2.1. SPI Controller IP Core Block Diagram

The SPI Controller IP includes the following layers:

- APB/AHB-Lite Interface
- LMMI Device
- SPI Top

2.2. Clocking

There are three clocks for the SPI Controller IP.

- `clk_i`: system clock, used to drive the entire IP.
- `lmmi_clk`: manage clock, used to configure registers.
- `sclk_o`: output clock, used to control peripherals.

2.2.1. Clocking Overview

Figure 2.2 shows the SPI Controller IP Clock Domain Block Diagram.

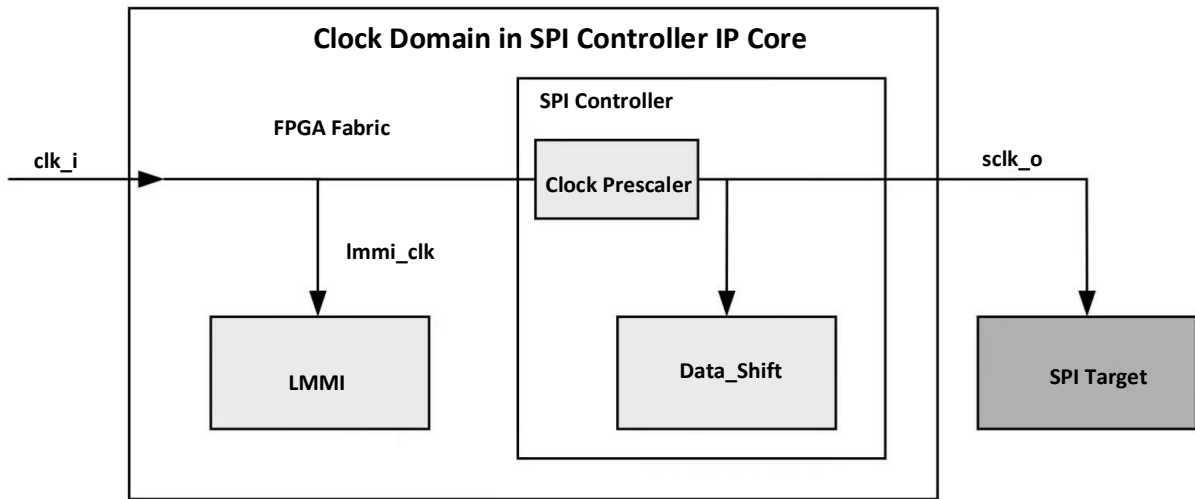


Figure 2.2. SPI Controller IP Clock Domain Block Diagram

Notes:

- `clk_i` is limited to 10-200 MHz.
- `sclk_o` is obtained by `ref_clk` using Clock Prescaler. Frequency division multiple can be 2-65535.
- SPI Actual Clock Frequency (MHz) = System Clock Frequency/Clock Prescaler × 2.

2.3. Reset

There is one reset for the SPI Controller IP.

2.3.1. Reset Overview

`rst_n_i` is an asynchronous active low reset. The reset assertion can be asynchronous but reset negation should be synchronous. When asserted, output ports and registers are forced to their reset values.

2.4. User Interfaces

Table 2.1 shows the user interfaces and supported protocols. The memory-mapped interface of SPI Controller IP Core is selected by the *Interface* attribute. It can be LMMI interface, AHB-Lite interface, or APB interface.

Table 2.1. User Interfaces and Supported Protocols

User Interface	Supported Protocols	Description
Selectable Memory-Mapped Interface	LMMI	For LMMI interface, refer to the Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039) document for information and timing diagram of the LMMI. - Immi_ready_o is always asserted, thus write and read transactions have no wait state;¹ - Read latency is one clock cycle.¹
	AHB-Lite	For AHB-Lite interface, refer to AMBA 3 AHB-Lite Protocol Specification for information and timing diagram of the APB interface. - Write transaction has no wait state;¹ - Read transaction has one wait state.¹
	APB	For APB interface, refer to AMBA 3 APB Protocol v1.0 Specification for information and timing diagram of the APB interface. - Write transaction has one wait state;¹ - Read transaction has two wait states.¹
Device Receiver/Transmitter Interface	SPI	The Serial Peripheral Interface can complete the communication between Lattice SPI Controller IP core and external SPI target devices, such as display drivers, SPI EPROMS, and analog-to-digital converters.

Note:

1. Take note of these details when checking the corresponding timing diagram in the said document.

2.5. Operation Details

2.5.1. Clock Generation

The SPI Controller IP Core generates Serial Clock (sclk_o) by dividing the system clock (clk_i) using a counter circuit that counts from 16'h0000 to the value set by Clock Prescaler Registers. The high period and low period of sclk_o is equal to {CLK_PRESCAL_REG, CLK_PRESCAL_REG} number of clk_i cycles. Thus, the frequency of sclk_o is calculated by dividing the clk_i frequency with 2 x {CLK_PRESCAL_REG, CLK_PRESCAL_REG} value.

The initial frequency of sclk_o is set through the *Desired SCLK Frequency* attribute and it can be modified by programming the Clock Prescaler Registers. It is prohibited to modify the sclk_o frequency during operation.

2.5.2. Data Transmission

To begin SPI transaction, the SPI Controller IP Core sends the clock signal and selects the target by asserting the ss_o bit for the target SPI target. Chip select is usually an active low signal but active high version is also supported. Thus, it is necessary to program the Chip Select Polarity Register with the chip select polarity of the corresponding SPI target. A four-wire SPI interface is full-duplex, both controller and target can send data at the same time through the MOSI and MISO lines respectively. During SPI transaction, the data is simultaneously transmitted (shifted out serially onto the mosi_o signal), and received (the miso_i is sampled/shifted-in). The serial clock (sclk_o) edge synchronizes the shifting and sampling of the data. For more information, see the [Clocking Modes: Clock Polarity and Clock Phase](#) section. The bit size of each data word transferred during SPI transaction is set by the *Data Width* attribute.

2.5.3. Clocking Modes: Clock Polarity and Clock Phase

The SPI Controller IP Core can select the clock polarity and clock phase. The CPOL (CFG_REG.cpol) bit sets the polarity of the clock signal during the idle state, when ss_o bits are in inactive logic based on Chip Select Polarity Register. Transitioning of ss_o bit(s) from inactive logic to active logic marks the start of the transmission. Transitioning from active logic to inactive logic marks the end of the transmission. The CPHA (CFG_REG.cpha) bit selects the clock phase. This bit selects the rising or falling clock edge used to sample and shift the data. The SPI Controller IP Core must select the clock polarity and clock phase required for the SPI target. There are four SPI clocking modes available based on the CPOL and CPHA bit selection, as shown in Table 2.2.

Table 2.2. Clocking Modes (CPOL and CPHA)

Clocking Mode	CPOL	CPHA	Action
0	0	0	The ss_o bit(s) transition to active logic shifts out a data bit. The first clock transition samples the data. Data is sampled on the sclk_o rising edge and is shifted out on the falling edge.
1	0	1	The first clock transition shifts out a data bit and the second clock transition samples the data. Data is sampled on the sclk_o falling edge and is shifted out on the rising edge.
2	1	0	The ss_o bit(s) transition to active logic shifts out a data bit. The first clock transition samples the data. Data is sampled on the sclk_o falling edge and is shifted out on the rising edge.
3	1	1	The first clock transition shifts out the data and the second clock transition samples the data. Data is sampled on the sclk_o rising edge and is shifted out on the falling edge.

The sample waveforms for the SPI clocking modes are shown in Figure 2.3, Figure 2.4, Figure 2.5, and Figure 2.6. In these examples, the *Number of Chip Select Lines* attribute is set to 1. Thus, ss_o is only one bit and is active low (CHP_SEL_POL_REG.ssp1=1'b0).

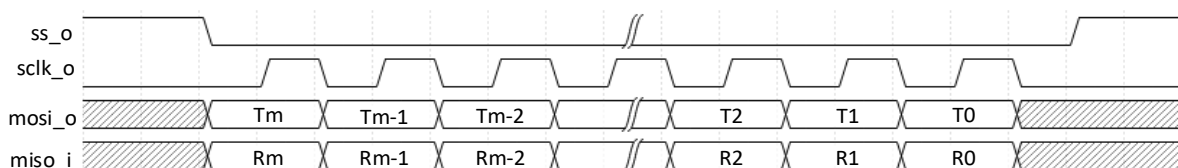


Figure 2.3. Clocking Mode 0 (CPOL=0, CPHA=0)

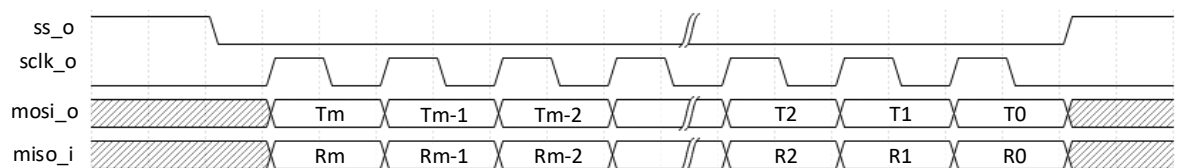


Figure 2.4. Clocking Mode 1 (CPOL=0, CPHA=1)

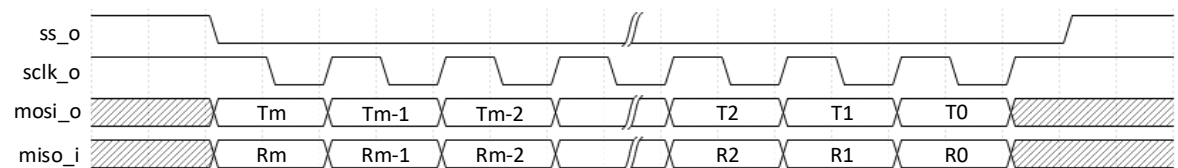


Figure 2.5. Clocking Mode 2 (CPOL=1, CPHA=0)

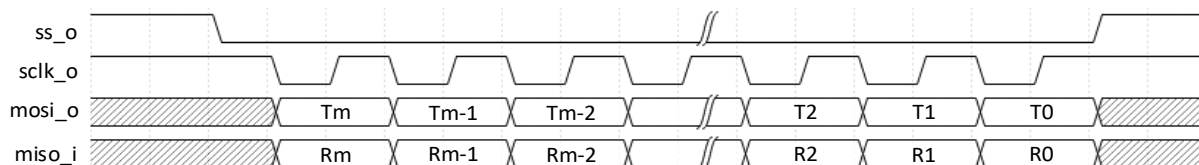


Figure 2.6. Clocking Mode 3 (CPOL=1, CPHA=1)

2.5.4. Chip Select Pulse Mode

CFG_REG.ssnp controls the behavior of ss_o signal. When CFG_REG.ssnp=1'b1, ss_o pulses in between consecutive data words, as shown in Figure 2.7. In this example, each data word is 8-bit (Data Width=8).

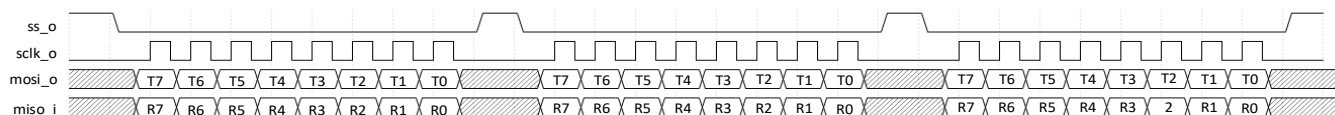


Figure 2.7. Timing Diagram for a Transfer with SSNP Set to 1

Figure 2.8 shows a sample timing diagram for data transfer with CFG_REG.ssnp=1'b0, and ss_o does not pulse in between consecutive data words.

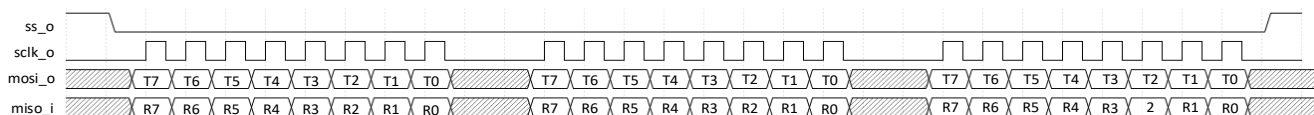


Figure 2.8. Timing Diagram for a Transfer with SSNP Set to 0

Note: When Transmit FIFO becomes empty within a transfer, ss_o de-asserts even if CFG_REG.ssnp=1'b0. Ensure that the next data is available in the Transmit FIFO before the transfer of each data word completes. You may confirm that Transmit FIFO is not empty by reading from FIFO_STATUS_REG.tx_fifo_empty.

2.5.5. SPI Enable

When CFG_REG.spi_en is set to 1, it enables the control of chip select line using SPI_ENABLE_REG. As shown in Figure 2.9, the ss_o is to be asserted when SPI transfer is enabled using SPI_ENABLE_REG. No clock and data is to be transmitted until data is written to the Transmit FIFO. The chip select line remains active even after Transmit FIFO is empty unless it is disabled using SPI_ENABLE_REG.

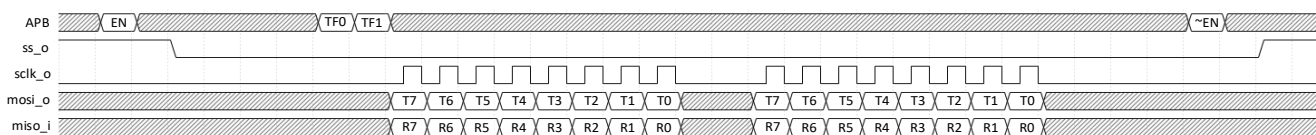


Figure 2.9. SPI Enable Register Enable and Disable

SPI transfer can also be enabled after writing data to Transmit FIFO, as shown in Figure 2.10. Or the Transmit FIFO may remain empty for a long time before writing the next data, as shown in Figure 2.11. For all figures presented in this section, CFG_REG.ssnp is set to 0.

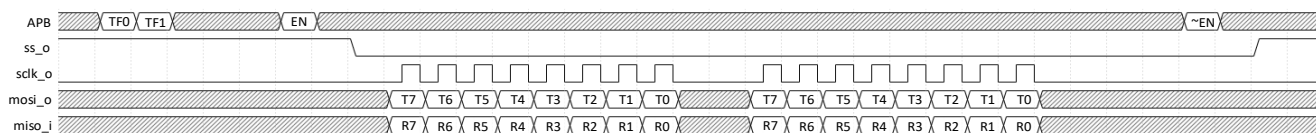


Figure 2.10. SPI Enable after Writing Data to Transmit FIFO

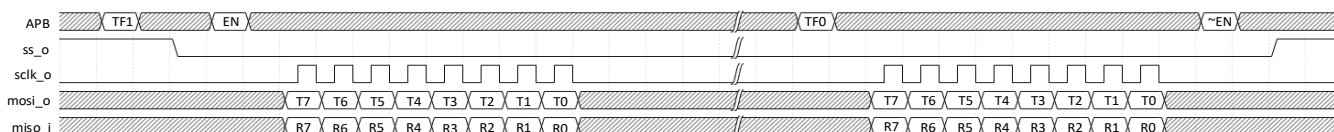


Figure 2.11. Long Gap between Writing Data to Transmit FIFO

2.6. Independent Target Configuration

In the independent target configuration, there is an independent chip select line for each target, as shown in Figure 2.12. This is the way SPI is normally used. The controller asserts only one chip select line at a time.

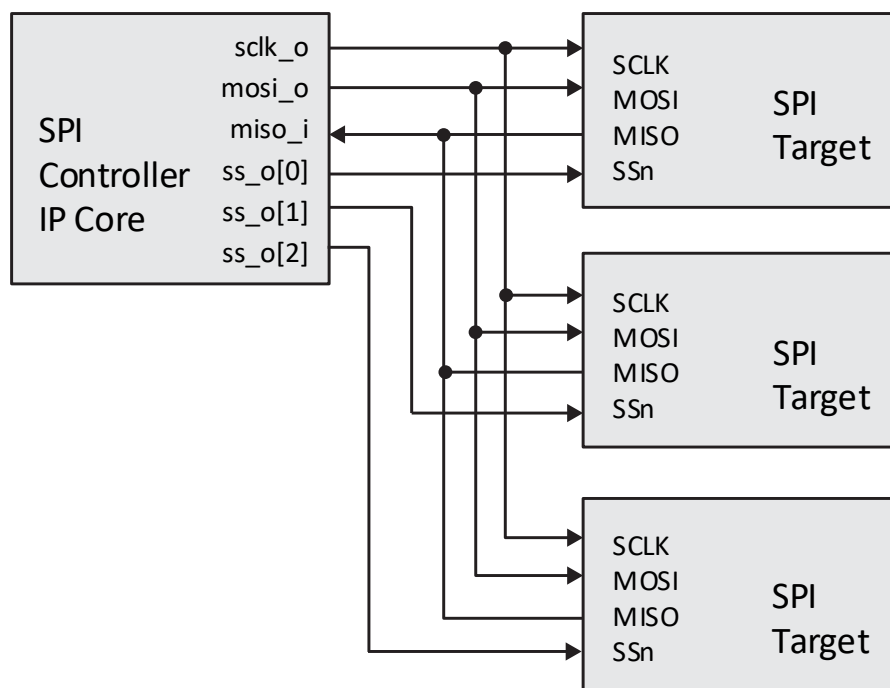


Figure 2.12. Independent Target Configuration

Since the MISO pins of the SPI target devices are connected together, they are required to be tri-state pins with high, low, or high-impedance state. The high-impedance output must be applied when the target is not selected.

2.7. Daisy Chain Configuration

SPI Controller IP Core also supports daisy chain configuration. As shown in Figure 2.13, the first target output is connected to the second target input. The SPI port of each target is designed to send out during the second group of clock pulses an exact copy of the data it received during the first group of clock pulses. The whole chain acts as a communication shift register. Each target copies input to output in the next clock cycle until active low SS line goes high.

Normally, a single chip select line from the SPI Controller is required for daisy chain configuration. Since SPI Controller IP Core can assert multiple target select lines, that is multiple ss_o bits, through the Chip Select Register, it is possible to perform daisy chain with separate SS line for each target.

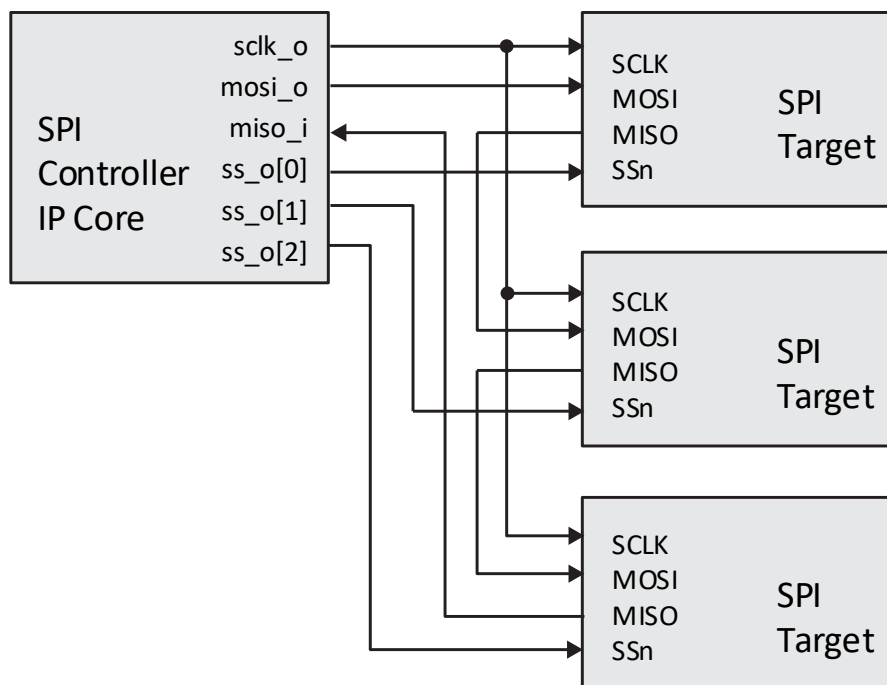


Figure 2.13. Daisy Chain Configuration

2.8. Programming Flow

Below are the recommended steps for performing the SPI transaction. Host can use interrupt mode, polling mode, or a combination of these two modes when transmitting and receiving SPI transactions.

2.8.1. Initialization

The following SPI Controller registers should be set properly before performing SPI transaction:

CHP_SEL_REG – Set 1'b1 to the bit for the corresponding target. Set 1'b0 to other bits.

CHP_SEL_POL_REG – Can be configured once after reset since this setting is usually fixed.

CLK_PRESCAL_REG – Set based on target sclk_o frequency.

CLK_PRESCH_REG – Set based on target sclk_o frequency.

The host device needs to update the above registers only when SPI Controller is switching to different target device. No need to perform initialization again, if the next transaction is for the currently selected target device.

2.8.2. Transmit/Receive Operation Interrupt Mode

Assuming that the module is not currently performing any operation.

1. Set the CFG_REG fields according to the intended SIP Target settings: cpha, cpol, ssnp, and lsb_first. Set the only_write field based on the current transaction. If only_write is 1'b0, SPI Controller performs both transmit and receive operations (full-duplex). On the other hand, if only_write is 1'b1, SPI Controller IP Core performs Transmit operation only.
2. Enable target interrupts in INT_ENABLE_REG.
3. Set the target n-word data in TGT_WORD_CNT_REG according to the number of words to transfer.
4. Reset WORD_CNT_REG by writing 8'hFF to WORD_CNT_RST_REG.
5. Write n-word data to WR_DATA_REG, amounting to less than or equal to Transmit FIFO depth.
If target n-word is greater than the Transmit FIFO depth, check the interrupt for Transmit FIFO full, INT_STATUS_REG.tx_fifo_full_int, before writing data to WR_DATA_REG to avoid data loss.
6. Clear the pending interrupts as needed.

7. Wait for interrupt, `int_o`, then read `INT_STATUS_REG`. Check if the pending interrupt is `tr_cmp_int`. This indicates that the SPI target has completed transmitting the target n-word data.
8. Clear the pending interrupt in `INT_STATUS_REG`.
9. If `CFG_REG.only_write = 1'b0`, read the n-word data in `RD_DATA_REG`.

2.8.3. Transmit/Receive Operation Polling Mode

Assuming that the module is not currently performing any operation.

1. Set the `CFG_REG` fields according to the intended SPI Target settings: `cpha`, `cpol`, `ssnp`, and `lsb_first`. Set the `only_write` field based on the current transaction. If `only_write` is `1'b0`, SPI Controller performs both transmit and receive operations (full-duplex). On the other hand, if `only_write` is `1'b1`, SPI Controller IP Core performs Transmit operation only.
2. Set the target n-word data in `TGT_WORD_CNT_REG` according to the number of words to transfer.
3. Reset `WORD_CNT_REG` by writing `8'hFF` to `WORD_CNT_RST_REG`.
4. Write n-word data to `WR_DATA_REG`, amounting to less than or equal to Transmit FIFO depth.
If the target n-word is greater than the Transmit FIFO depth, read `FIFO_STATUS_REG.tx_fifo_full` before writing data to `WR_DATA_REG`.
5. Poll `WORD_CNT_REG` to check the number of n-word data transmitted through the SPI interface. If the `WORD_CNT_REG` matches the target n-word data based on SPI command and address, target SPI transaction is now complete.
6. If `CFG_REG.only_write = 1'b0`, read the n-word data in `RD_DATA_REG`.
Read `FIFO_STATUS_REG.rx_fifo_empty` to check if the Receive FIFO is empty.

3. IP Parameter Description

The configurable attributes of the SPI Controller IP are shown in the following table. You can configure the IP by setting the attributes accordingly in the IP Catalog's 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
General		
Interface	LMMI, APB, AHBL	Selects memory-mapped interface for register access by the host. The unselected interfaces are not available in the generated IP.
Number of Chip Select Lines	1 –8	Specifies the number of chip-select signal bits. Each bit used for selecting an SPI target device.
First Transmitted Bit	MSB , LSB	Specifies the shifting direction of data bits into and out of the SPI interface by setting the reset value of CFG_REG.lsb_first. MSB – Reset value is 1'b0. LSB – Reset value is 1'b1.
Read-Write Operation	Read and Write , Only Write	Enables or disables the sampling or reading of miso_i signal during transaction by setting the reset value of CFG_REG.only_write. Read and Write – Reset value is 1'b0. Only Write – Reset value is 1'b1.
Data Width	8, 16, 24, 32	Specifies the bit size of each SPI transaction. This also sets the value of FIFO width. This cannot be changed at run-time. The setting value of this attribute can be identified by reading CFG_REG.ds. 8 – CFG_REG.ds = 2'b00 16 – CFG_REG.ds = 2'b01 24 – CFG_REG.ds = 2'b10 32 – CFG_REG.ds = 2'b11
Chip Select Pulse Mode	Non-Pulse (0), Pulse (1)	Specifies the reset value of CFG_REG.ssnp. Non-Pulse – CFG_REG.ssnp = 1'b0 Pulse – CFG_REG.ssnp = 1'b1
SPI Clock Polarity	0 (Active Low), 1 (Active High)	Specifies the reset value of CFG_REG.cpol.
SPI Clock Phase	0 , 1	Specifies the reset value of CFG_REG.cpha.
Chip Select Polarity (0x)	00 –FF	Specifies the reset value of CHP_SEL_POL_REG. Each bit specifies the polarity of the corresponding ss_o bit. The size depends on <i>Number of Chip Select Lines</i> .
SPI Enable Register	Checked, Unchecked	Specifies the reset value of CFG_REG.spi_en. Refer to the SPI Enable section for more details.
Include I/O Buffer	Checked, Unchecked	Specifies whether SPI signals are passed through bidirectional buffer or directly connected to the top ports of the IP.

Attribute	Selectable Values	Description
FIFO		
FIFO Width	8, 16, 24, 32	Specifies the bit width of each data word of the internal FIFO. This is not editable. It takes the value from <i>Data Width</i> .
FIFO Depth	16 , 32, 64, 128, 256, 512	Specifies the number of FIFO levels. Only power of two values are allowed.
Implementation of FIFO	EBR , LUT	Selects the FPGA resource that is used to implement the FIFO.
TX FIFO Almost Empty Flag	3 , 1–(<i>FIFO Depth</i> – 1)	Specifies the trigger level for asserting INT_STATUS_REG.tx_fifo_aempty_int. Refer to the Interrupt Status Register (INT_STATUS_REG) section for more details.
RX FIFO Almost Full Flag	12 , 1–(<i>FIFO Depth</i> – 1)	Specifies the trigger level for asserting INT_STATUS_REG.rx_fifo_afull_int. Refer to the Interrupt Status Register (INT_STATUS_REG) section for more details.
Clock		
System Clock Frequency (MHz)	50 , 10–200	Specifies the frequency of system clock.
Desired SCLK Frequency (MHz)	0.1 , 0.01–50	Used for calculating <i>Clock Prescaler</i> value. Must be less than or equal to <i>System Clock Frequency</i> /4.
Clock Prescaler	250 , 2–65535	Sets the frequency of the sclk_o signal by specifying the default value of Clock Prescaler registers. This is calculated by dividing <i>System Clock Frequency</i> with $2 \times \text{Desired SCLK Frequency}$. Both the floor value and ceiling value are calculated. The value that generates <i>SPI Actual Clock Frequency</i> closest to <i>Desired SCLK Frequency</i> is selected.
SPI Actual Clock Frequency (MHz)	0.1 , Output Value	The frequency of sclk_o. This is calculated by dividing <i>System Clock Frequency</i> with $2 \times \text{Clock Prescaler}$.

4. Signal Description

Table 4.1 lists the input and output signals for the SPI Controller IP.

Table 4.1. SPI Controller IP Signal Description

Port	Width	Type	Output Reset Value	Description
Clock and Reset				
clk_i	1	Input	—	System clock The frequency range is limited to 10–200 MHz.
rst_n_i	1	Input	—	Asynchronous active low reset Reset assertion can be asynchronous but reset de-assertion should be synchronous. When asserted, output ports and registers are forced to their reset values.
Interrupt Port				
int_o	1	Output	1'b0	Interrupt signal
SPI Interface				
sclk_o	1	Output	Idle value as specified by <i>SPI Clock Polarity</i>	Serial clock Generated by the SPI Controller to synchronize data transfers.
mosi_o	1	Output	High impedance	Controller Out, Target In (MOSI) signal Transfers data from the Controller to the Target.
miso_i	1	Input	—	Controller In, Target Out (MISO) signal Transfers data to the Controller from the Target.
ss_o	<i>Number of Chip Select Lines²</i>	Output	0	Chip Select signal Asserted by the Controller to begin data transfer. There are up to eight Chip Select outputs for selecting eight different SPI lines.
LMMI Interface¹				
lmmi_request_i	1	Input	—	Starts transaction.
lmmi_wr_rdn_i	1	Input	—	Write = 1'b1, Read = 1'b0.
lmmi_offset_i	4	Input	—	Register offset, starting at offset 0.
lmmi_wdata_i	<i>Data Width²</i>	Input	—	Output data bus
lmmi_rdata_o	<i>Data Width²</i>	Output	0	Input data bus
lmmi_rdata_valid_o	1	Output	1'b0	Read transaction is complete and lmmi_rdata_o contains valid data.
lmmi_ready_o	1	Output	Tied to 1'b1	The IP is ready to receive a new transaction.
AHB-Lite Interface¹				
ahbl_hsel_i	1	Input	—	AHB-Lite Select signal Indicates the device is selected and transfer is required.
ahbl_hready_i	1	Input	—	AHB-Lite Ready Input signal Indicates data phase of previous transfer is completed.
ahbl_haddr_i	6	Input	—	AHB-Lite Address signal
ahbl_hburst_i	3	Input	—	AHB-Lite Burst Type signal Indicates if the transfer is a single transfer or forms part of a burst.
ahbl_hsize_i	3	Input	—	AHB-Lite Transfer Size signal Indicates the size of the transfer that is a byte, half-word or word.

Port	Width	Type	Output Reset Value	Description
ahbl_hmastlock_i	1	Input	—	AHB-Lite Lock signal This signal is unused.
ahbl_hprot_i	4	Input	—	AHB-Lite Protection Control signal This signal is unused.
ahbl_htrans_i	2	Input	—	AHB-Lite Transfer Type signal Indicates the transfer type of the current transfer.
ahbl_hwrite_i	1	Input	—	AHB-Lite Direction signal Write = High, Read = Low.
ahbl_hwdata_i	32	Input	—	AHB-Lite Write Data signal
ahbl_hreadyout_o	1	Output	1'b1	AHB-Lite Ready Output signal Indicates transfer completion.
ahbl_hrdata_o	32	Output	0	AHB-Lite Read Data signal
ahbl_hresp_o	1	Output	Tied to 1'b0	AHB-Lite Transfer Response signal
APB Interface¹				
apb_psel_i	1	Input	—	APB Select signal Indicates that the IP is selected and a data transfer is required.
apb_paddr_i	6	Input	—	APB Address signal
apb_pwdata_i	32	Input	—	APB Write Data signal
apb_pwrite_i	1	Input	—	APB Direction signal Write = 1, Read = 0
apb_penable_i	1	Input	—	APB Enable signal Indicates the second and subsequent cycles of an APB transfer.
apb_pready_o	1	Output	1'b0	APB Ready signal Indicates transfer completion. The IP uses this signal to extend an APB transfer.
apb_pslverr_o	1	Output	Tied to 1'b0	APB Error signal Indicates a transfer failure.
apb_prdata_o	32	Output	0	APB Read data signal Bits [31:8] are not used.

Notes:

1. Only one of the three interfaces is available as selected by the *Interface* attribute.
2. The bit width of some signals is set by the attribute. Refer to [Table 3.1](#) for more information on the attribute.

5. Register Description

5.1. Overview

Table 5.1 lists the address map and specifies the registers available. The offset of each register is dependent on the *Interface* attribute setting as follows:

- Interface selected to be LMMI: the offset increments by one.
- Interface selected to be either AHBL or APB: the offset increments by four to allow easy interfacing with the processor and system buses. In this mode, each register is 32 bits wide wherein the upper unused bits are reserved and the lower bits are described in the following sections.

Table 5.1. Summary of SPI Controller IP Core Registers

Offset LMMI	Offset APB/AHBL	Register Name	Access Type	Description
0x0	0x00	WR_DATA_REG ¹	WO	Write Data Register
0x0	0x00	RD_DATA_REG ¹	RO	Read Data Register
0x1	0x04	CHP_SEL_REG	RW	Chip Select Register
0x2	0x08	CFG_REG	RW	Configuration Register
0x3	0x0C	CLK_PRESCAL_REG	RW	Clock Pre-scaler Low Register
0x4	0x10	CLK_PRESCAL_REG	RW	Clock Pre-scaler High Register
0x5	0x14	INT_STATUS_REG	RW1C	Interrupt Status Register
0x6	0x18	INT_ENABLE_REG	RW	Interrupt Enable Register
0x7	0x1C	INT_SET_REG	WO	Interrupt Set Register
0x8	0x20	WORD_CNT_REG	RO	Word Count Register
0x9	0x24	WORD_CNT_RST_REG	WO	Word Count Reset Register
0xA	0x28	TGT_WORD_CNT_REG	RW	Target Word Count Register
0xB	0x2C	FIFO_RST_REG	WO	FIFO Reset Register
0xC	0x30	CHP_SEL_POL_REG	RW	Chip Select Polarity Register
0xD	0x34	FIFO_STATUS_REG	RO	FIFO Status Register
0xE	0x38	SPI_ENABLE_REG	RW	SPI Enable Register
0xF	0x3C	Reserved	RSVD	Reserved.

Note:

- RD_DATA_REG and WR_DATA_REG share the same offset. Write access to this offset goes to WR_DATA_REG while read access goes to RD_DATA_REG.

Table 5.2 defines the register access types.

Table 5.2. 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 the register bit clears the bit to 1'b0. Writing 1'b0 on the register bit is ignored.
Reserved	RSVD	Returns 0	Ignores write access

5.2. Write Data Register (WR_DATA_REG)

The WR_DATA_REG is the interface to the Transmit FIFO. The bit width of this register is set by the *Data Width* attribute.

Table 5.3. Write Data Register

Field	Name	Description	Access	Default
[<i>Data Width</i> –1:0]	tx_fifo	Writing to WR_DATA_REG pushes a word to the Transmit FIFO. When the Transmit FIFO has at least one data word, the SPI Controller IP Core initiates an SPI transaction, which pops one data word and sends it through the SPI interface. Writing to this register is recommended only when other registers have been initialized. The start of SPI transaction changes if the <i>SPI Enable Register</i> attribute is checked. Refer to the SPI Enable section for more information. When writing to WR_DATA_REG, the host should ensure that Transmit FIFO is not full. This can be done by enabling and observing the interrupt, INT_STATUS_REG.tx_fifo_full_int, or monitoring the Transmit FIFO status, FIFO_STATUS_REG.tx_fifo_full. When reset is performed, the contents of Transmit FIFO are not reset but the FIFO control logic is reset. Thus, the content is not guaranteed.	WO	Not guaranteed

5.3. Read Data Register (RD_DATA_REG)

The RD_DATA_REG is the interface to the Receive FIFO. The bit width of this register is set by the *Data Width* attribute.

Table 5.4. Read Data Register

Field	Name	Description	Access	Default
[<i>Data Width</i> –1:0]	rx_fifo	Reading from RD_DATA_REG pops a word from the Receive FIFO. During an SPI transaction, the received data in MISO is accumulated and pushed to Receive FIFO if the length of the accumulated data matches the Data Width. If the Receive FIFO is empty and a new word is added, INT_STATUS_REG.rx_fifo_ready_int asserts. The host should ensure that Receive FIFO has data before reading RD_DATA_REG. Data is not guaranteed when this register is read during the Receive FIFO empty condition. On the other hand, if Receive FIFO is full and this register is not read but SPI Controller continues to receive additional data, the new word is not pushed into the Receive FIFO and is lost. To prevent these two conditions, monitor Receive FIFO status, FIFO_STATUS_REG.rx_fifo_full and rx_fifo_empty. Similar to Transmit FIFO, the contents of Receive FIFO are not guaranteed when reset is performed.	RO	Not guaranteed

5.4. Chip Select Register (CHP_SEL_REG)

The bit width of CHP_SEL_REG is set by the *Number of Chip Select Lines* attribute. The maximum supported size is eight. Setting a bit to 1 means the SPI Controller IP Core selects the corresponding target in the next transaction. After the data is written to WR_DATA_REG, the SPI Controller IP Core asserts the corresponding bit of the ss_o signal, triggering the start of SPI transaction.

Table 5.5. Chip Select Register

Field	Name	Description	Access	Default
[7]	ss8	Chip Select 8. It selects Target 8. ss_o[7] asserts during SPI transaction. 1'b0 – Target 8 is not selected. 1'b1 – Target 8 is selected.	RW	1'b0
[6]	ss7	Chip Select 7. It selects Target 7. ss_o[6] asserts during SPI transaction. 1'b0 – Target 7 is not selected. 1'b1 – Target 7 is selected.	RW	1'b0
[5]	ss6	Chip Select 6. It selects Target 6. ss_o[5] asserts during SPI transaction. 1'b0 – Target 6 is not selected. 1'b1 – Target 6 is selected.	RW	1'b0
[4]	ss5	Chip Select 5. It selects Target 5. ss_o[4] asserts during SPI transaction. 1'b0 – Target 5 is not selected. 1'b1 – Target 5 is selected.	RW	1'b0
[3]	ss4	Chip Select 4. It selects Target 4. ss_o[3] asserts during SPI transaction. 1'b0 – Target 4 is not selected. 1'b1 – Target 4 is selected.	RW	1'b0
[2]	ss3	Chip Select 3. It selects Target 3. ss_o[2] asserts during SPI transaction. 1'b0 – Target 3 is not selected. 1'b1 – Target 3 is selected.	RW	1'b0
[1]	ss2	Chip Select 2. It selects Target 2. ss_o[1] asserts during SPI transaction. 1'b0 – Target 2 is not selected. 1'b1 – Target 2 is selected.	RW	1'b0
[0]	ss1	Chip Select 1. It selects Target 1. ss_o[0] asserts during SPI transaction. 1'b0 – Target 1 is not selected. 1'b1 – Target 1 is selected.	RW	1'b1

5.5. Configuration Register (CFG_REG)

The CFG_REG controls the behavior of the SPI transaction. It is prohibited to modify this register during SPI transaction or after data is written to WR_DATA_REG. The host should set this register according to the requirement of the target.

Table 5.6. Configuration Register

Field	Name	Description	Access	Default ¹
[7]	spi_en	SPI Enable Enables the SPI_ENABLE_REG for chip select line control. 1'b0 – SPI_ENABLE_REG is not used. 1'b1 – SPI_ENABLE_REG is used.	RW	<i>SPI Enable Register</i>
[6]	lsb_first	LSB First Specifies the shifting order of data bits when transmitting or receiving data. 1'b0 – MSB is transmitted/received first. 1'b1 – LSB is transmitted/received first.	RW	<i>First Transmitted Bit</i>
[5]	only_write	Only Write Disables data reception during SPI transaction. 1'b0 – Data reception is enabled during SPI transaction. 1'b1 – Data reception is disabled during SPI transaction. Data from SPI Target is ignored.	RW	<i>Read-Write Operation</i>
[4:3]	ds	Data Size <i>Data Width</i> attribute defines the value of this field. This field specifies the size of data in each SPI transaction. 2'b00 – Data size is 8 bits. 2'b01 – Data size is 16 bits. 2'b10 – Data size is 24 bits. 2'b11 – Data size is 32 bits.	RO	<i>Data Width</i>
[2]	ssnp	Chip Select Pulse Mode This bit allows the SPI Controller IP Core to generate ss_o pulse between consecutive data words when doing continuous transfers. Regardless of this setting, selected ss_o is always asserted during active data transfer. 1'b0 – Do not generate ss_o pulse between consecutive data words during SPI transaction. 1'b1 – Generates ss_o pulse between consecutive data words during SPI transaction.	RW	<i>Chip Select Pulse Mode</i>
[1]	cpol	Clock Polarity Specifies the logic level of sclk_o when idle. 1'b0 – sclk_o is set to 1'b0 when idle. 1'b1 – sclk_o is set to 1'b1 when idle.	RW	<i>SPI Clock Polarity</i>
[0]	cpha	Clock Phase Specifies clock edge for shifting out data in mosi_o and sampling data in miso_i. 1'b0 – Data from miso_i sampled on the rising edge and shifted out to mosi_o on the falling edge. Also, the first clock transition is the first data capture edge. 1'b1 – Data from miso_i sampled on the rising edge and shifted out to mosi_o on the falling edge. Also, the second clock transition is the first data capture edge.	RW	<i>SPI Clock Phase</i>

Note:

1. Refer to [Table 3.1](#) for the attribute values.

5.6. Clock Prescaler Registers (CLK_PRESC_L_REG, CLK_PRESC_H_REG)

The default values of Clock Prescaler Registers (CLK_PRESC_L_REG, CLK_PRESC_H_REG) are defined by the *Clock Prescaler* attribute.

Table 5.7. Clock Prescaler Lower Register

Field	Name	Description	Access	Default
[7:0]	clk_presc_low	Refer to the Clock Generation section for details. This register should not be modified during operation.	RW	<i>Clock Prescaler</i> [7:0]

Table 5.8. Clock Prescaler Higher Register

Field	Name	Description	Access	Default
[7:0]	clk_presc_high	Refer to the Clock Generation section for details. This register should not be modified during operation.	RW	<i>Clock Prescaler</i> [15:8]

5.7. Interrupt Status Register (INT_STATUS_REG)

The Interrupt Status Register contains all the interrupts currently pending in the SPI Controller IP Core. When an interrupt bit asserts, it remains asserted until it is cleared by the host by writing 1'b1 to the corresponding bit.

The interrupt status bits are independent of the interrupt enable bits. This means that status bits may indicate pending interrupts, even though those interrupts are disabled in the Interrupt Enable Register. Refer to the [Interrupt Enable Register \(INT_ENABLE_REG\)](#) section for details. The logic which handles interrupts should mask (bitwise and logic) the contents of INT_STATUS_REG and INT_ENABLE_REG registers to determine the interrupts to service. The int_o interrupt signal is asserted whenever both an interrupt status bit and the corresponding interrupt enable bit are set.

Table 5.9. Interrupt Status Register

Field	Name	Description	Access	Default
[7]	tr_cmp_int	Transfer Complete Interrupt Status This interrupt status bit asserts when the number of transmitted words, through the SPI interface, reaches the TGT_WORD_CNT_REG.target_word_cnt setting value. 1'b0 – No interrupt 1'b1 – Interrupt pending	RW1C	1'b0
[6]	reserved	Reserved	RSVD	—
[5]	tx_fifo_full_int	Transmit FIFO Full Interrupt Status This interrupt status bit asserts when Transmit FIFO changes from not full state to full state. 1'b0 – No interrupt 1'b1 – Interrupt pending	RW1C	1'b0
[4]	tx_fifo_aempty_int	Transmit FIFO Almost Empty Interrupt Status This interrupt status bit asserts when the amount of data words in Transmit FIFO changes from <i>TX FIFO Almost Empty Flag + 1</i> to <i>TX FIFO Almost Empty Flag</i> . 1'b0 – No interrupt 1'b1 – Interrupt pending	RW1C	1'b0
[3]	tx_fifo_empty_int	Transmit FIFO Empty Interrupt Status This interrupt status bit asserts when the last data in Transmit FIFO is popped-out, causing the FIFO to become empty. 1'b0 – No interrupt 1'b1 – Interrupt pending	RW1C	1'b0
[2]	rx_fifo_full_int	Receive FIFO Full Interrupt Status This interrupt status bit asserts when RX FIFO full status changes from not full to full state. 1'b0 – No interrupt 1'b1 – Interrupt pending	RW1C	1'b0
[1]	rx_fifo_afull_int	Receive FIFO Almost Full Interrupt Status This interrupt status bit asserts when the amount of data words in Receive FIFO changes from <i>RX FIFO Almost Full Flag – 1</i> to <i>RX FIFO Almost Full Flag</i> . 1'b0 – No interrupt 1'b1 – Interrupt pending	RW1C	1'b0
[0]	rx_fifo_ready_int	Receive FIFO Ready Interrupt Status This interrupt status bit asserts when Receive FIFO is empty and receives a data word from SPI interface. 1'b0 – No interrupt 1'b1 – Interrupt pending	RW1C	1'b0

5.8. Interrupt Enable Register (INT_ENABLE_REG)

The Interrupt Enable Register contains the interrupt enable bits corresponding to the interrupt status bits. It does not affect the contents of the Interrupt Status Register. If an INT_STATUS_REG bits asserts and the corresponding INT_ENABLE_REG bit is 1'b1, the interrupt signal int_o asserts.

Table 5.10. Interrupt Enable Register

Field	Name	Description	Access	Default
[7]	tr_cmp_en	Transfer Complete Interrupt Enable Interrupt enable bit corresponding to Transfer Complete Interrupt Status. 0 – Interrupt disabled 1 – Interrupt enabled	RW	1'b0
[6]	reserved	Reserved	RSVD	–
[5]	tx_fifo_full_en	Transmit FIFO Full Interrupt Enable Interrupt enable bit corresponding to Transmit FIFO Full Interrupt Status. 0 – Interrupt disabled 1 – Interrupt enabled	RW	1'b0
[4]	tx_fifo_aempty_en	Transmit FIFO Almost Empty Interrupt Enable Interrupt enable bit corresponding to Transmit FIFO Almost Empty Interrupt Status. 0 – Interrupt disabled 1 – Interrupt enabled	RW	1'b0
[3]	tx_fifo_empty_en	Transmit FIFO Empty Interrupt Enable Interrupt enable bit corresponding to Transmit FIFO Empty Interrupt Status. 0 – Interrupt disabled 1 – Interrupt enabled	RW	1'b0
[2]	rx_fifo_full_en	Receive FIFO Full Interrupt Enable Interrupt enable bit corresponding to Receive FIFO Full Interrupt Status. 0 – Interrupt disabled 1 – Interrupt enabled	RW	1'b0
[1]	rx_fifo_afull_en	Receive FIFO Almost Full Interrupt Enable Interrupt enable bit corresponding to Receive FIFO Almost Full Interrupt Status. 0 – Interrupt disabled 1 – Interrupt enabled	RW	1'b0
[0]	rx_fifo_ready_en	Receive FIFO Ready Interrupt Enable Interrupt enable bit corresponding to Receive FIFO Ready Interrupt Status. 0 – Interrupt disabled 1 – Interrupt enabled	RW	1'b0

5.9. Interrupt Set Register (INT_SET_REG)

The Interrupt Set Register is intended for testing purposes only. Writing 1'b1 to a register bit in INT_SET_REG asserts the corresponding interrupt status bit in INT_STATUS_REG. Writing 1'b0 is ignored.

Table 5.11. Interrupt Set Register

Field	Name	Description	Access	Default
[7]	tr_cmp_set	Transfer Complete Interrupt Set Interrupt set bit corresponding to Transfer Complete Interrupt Status. 0 – No action 1 – Asserts INT_STATUS_REG.tr_cmp_int	WO	1'b0
[6]	reserved	Reserved	RSVD	—
[5]	tx_fifo_full_set	Transmit FIFO Full Interrupt Set Interrupt set bit corresponding to Transmit FIFO Full Interrupt Status. 0 – No action 1 – Asserts INT_STATUS_REG.tx_fifo_full_int	WO	1'b0
[4]	tx_fifo_aempty_set	Transmit FIFO Almost Empty Interrupt Set Interrupt set bit corresponding to Transmit FIFO Almost Empty Interrupt Status. 0 – No action 1 – Asserts INT_STATUS_REG.tx_fifo_aempty_int	WO	1'b0
[3]	tx_fifo_empty_set	Transmit FIFO Empty Interrupt Set Interrupt set bit corresponding to Transmit FIFO Empty Interrupt Status. 0 – No action 1 – Asserts INT_STATUS_REG.tx_fifo_empty_int	WO	1'b0
[2]	rx_fifo_full_set	Receive FIFO Full Interrupt Set Interrupt set bit corresponding to Receive FIFO Full Interrupt Status. 0 – No action 1 – Asserts INT_STATUS_REG.rx_fifo_full_int	WO	1'b0
[1]	rx_fifo_afull_set	Receive FIFO Almost Full Interrupt Set Interrupt set bit corresponded to Receive FIFO Almost Full Interrupt Status. 0 – No action 1 – Asserts INT_STATUS_REG.rx_fifo_afull_int	WO	1'b0
[0]	rx_fifo_ready_set	Receive FIFO Ready Interrupt Set Interrupt set bit corresponded to Receive FIFO Ready Interrupt Status. 0 – No action 1 – Asserts INT_STATUS_REG.rx_fifo_ready_int	WO	1'b0

5.10. Word Count Register (WORD_CNT_REG)

Table 5.12. Word Count Register

Field	Name	Description	Access	Default
[7:0]	word_cnt	The number of words sent by the SPI Controller IP Core to the SPI Target is reflected in the Word Count Register. This register is updated after the entire data word is sent.	RO	8'h00

5.11. Word Count Reset Register (WORD_CNT_RST_REG)

Table 5.13. Word Count Reset Register

Field	Name	Description	Access	Default
[7:0]	word_cnt_rst	Writing 8'hFF to this register resets the WORD_CNT_REG.word_cnt to 8'h00.	WO	8'h00

5.12. Target Word Count Register (TGT_WORD_CNT_REG)

Table 5.14. Target Word Count Register

Field	Name	Description	Access	Default
[7:0]	target_word_cnt	The Target Word Count Register is used for Transfer Complete interrupt generation when the target word count is achieved.	RW	8'h00

5.13. FIFO Reset Register (FIFO_RST_REG)

The FIFO Reset Register is used for clearing the Transmit and Receive FIFO.

Table 5.15. FIFO Reset Register

Field	Name	Description	Access	Default
[7:2]	reserved	Reserved	—	—
[1]	tx_fifo_rst	Transmit FIFO Reset This bit resets Transmit FIFO for one clock cycle only. 1'b0 – No action 1'b1 – Resets Transmit FIFO	WO	1'b0
[0]	rx_fifo_rst	Receive FIFO Reset This bit resets Receive FIFO for one clock cycle only. 1'b0 – No action 1'b1 – Resets Receive FIFO	WO	1'b0

5.14. Chip Select Polarity Register (CHP_SEL_POL_REG)

Each bit of the Chip Select Polarity Register defines the polarity of the chip select signal for the corresponding target. The host should set this according to the polarity of the chip select signal for the corresponding target.

Table 5.16. Chip Select Polarity Register

Field	Name	Description	Access	Default
[7]	ssp8	Chip Select Polarity 8 Specifies the polarity of chip select signal for target 8. 1'b0 – Target select 8 signal - ss_o[7] is active low. 1'b1 – Target select 8 signal - ss_o[7] is active high.	RW	1'b0
[6]	ssp7	Chip Select Polarity 7 Specifies the polarity of chip select signal for target 7. 1'b0 – Target select 7 signal - ss_o[6] is active low. 1'b1 – Target select 7 signal - ss_o[6] is active high.	RW	1'b0
[5]	ssp6	Chip Select Polarity 6 Specifies the polarity of chip select signal for target 6. 1'b0 – Target select 6 signal - ss_o[5] is active low. 1'b1 – Target select 6 signal - ss_o[5] is active high.	RW	1'b0
[4]	ssp5	Chip Select Polarity 5 Specifies the polarity of chip select signal for target 5. 1'b0 – Target select 5 signal - ss_o[4] is active low. 1'b1 – Target select 5 signal - ss_o[4] is active high.	RW	1'b0
[3]	ssp4	Chip Select Polarity 4 Specifies the polarity of chip select signal for target 4. 1'b0 – Target select 4 signal - ss_o[3] is active low. 1'b1 – Target select 4 signal - ss_o[3] is active high.	RW	1'b0
[2]	ssp3	Chip Select Polarity 3 Specifies the polarity of chip select signal for target 3. 1'b0 – Target select 3 signal - ss_o[2] is active low. 1'b1 – Target select 3 signal - ss_o[2] is active high.	RW	1'b0
[1]	ssp2	Chip Select Polarity 2 Specifies the polarity of chip select signal for target 2. 1'b0 – Target select 2 signal - ss_o[1] is active low. 1'b1 – Target select 2 signal - ss_o[1] is active high.	RW	1'b0
[0]	ssp1	Chip Select Polarity 1 Specifies the polarity of chip select signal for target 1. 1'b0 – Target select 1 signal - ss_o[0] is active low. 1'b1 – Target select 1 signal - ss_o[0] is active high.	RW	1'b0

5.15. FIFO Status Register (FIFO_STATUS_REG)

The FIFO Status Register reflects the status of Transmit FIFO and Receive FIFO.

Table 5.17. FIFO Status Register

Field	Name	Description	Access	Default
[7:6]	reserved	Reserved	RSVD	—
[5]	tx_fifo_full	Transmit FIFO Full This bit reflects the full condition of Transmit FIFO. 1'b0 – Transmit FIFO is not full. 1'b1 – Transmit FIFO is full.	RO	1'b0
[4]	tx_fifo_aempty	Transmit FIFO Almost Empty This bit reflects the almost empty condition of Transmit FIFO. 1'b0 – Data words in Transmit FIFO is greater than TX FIFO Almost Empty Flag attribute. 1'b1 – Data words in Transmit FIFO is less than or equal to the TX FIFO Almost Empty Flag attribute.	RO	1'b1
[3]	tx_fifo_empty	Transmit FIFO Empty This bit reflects the empty condition of Transmit FIFO. 1'b0 – Transmit FIFO is not empty – has at least one data word. 1'b1 – Transmit FIFO is empty.	RO	1'b1
[2]	rx_fifo_full	Receive FIFO Full This bit reflects the full condition of Receive FIFO. 1'b0 – Receive FIFO is not full. 1'b1 – Receive FIFO is full.	RO	1'b0
[1]	rx_fifo_afull	Receive FIFO Full This bit reflects the almost full condition of Receive FIFO. 1'b0 – Data words in Receive FIFO is less than the RX FIFO Almost Full Flag attribute. 1'b1 – Data words in Receive FIFO is greater than or equal to RX FIFO Almost Full Flag attribute.	RO	1'b0
[0]	rx_fifo_empty	Receive FIFO Full This bit reflects the empty condition of Receive FIFO. 1'b0 – Receive FIFO is not empty – has at least one data word. 1'b1 – Receive FIFO is empty.	RO	1'b1

5.16. SPI Enable Register (SPI_ENABLE_REG)

The SPI Enable Register is used for controlling the chip select line when CFG_REG.spi_en is set to 1.

Table 5.18. FIFO Reset Register

Field	Name	Description	Access	Default
[7:1]	reserved	Reserved	—	—
[0]	spi_en_reg	SPI Enable This bit enables or disables the SPI transfer through the chip select line control. For more information, refer to the SPI Enable section. 1'b0 – Disables SPI transfer. 1'b1 – Enables SPI transfer.	RW	1'b0

6. Example Design

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

- MachXO5-NX Development Board
- Avant-E Evaluation Board

6.1. Example Design Supported Configuration

Note: In the table below, ✓ refers to a checked option and × refers to an unchecked option or a non-applicable option in the SPI Controller IP example design.

Table 6.1. SPI Controller IP Configuration Used on Example Design

SPI Controller IP GUI Parameter	SPI Controller IP Configuration
General	
Interface	APB
Number of Chip Select Lines	2
First Transmitted Bit	MSB
Read-Write Operation	Read and Write
Data Width	32
Chip Select Pulse Mode	Pulse
SPI Clock Polarity	0
SPI Clock Phase	0
Chip Select Polarity (0x)	0
SPI Enable Register	×
Include I/O Buffer	✓
FIFO	
FIFO Width	32 (Display only)
FIFO Depth	16
Implementation of FIFO	EBR
TX FIFO Almost Empty Flag	3
RX FIFO Almost Full Flag	12
Clock	
System Clock Frequency (MHz)	50
Desired SCLK Frequency (MHz)	0.1
Clock Prescaler	250 (Calculated value)
SPI Actual Clock Frequency (MHz)	0.1 (Calculated value)

6.2. Overview of the Example Design and Features

The example design discussed in this section is created using the *RISC-V MC SoC Project* template in the Lattice Propel™ Design Environment. The generated project includes the following components:

- Processor – RISC-V (MC w/ PIC/Timer)
- GPIO
- Asynchronous SRAM
- UART – Serial port
- PLL
- Glue Logic

SPI Controller and SPI Target IPs are instantiated and connected in the project as shown in Figure 6.1. In this example, SPI Controller and SPI Target IPs are instantiated in the same system. In actual hardware or use case, you can connect the SPI Controller to external SPI Target devices.

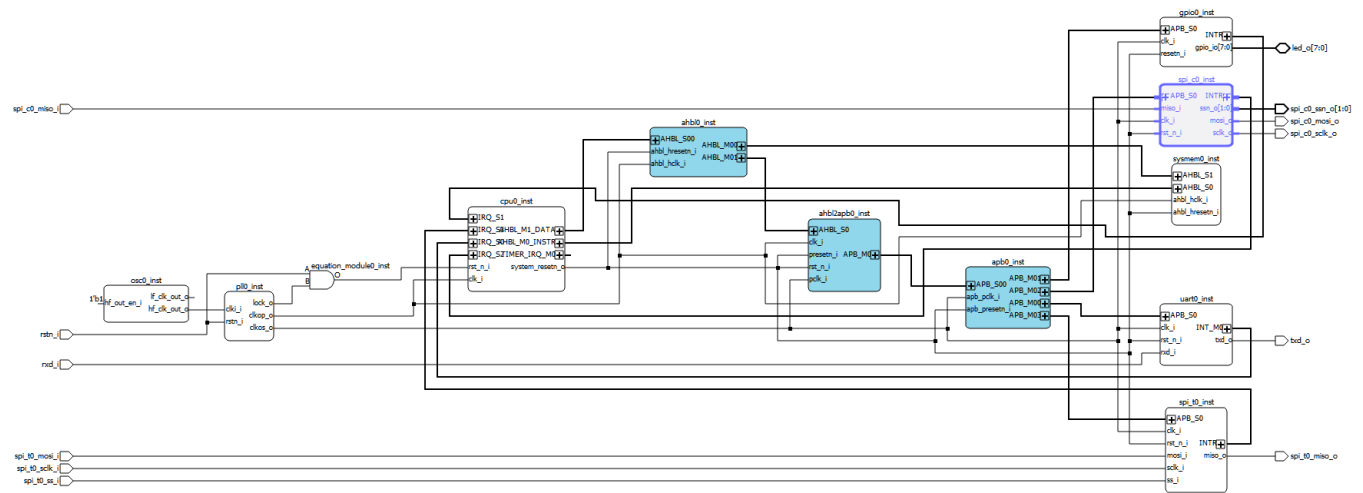


Figure 6.1. SPI Controller IP Example Design in Propel SoC Project

An embedded C/C++ project is also created in the Propel software to enable developing and debugging of application code for different IP features. SPI Controller IP features can be tested by sending SPI Commands from the SPI Controller IP to the SPI Target IP. Runtime configuration of IP and feature testing can be done through C-code test routine. Figure 6.2 shows an example routine to initiate data transfer from SPI Controller. This sample test routine can be found in the driver file included in the generated IP.

```

103 uint8_t spi_master_transfer(struct spim_instance *this_spim,
104                             uint32_t *data_buffer_out,
105                             uint32_t byte_out_size,
106                             uint32_t byte_in_size)
107 {
108     uint32_t byte_out_cnt = 0;
109     uint8_t int_status = 0;
110     uint8_t fifo;
111
112     if(this_spim->spim_current_status == SPIM_STATE_BUSY){
113         return STATE_ERROR;
114     }
115
116     if(this_spim == NULL)
117         return PTR_ERROR;
118
119     // clear the tx/rx fifo
120     reg_8b_write(this_spim->base_address | SPI_MASTER_FIFO_RST,
121                 SPI_MASTER_TX_FIFO_RST|SPI_MASTER_RX_FIFO_RST);
122
123     // clear all the interrupt bits
124     reg_8b_write(this_spim->base_address | SPI_MASTER_INT_STATUS,
125                 BITS_ALL_SET);
126
127     // clear the tx/rx fifo reset flag
128     reg_8b_write(this_spim->base_address | SPI_MASTER_FIFO_RST, 0);
129
130     reg_8b_write(this_spim->base_address | SPI_MASTER_BYTE_COUNT_RST, BITS_ALL_SET);
131     // set the word count register
132     reg_8b_write(this_spim->base_address | SPI_MASTER_TARGET_WORD_COUNT, byte_out_size+byte_in_size);
133
134     // write the data into the out buffer
135
136     this_spim->spim_current_status = SPIM_STATE_BUSY;
137
138     for(byte_out_cnt = 0; byte_out_cnt < byte_out_size; byte_out_cnt++)
139     {
140         reg_32b_write(this_spim->base_address |
141                     SPI_MASTER_WR_DATA, *data_buffer_out);
142         data_buffer_out++;
143     }
144
145     for(; byte_out_cnt < byte_in_size + byte_out_size ; byte_out_cnt++)
146     {
147         reg_32b_write(this_spim->base_address | SPI_MASTER_WR_DATA, 0xffffffff); // write dummy byte
148     }
149
150     return SUCCESS;
151 }

```

Figure 6.2. Sample C-Code Test Routine

6.3. Design Components Example

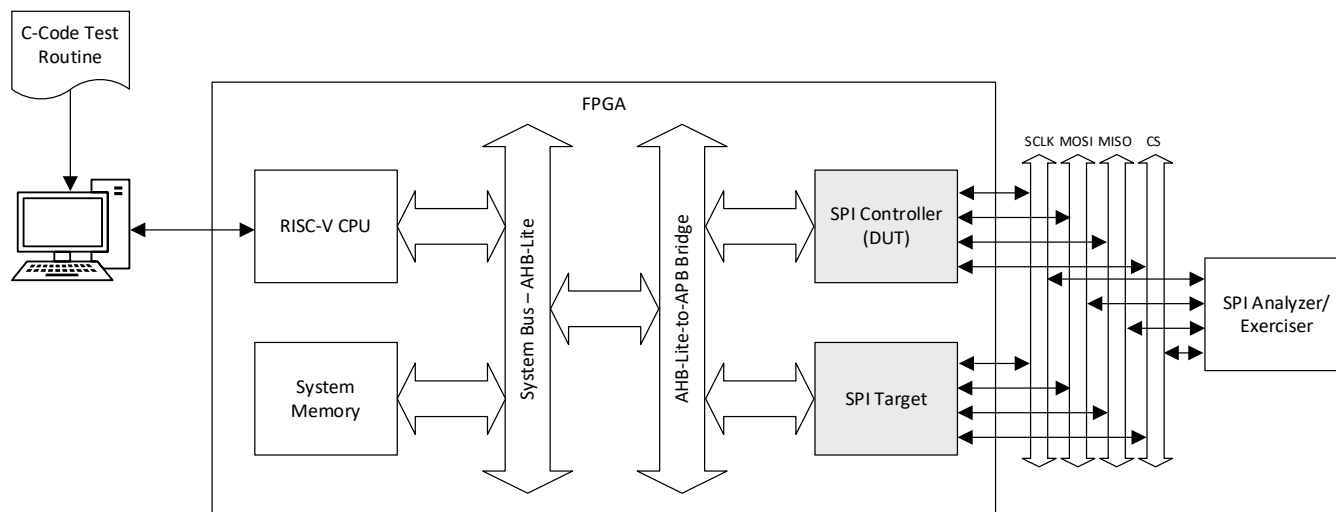


Figure 6.3. SPI Controller Example Design Block Diagram

The SPI Controller example design includes the following blocks:

- RISC-V CPU – Passes the C-code test routine from system memory to system bus. Handles interrupts.
- Memory – Contains commands for testing.
- System bus – AHB-Lite systems bus for transfers between memory and IP
- SPI Controller IP – IP instance connected to SPI bus (MISO, MOSI, CS, SCLK)
- SPI target device(s)

6.4. Generating the Example Design

Refer to the Lattice Propel SDK User Guide for more details on the Lattice Propel software.

1. Launch Lattice Propel software and set your workspace directory.
2. In the Propel software, create a new Lattice SoC Design Project by navigating to **File > New > Lattice SoC Design Project**. The **Create SoC Project** window opens.
3. In the **Device Select** section, indicate the correct details of the device or board that you are using. In [Figure 6.4](#), device is set to LFMXO5-25-9BBG400C because the MachXO5-NX Development Board is used in the hardware testing.
4. In the **Template Design** section, choose **RISC-V MC SoC Project**.
5. Click **Finish**.

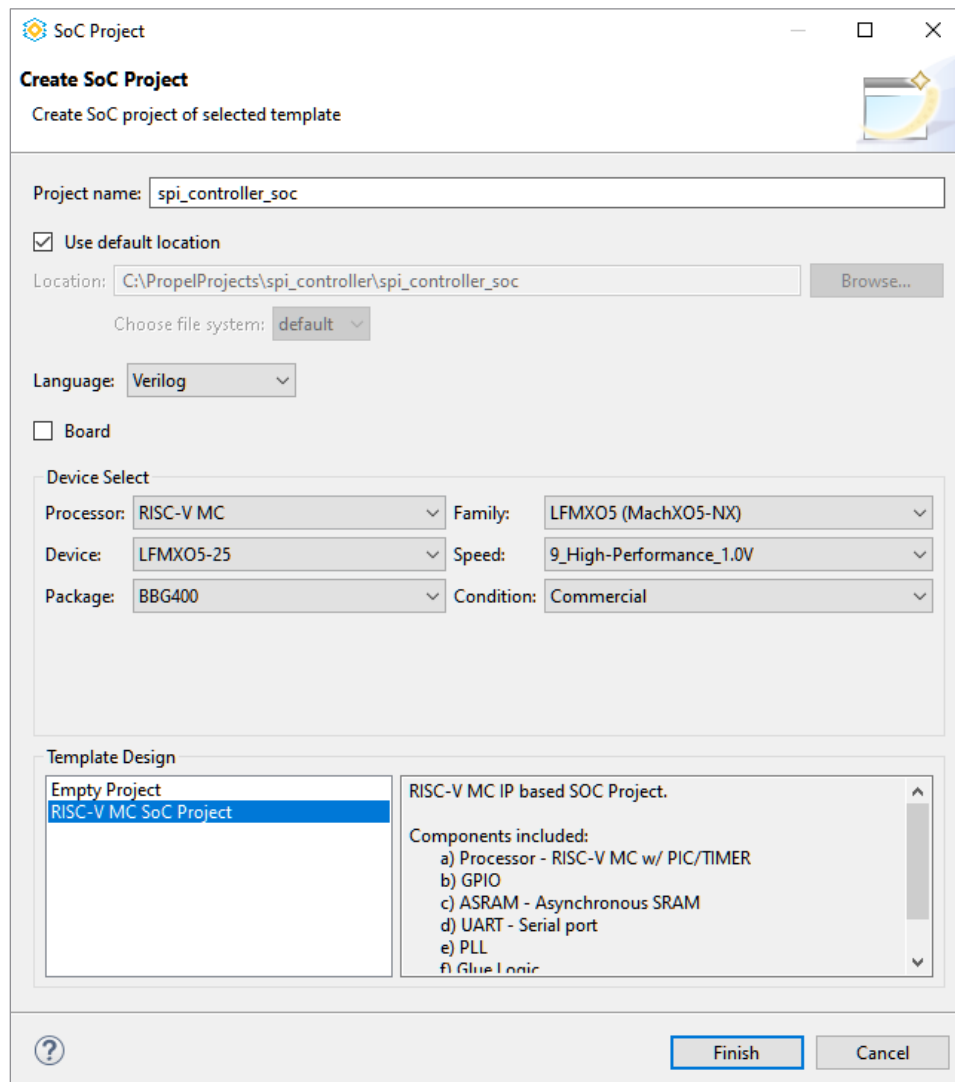


Figure 6.4. Create SoC Project

6. Run Propel Builder by clicking the  icon or navigate to **LatticeTools > Open Design** in Propel Builder. The Propel Builder opens and loads the design template.
7. In the **IP Catalog** tab, instantiate the SPI Controller IP. Refer to the [Generating and Instantiating the IP](#) section for more details. In this example, there is one instance of the SPI Controller IP. See the [Example Design Supported Configuration](#) section for the corresponding parameter settings.

8. After generating the IP, the **Define Instance** window opens. Modify the instance name if needed, then click **OK**.

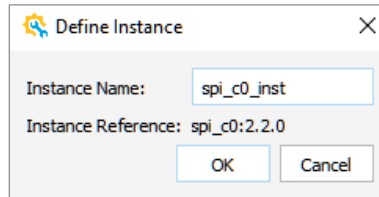


Figure 6.5. Define Instance

9. Connect the instantiated IPs to the system. Refer to [Figure 6.1](#) for the connections used in this IP. You will need to update other components of the system for clock and reset sources, interrupt, and bus interface.
10. Click the  icon or navigate to **Design > Run Radiant** to launch the Lattice Radiant Software.
11. Update your constraints file accordingly and generate the programming file.
12. In the Lattice Propel software, build your SoC project to generate the system environment needed for the embedded C/C++ project. Select your SoC project then navigate to **Project > Build Project**.
13. Check the build result from the **Console** view.

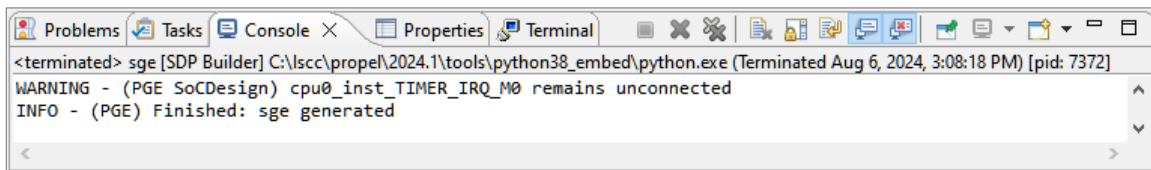


Figure 6.6. Build SoC Project Result

14. Generate a new Lattice C/C++ project by navigating to **File > New > Lattice C/C++ Project**. Update your **Project name**, click **Next**, and then click **Finish**.

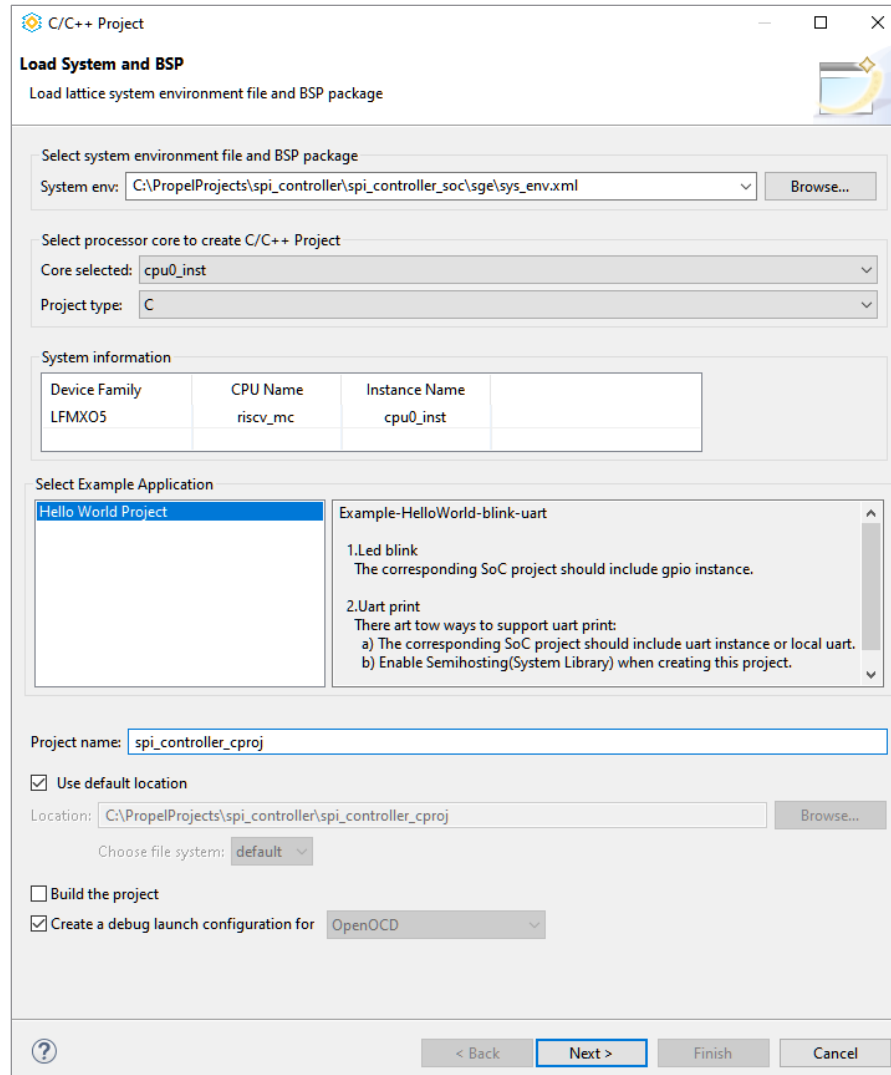


Figure 6.7. Lattice C/C++ Design Project

15. Select your C/C++ project, then select **Project > Build**.
16. Check the build result from the **Console** view.

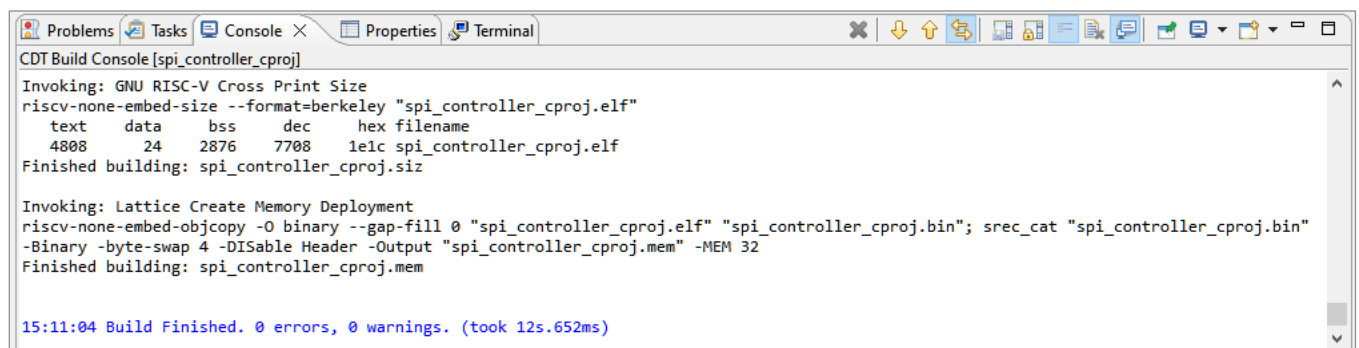


Figure 6.8. Build C/C++ Project Result

This environment is now ready for running your tests on the device. Refer to the *Running Demo on MachXO3D Breakout Board – Hello World* section of the Lattice Propel SDK User Guide for step-by-step guide.

6.5. Hardware Testing

6.5.1. Hardware Testing Setup

Download the generated bitstream file from the [Generating the Example Design](#) section to the MachXO5-NX Development Board using the Lattice Radiant Programmer.

6.5.2. Expected Output

The following is a sample waveform captured using the SPI Analyzer/Exerciser tool.

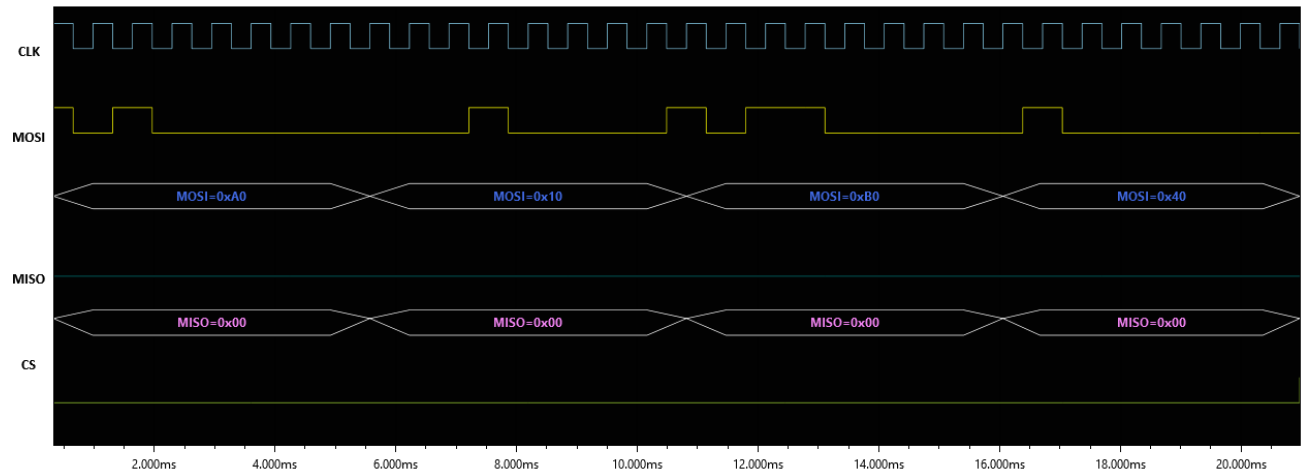


Figure 6.9 Sample SPI Transaction between Controller and Target

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](#).

7.1. Generating and Instantiating the IP

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

To generate the SPI Controller IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **SPI Controller** under **IP, Processors, Controllers, and Peripherals** category. The **Module/IP Block Wizard** opens, as shown in [Figure 7.1](#). Enter values in the **Component name** and the **Create in** fields and click **Next**.

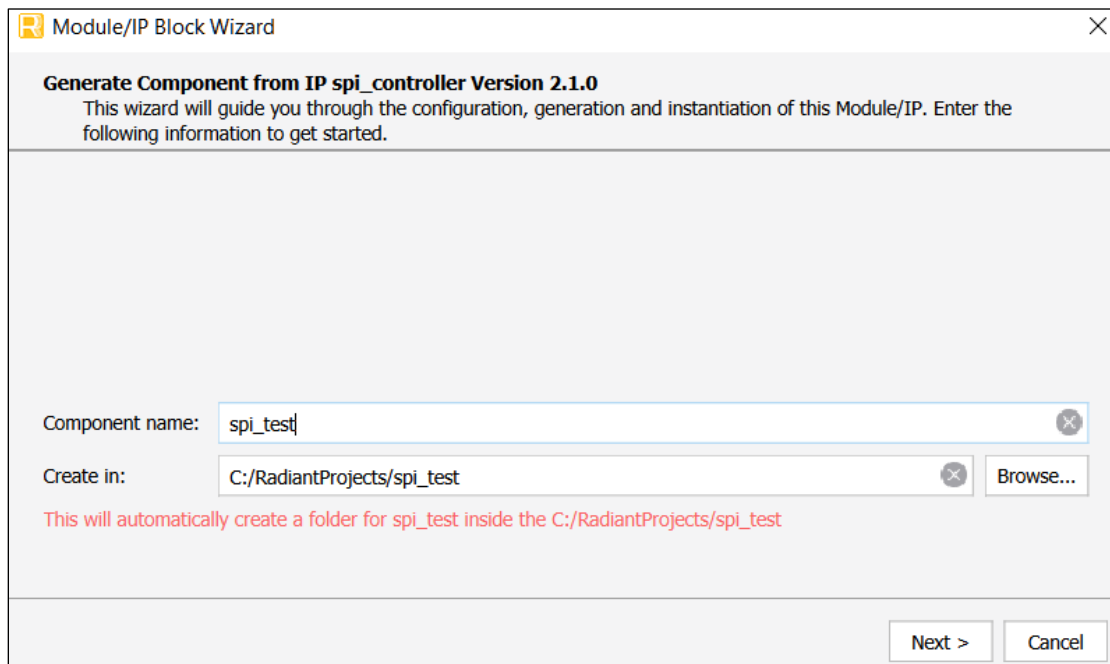


Figure 7.1. Module/IP Block Wizard

3. In the **Module/IP Block Wizard** window, customize the selected SPI Controller IP using drop-down menus and check boxes. [Figure 7.2](#) shows an example configuration of the SPI Controller IP. For details on the configuration options, refer to the [IP Parameter Description](#) section.

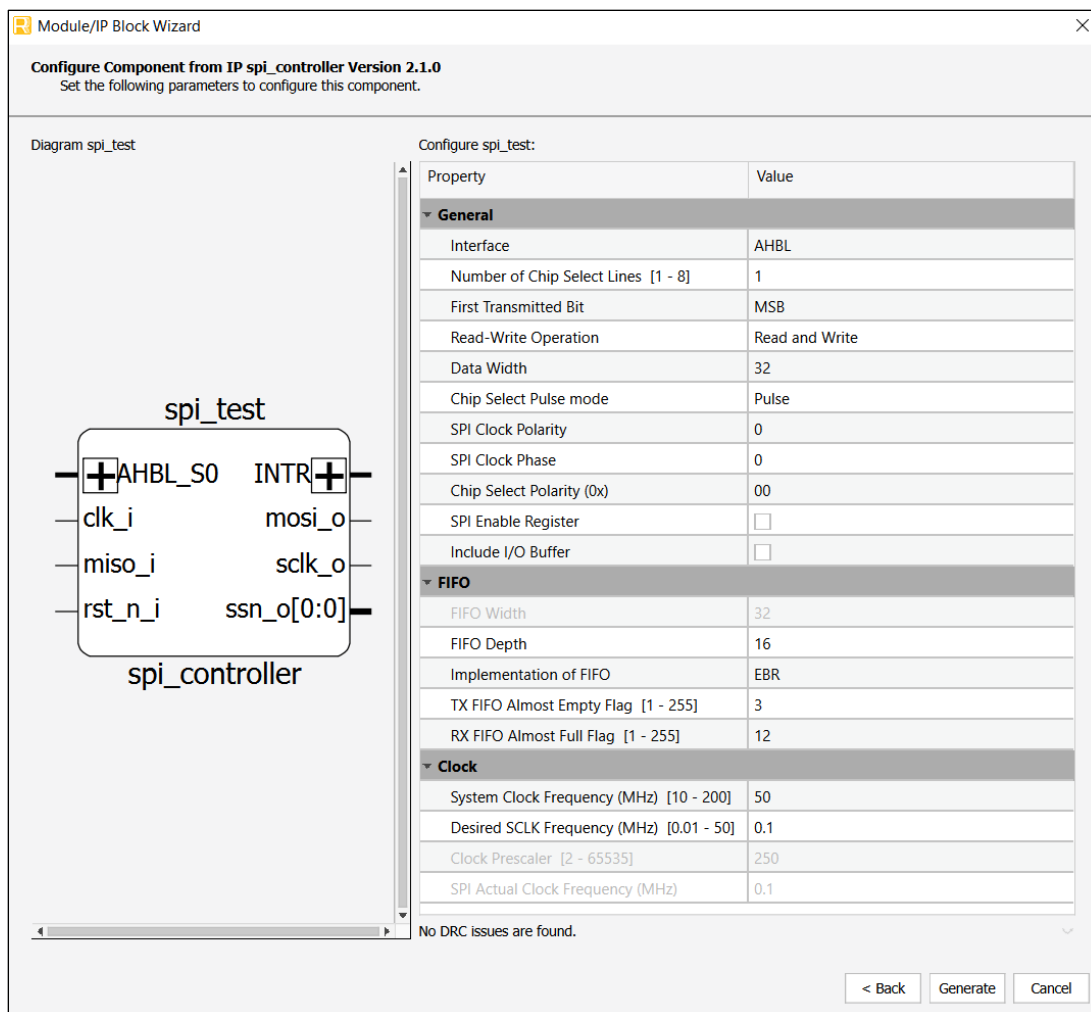


Figure 7.2. IP Configuration

- Click **Generate**. The **Check Generated Result** dialog box opens, showing design block messages and results (Figure 7.3).

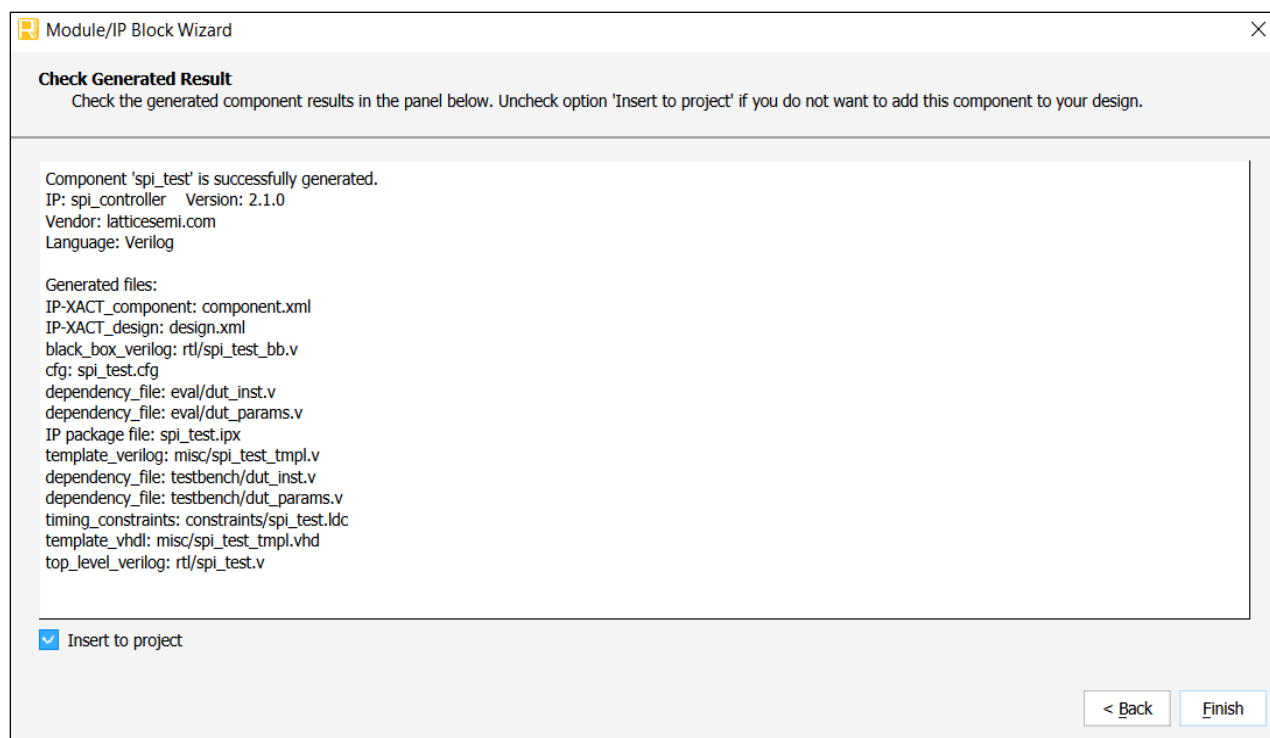


Figure 7.3. Check Generated Result

- Click **Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields shown in [Figure 7.1](#).

7.1.1. Generated Files and File Structure

The generated SPI Controller module package includes the black 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 top-level reference as the starting template for the complete top-level design. The generated files are listed in [Table 7.1](#).

Table 7.1. Generated File List

Attribute	Description
<Component name>.ipx	This file contains the information on the files associated to the generated IP.
<Component name>.cfg	This file 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	This file provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	This file provides the synthesis black box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	These files provide instance templates for the module.
eval/constraint.pdc	This file provides information on how to constrain this IP. Refer to the Constraining the IP section on how to use this file.

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.

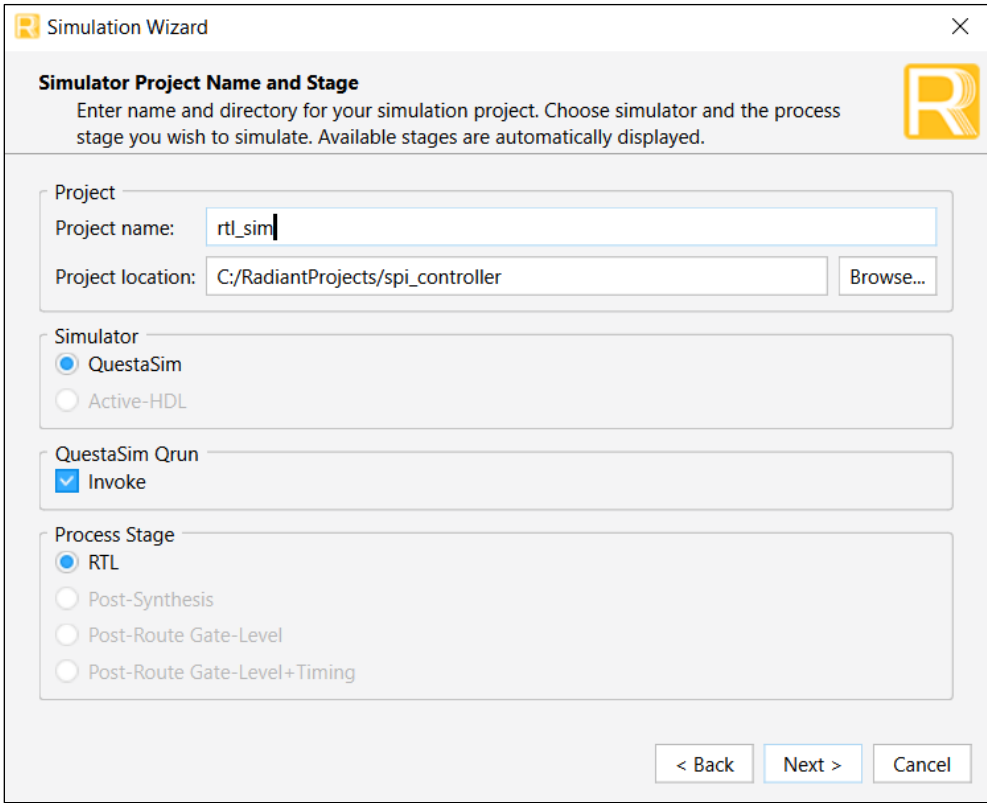
Refer to the relevant sections in the [Lattice Radiant](#) Software User Guide for more information on how to create or edit constraints and how to use the Device Constraint Editor.

7.3. Running Functional Simulation

You can run functional simulation after the IP is generated.

To run functional simulation:

1. Click the  icon located on the **Toolbar** to initiate the **Simulation Wizard** shown in [Figure 7.4](#).



The **Simulation Wizard** dialog box is shown. It has a title bar with a close button. The main content area is titled **Simulator Project Name and Stage** and includes the instruction: "Enter name and directory for your simulation project. Choose simulator and the process stage you wish to simulate. Available stages are automatically displayed." The dialog is divided into several sections:

- Project**: Contains a "Project name:" text field with "rtl_sim" entered, and a "Project location:" text field with "C:/RadiantProjects/spi_controller" entered. A "Browse..." button is next to the location field.
- Simulator**: Contains two radio buttons: "QuestaSim" (selected) and "Active-HDL".
- QuestaSim Qrun**: Contains a checked checkbox labeled "Invoke".
- Process Stage**: Contains four radio buttons: "RTL" (selected), "Post-Synthesis", "Post-Route Gate-Level", and "Post-Route Gate-Level+Timing".

At the bottom right, there are three buttons: "< Back", "Next >", and "Cancel".

Figure 7.4. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window as shown in [Figure 7.5](#).

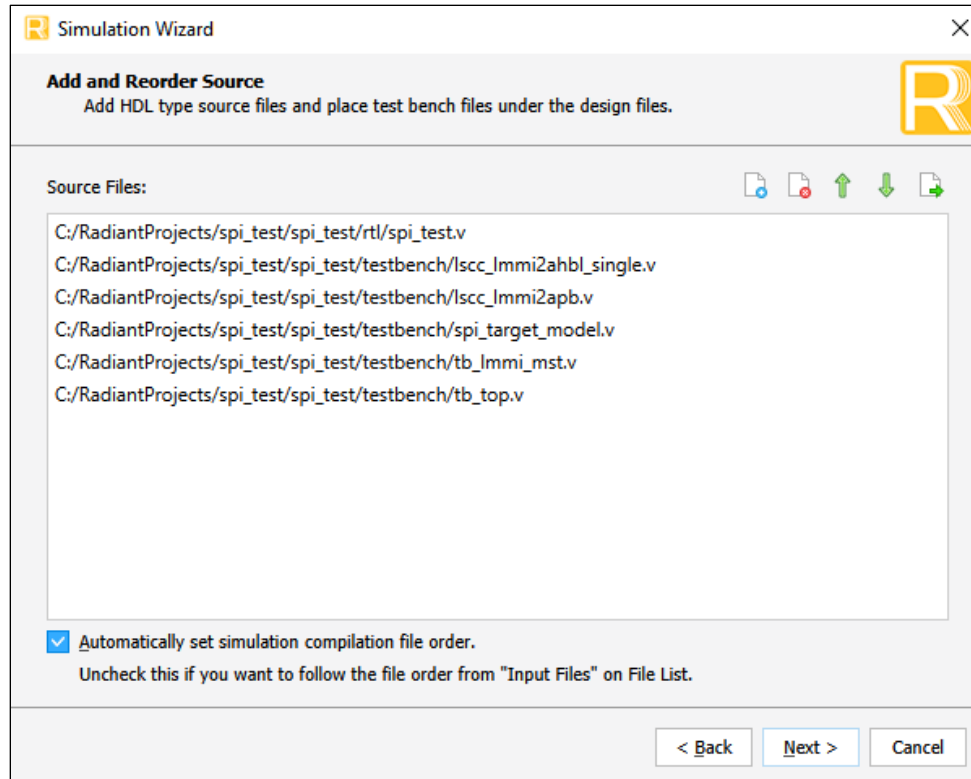


Figure 7.5. Add and Reorder Source

3. Click **Next**. The **Summary** window is shown.
4. Click **Finish** to run the simulation.

The waveform in Figure 7.6 shows an example of simulation result.

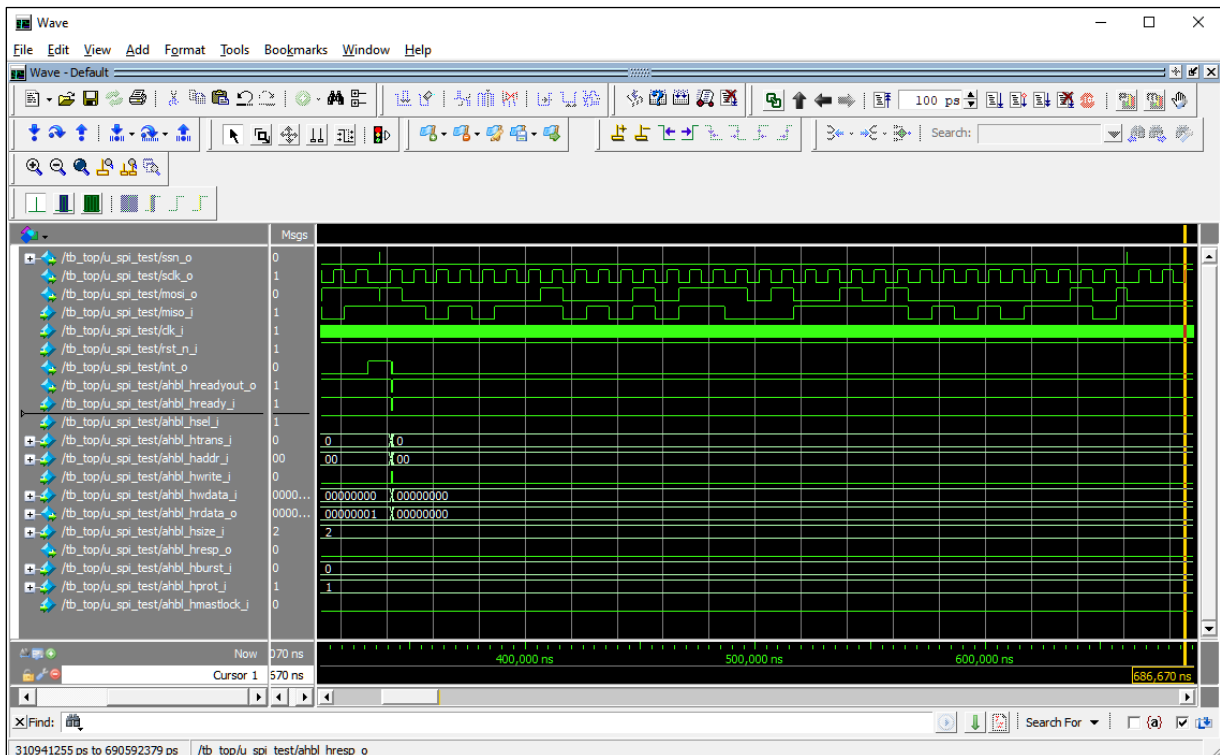


Figure 7.6. Simulation Waveform

7.3.1. Simulation Results

Figure 7.7 shows AHB-Lite write operation, writing the data (32'h8484d609) to address (32'h00000004).

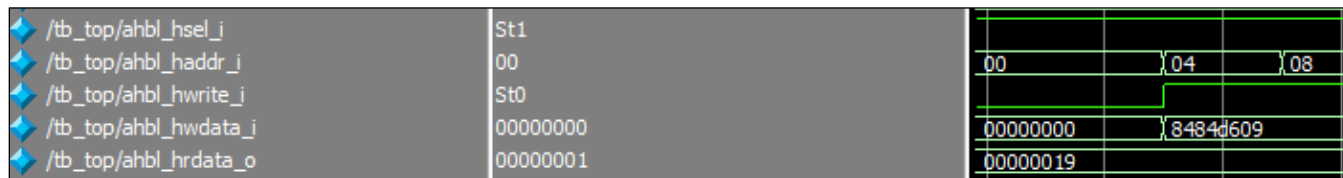


Figure 7.7. AHB-Lite Write Operation

Figure 7.8 shows AHB-Lite read operation, reading the data (32'h00000019) from address (32'h00000038).



Figure 7.8. AHB-Lite Read Operation

Figure 7.9 shows SPI Controller Transaction.

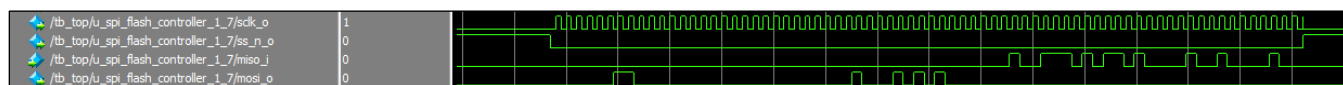


Figure 7.9. SPI Controller Transaction

Figure 7.10 shows the simulation has completed successfully when the ssn_o raise the level and maintain it for a period of time.

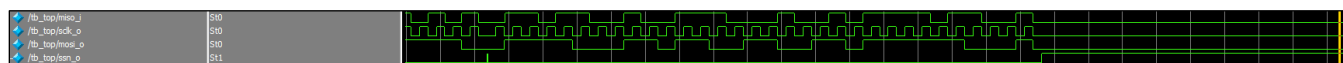


Figure 7.10. SPI Controller Simulation Completed Successfully

7.4. Constraining the IP

You need to provide proper timing and physical design constraints to ensure that your design meets the desired performance goals on the FPGA. Add the content of the following IP constraint file to your design constraints:

<IP_Instance_Path>/<IP_Instance_Name>/eval/constraint.pdc.

The constraint file has been verified during IP evaluation with the IP instantiated directly in the top-level module. You can modify the constraints in this file with thorough understanding of the effect of each constraint.

To use this constraint file:

Copy the content of constraint.pdc to the top-level design constraint for post-synthesis.

Refer to [Lattice Radiant Timing Constraints Methodology \(FPGA-AN-02059\)](#) for details on how to constraint your design.

Appendix A. Resource Utilization

Table A.1 shows the resource utilization of the SPI Controller IP Core for the LAV-AT-E70-3LFG1156I device using Synplify Pro of Lattice Radiant software 2023.2. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.1. Resource Utilization for LAV-AT-70E-3LFG1156I

Configuration	Clk f_{MAX} (MHz) ¹	Registers	LUTs	EBRs
Default	250.00	451	631	2
Interface: APB, Others = Default	250.00	368	446	2
Interface: LMMI, Others = Default	250.00	303	446	2
FIFO Depth: 256, TX FIFO Almost Empty Flag: 255, RX FIFO Almost Full Flag: 255, Others = Default	250.00	600	734	2
Implementation of FIFO: LUT, TX FIFO Almost Empty Flag: 1, RX FIFO Almost Full Flag: 1, Others = Default	215.47	627	979	0

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Controller IP Core, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

Table A.2 shows the resource utilization of the SPI Controller IP Core for the LFMX05-25-9BBG400I device using Synplify Pro of Lattice Radiant software 2022.1. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.2. Resource Utilization for LFMX05-25-9BBG400I

Configuration	Clk f_{MAX} (MHz) ¹	Registers	LUTs	EBRs
Default	176.84	451	618	2
Interface: APB, Others = Default	157.51	368	439	2
Interface: LMMI, Others = Default	200.00	303	416	2
FIFO Depth: 256, TX FIFO Almost Empty Flag: 255, RX FIFO Almost Full Flag: 255, Others = Default	151.04	600	754	2
Implementation of FIFO: LUT, TX FIFO Almost Empty Flag: 1, RX FIFO Almost Full Flag: 1, Others = Default	196.04	515	712	0

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Controller IP Core, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

Table A.3 shows the resource utilization of the SPI Controller IP Core for the LFCPNX-100-9BBG484I device using Synplify Pro of Lattice Radiant software 2022.1. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.3. Resource Utilization for LFCPNX-100-9BBG484I

Configuration	Clk f_{MAX} (MHz) ¹	Registers	LUTs	EBRs
Default	165.10	451	618	2
Interface: APB, Others = Default	186.88	368	439	2
Interface: LMMI, Others = Default	200.00	303	416	2
FIFO Depth: 256, TX FIFO Almost Empty Flag: 255, RX FIFO Almost Full Flag: 255, Others = Default	149.84	600	754	2
Implementation of FIFO: LUT, TX FIFO Almost Empty Flag: 1, RX FIFO Almost Full Flag: 1, Others = Default	184.50	515	712	0

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Controller IP Core, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

Table A.4 shows the resource utilization of the SPI Controller IP Core for the LFD2NX-40-9BG256I device using Synplify Pro of Lattice Radiant software 2022.1. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.4. Resource Utilization for LFD2NX-40-9BG256I

Configuration	Clk f_{MAX} (MHz) ¹	Registers	LUTs	EBRs
Default	166.25	451	618	2
Interface: APB, Others = Default	137.97	368	439	2
Interface: LMMI, Others = Default	200.00	303	416	2
FIFO Depth: 256, TX FIFO Almost Empty Flag: 255, RX FIFO Almost Full Flag: 255, Others = Default	155.74	600	754	2
Implementation of FIFO: LUT, TX FIFO Almost Empty Flag: 1, RX FIFO Almost Full Flag: 1, Others = Default	184.98	515	712	0

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Controller IP Core, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

Table A.5 shows the resource utilization of the SPI Controller IP Core for the LIFCL-40-9BG400I device using Synplify Pro of Lattice Radiant software 2022.1. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.5. Resource Utilization for LIFCL-40-9BG400I

Configuration	Clk f_{MAX} (MHz) ¹	Registers	LUTs	EBRs
Default	165.32	451	618	2
Interface: APB, Others = Default	146.59	368	439	2
Interface: LMMI, Others = Default	200.00	303	416	2
FIFO Depth: 256, TX FIFO Almost Empty Flag: 255, RX FIFO Almost Full Flag: 255, Others = Default	173.55	600	754	2
Implementation of FIFO: LUT, TX FIFO Almost Empty Flag: 1, RX FIFO Almost Full Flag: 1, Others = Default	169.35	515	712	0

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Controller IP Core, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

Table A.6 shows the resource utilization of the SPI Controller IP Core for the LN2-CT-20-1CBG484C device using Synplify Pro of Lattice Radiant software 2024.2. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.6. Resource Utilization for LN2-CT-20-1CBG484C

Configuration	Clk f_{MAX} (MHz) ¹	Registers	LUTs	EBRs
Default	244.978	473	704	2
Interface: APB, Others = Default	250	368	468	2
Interface: LMMI, Others = Default	250	323	440	2
FIFO Depth: 256, TX FIFO Almost Empty Flag: 1, RX FIFO Almost Full Flag: 255, Others = Default	218.150	565	806	2
Implementation of FIFO: LUT, TX FIFO Almost Empty Flag: 1, RX FIFO Almost Full Flag: 1, Others = Default	250	451	628	0

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Controller IP Core, and the target frequency is 50 MHz. These values may be reduced when user logic is added to the FPGA design.

References

- [SPI Controller IP Release Notes \(FPGA-RN-02015\)](#)
- [Lattice Radiant Timing Constraints Methodology \(FPGA-AN-02059\)](#)
- [Avant-E](#) web page
- [Avant-G](#) web page
- [Avant-X](#) web page
- [Certus-N2](#) web page
- [Certus-NX](#) web page
- [CertusPro-NX](#) web page
- [CrossLink-NX](#) web page
- [MachXO5-NX](#) web page
- [SPI Controller IP Core](#) web page
- [Avant-E Evaluation Board](#) web page
- [MachXO5-NX Development Board](#) web page
- [Lattice Radiant Software](#) web page
- [Lattice Propel Design Environment](#) web page
- [Lattice Solutions IP Cores](#) web page
- [Lattice Solutions Reference Designs](#) web page
- [Lattice Insights](#) for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

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

Revision History

Revision 2.3, IP v2.4.0, July 2025

Section	Change Summary
Introduction	In Table 1.1. Summary of the SPI Controller IP: - Renamed <i>Supported FPGA Family</i> to <i>Supported Devices</i> and rearranged items. - Removed <i>Targeted Devices</i> row. - Added IP core version to <i>Lattice Implementation</i>.
References	- Added SPI Controller IP Core web page. - Updated listing name to Lattice Radiant Software web page.

Revision 2.2, IP v2.3.0, December 2024

Section	Change Summary
All	Minor editorial changes.
Cover	Added IP version.
Abbreviations in This Document	- Updated section title, description, and table header. - Added items <i>AHB-Lite</i>, <i>DUT</i>, <i>EBR</i>, <i>GPIO</i>, <i>LUT</i>, <i>MC</i>, <i>PIC</i>, <i>SRAM</i>, and <i>UART</i>.
Introduction	- In Table 1.1. Summary of the SPI Controller IP: - Added Certus-NX-RT, CertusPro-NT-RX, and Certus-N2 to <i>Supported FPGA Family</i>. - Added IP Changes row. - Removed IP Version row. - Added LFD2NX-9, LFD2NX-28, and LN2-CT-20 to <i>Targeted Devices</i>. - Added Resources row. - Added IP Core v2.3.0 to <i>Lattice Implementation</i>. - Added the IP Support Summary section. - Removed the IP Validation Summary section. - Added the Hardware Support section. - Removed the Host section under the Conventions section.
Functional Description	Added note regarding <code>ss_o</code> when Transmit FIFO becomes empty within a transfer in the Chip Select Pulse Mode section.
IP Parameter Description	- Added statement regarding default values in bold. - In Table 3.1. General Attributes: - Updated title. - Added Description column and merged descriptions from Table 3.2. Attributes Descriptions into table. - Merged information under Dependency on Other Attributes into description and removed column. - Merged default settings under Default into Selectable Values and removed Default column. - Updated selectable values range for <i>TX FIFO Almost Empty Flag</i> and <i>RX FIFO Almost Full Flag</i>. - Removed Table 3.2. Attributes Descriptions.
Signal Description	- In Table 4.1. SPI Controller IP Signal Description: - Renamed column from I/O to Type. - Rearranged Width column. - Added Output Reset Value column.
Register Description	- Renamed Table 5.2. Register Access Types and added Access Type Abbreviation column. - Revamped content organization: - Merged register bit/field information into existing tables. - Restructured tables to include Description column and removed Width column.

Section	Change Summary
	Updated descriptions.
Example Design	Added new section.
Designing with the IP	- Updated Figure 7.4. Simulation Wizard. - Removed the IP Evaluation and Hardware Validation sections
Appendix A. Resource Utilization	Added Table A.6. Resource Utilization for LN2-CT-20-1CBG484C.
Appendix B. Limitations	Removed section.
References	- Removed statement about Lattice Radiant Software User Guide. - Added SPI Controller IP Release Notes and Certus-N2 web page. - Added various other links for Radiant software, IP cores, reference designs, Propel design environment, MachXO5-NX Development Board, Avant E Evaluation Board, and Lattice Radiant Timing Constraints Methodology.

Revision 2.1, December 2023

Section	Change Summary
All	Revamped the document structure.
Introduction	- Organized the general description of the IP into the Overview of the IP subsection. - Updated licensing information of the IP and moved it to the Licensing and Ordering Information subsection under Introduction. - Newly added the IP Validation Summary subsection.
Functional Description	- IP Architecture Overview: - changed the subsection name from <i>Overview</i> to <i>IP Architecture Overview</i>; - newly added <i>the SPI Controller IP includes the following layers: APB/AHB-Lite Interface, LMMI Device, SPI Top.</i> - Newly added the Clocking and Reset subsections. - Newly added the User Interfaces subsection and merged the original subsection 2.8 Selectable Memory-Mapped Interface into User Interfaces.
IP Parameter Description	- Moved the original Table 2.2. Attributes Table and Table 2.3. Attributes Descriptions to this section. - Newly added the Include I/O Buffer attribute to Table 3.1. Attributes Table and Table 3.2. Attributes Description.
Signal Description	- Moved the original subsection 2.2 Signal Description from Functional Description to this section. - Changed <i>Input</i> to <i>In</i> for <i>miso_i</i> in Table 4.1. SPI Controller IP Core Signal Description.
Register Description	Moved the original subsection 2.4 Register Description from Functional Description to this section.
Designing with the IP	- Changed the section name from <i>IP Generation and Evaluation</i> to <i>Designing with the IP</i>. - Changed the subsection name from <i>Generation and Synthesis</i> to <i>Generating and Instantiating the IP</i>. - Updated Figure 6.1. Module/IP Block Wizard, Figure 6.2. IP Configuration, and Figure 6.3. Check Generated Result. - Arranged information of generated files of the IP core into a new subsection, Generated Files and File Structure. - Newly added the Design Implementation subsection. - Newly added the Simulation Results subsection.

Revision 2.0, December 2023

Section	Change Summary
All	- Updated title from <i>SPI Controller IP Core - Lattice Radiant Software</i> to <i>SPI Controller IP Core</i>. - Updated instances of <i>LAV-AT-500E</i> to <i>LAV-AT-E70</i>.

Section	Change Summary
Disclaimers	Updated this section.
Introduction	Added new targeted devices <i>LAV-AT-G70</i> , and <i>LAV-AT-X70</i> to Table 1.1. Quick Facts.
Functional Description	Updated <i>FIFO Depth</i> in Table 2.2. Attributes Table.
IP Generation and Evaluation	- Added <i>eval/constraint.pdc</i> to Table 3.1. Generated File List. - Added section 3.4. Constraining the IP. - Updated section 3.6. Hardware Validation.
Resource Utilization	Updated the <i>Clk Fmax</i> , <i>Slice Registers</i> , and <i>LUTs</i> values of all configurations in Table A.1. Resource Utilization for <i>LAV-AT-E70-3LFG1156I</i> .
Read Data Path Optimization by Synthesis Tool	Removed this section.
References	Updated this section.

Revision 1.9, June 2023

Section	Change Summary
All	Updated terminologies to replace with the following: <i>Master</i> is replaced with <i>Controller</i> <i>Slave</i> is replaced with <i>Target</i> <i>Slave Select</i> is replaced with <i>Chip Select</i>
Inclusive Language	Added Inclusive Language information.
Introduction	Updated Table 1.1. Quick Facts: - Replaced <i>LFD2NX-40</i>, <i>LFD2NX-17</i>, <i>LFCPNX-100</i>, <i>LFMXO5-25</i> with <i>LFD2NX-40</i>, <i>LFD2NX-17</i>, <i>LFCPNX-50</i>, <i>LFCPNX-100</i>, <i>LFMXO5-25</i>, <i>LFMXO5-55T</i>, <i>LFMXO5-100T</i>, <i>UT24C40</i>, <i>UT24CP100</i> in Target Devices list. - Updated IP Version to v2.xx and Lattice Radiant software version to 2022.1
Functional Description	Updated below figures: - Updated to SPI Controller IP in Figure 2.1. SPI Controller IP Core Block Diagram. - Updated APB command sequence in Figure 2.10. Long Gap between Writing Data to Transmit FIFO. - Updated terminology from <i>Slave</i> to <i>Target</i> in Figure 2.11. Independent Target Configuration.
IP Generation and Evaluation	- Updated the below figures for updating as per spi_controller version 2.0.0: - Figure 3.1. Module/IP Block Wizard - Figure 3.2. Configure User Interface of SPI Controller IP Core - Figure 3.3. Check Generating Result - Updated the project name and location in Figure 3.4. Simulation Wizard. - Updated the source files list in Figure 3.5. Adding and Reordering Source. - Updated the wave form in Figure 3.6. Simulation Waveform.
Technical Support Assistance	Added Lattice Answer database link.
Reference	Added links for Crosslink-NX, Certus-NX, CertusPro-NX, MachXO5-NX, and Lattice Avant-E.

Revision 1.8, November 2022

Section	Change Summary
Introduction	Updated Table 1.1. Quick Facts: - Added Lattice Avant to Supported FPGA Family. - Added <i>LAV-AT-500E</i> to Targeted Device.
Functional Description	- In Table 2.2. Attributes Table: - Changed the selectable values of 'TX FIFO Almost Empty Flag' and 'RX FIFO Almost Full Flag' from '0–255' to '1–255' in Table 2.2. - Changed the selectable values of 'Clock Prescaler' from '2–65536' to '2–65535'. - The title of Figure 2.2 changed from 'Clocking Mode 0 (CPOL=0, CPHA=1)' to 'Clocking

Section	Change Summary
	Mode 0 (CPOL=0, CPHA=0)'. - Added Table A.1. Resource Utilization for LAV-AT-500E-3LFG1156I. - Updated the values for Clk Fmax (MHz) and Registers in Table A.2. Resource Utilization for LFMXO5-25-9BBG400I, Table A.3. Resource Utilization for LFCPNX-100-9BBG484I, Table A.4. Resource Utilization for LFD2NX-40-9BG256I, and Table A.5. Resource Utilization for LIFCL-40-9BG400I.

Revision 1.7, July 2022

Section	Change Summary
Introduction	Added LIFCL-33 to the Targeted Device in Table 1.1. Quick Facts.
Functional Description	- Updated apb_pwdata_i description in Table 2.1. SPI Controller IP Core Signal Description. - Added SPI Enable Register attribute to Table 2.2. Attributes Table and Table 2.3. Attributes Descriptions. - Added SPI_ENABLE_REG to Table 2.4. Summary of SPI Controller IP Core Registers. - Made general updates in the Write Data Register (WR_DATA_REG) section. - Made general updates in the Read Data Register (RD_DATA_REG) section. - Added spi_en as bit [7] in the Configuration Register (CFG_REG) section. - Newly added the SPI Enable Register (SPI_ENABLE_REG) section. - Newly added the SPI Enable section. - Updated Programming Flow details and separated the contents to Transmit/Receive Operation Interrupt Mode and Transmit/Receive Operation Polling Mode.
Designing with the IP	Updated: - Figure 3.1. Module/IP Block Wizard, - Figure 3.2. Configure User Interface of SPI Controller IP Core, - Figure 3.3. Check Generating Result, - Figure 3.4. Simulation Wizard, - Figure 3.5. Adding and Reordering Source, and - Figure 3.6. Simulation Waveform to show the latest software version 1.3.1.
Resource Utilization	- Updated Table A.2. Resource Utilization for LFMXO5-25-9BBG400I for resource utilization of the SPI Master IP Core for the LFMXO5-25-9BBG400I. - Updated Table A.3. Resource Utilization for LFCPNX-100-9BBG484I for resource utilization of the SPI Master IP Core for the LFCPNX-100-9BBG484I. - Added Table A.4. Resource Utilization for LFD2NX-40-9BG256I for resource utilization of the SPI Master IP Core for the LFD2NX-40-9BG256I. - Added Table A.5. Resource Utilization for LIFCL-40-9BG400I for resource utilization of the SPI Master IP Core for the LIFCL-40-9BG400I.

Revision 1.6, May 2022

Section	Change Summary
Introduction	Added MachXO5-NX to the Supported FPGA Family, and LFMXO5-25 to the Supported User Interfaces in Table 1.1.
IP Core Generation, Simulation, and Validation	Updated Figure 3.1, Figure 3.2, and Figure 3.3 to show the latest version 1.3.0.
Appendix A. Resource Utilization	- Added Table A.2. Resource Utilization for LFMXO5-25-9BBG400I for resource utilization of the SPI Master IP Core for the LFMXO5-25-9BBG400I. - Added Table A.3. Resource Utilization for LFCPNX-100-9BBG484I for resource utilization of the SPI Master IP Core for the LFMXO5-25-7BBG400I.

Revision 1.5, October 2021

Section	Change Summary
All	Minor adjustments in formatting across the document.
Functional Description	Updated Table 2.2 to change Selectable Values and Dependency on Other Attributes of TX FIFO Almost Empty Flag and RX FIFO Almost Full Flag attributes.

Revision 1.4, June 2021

Section	Change Summary
Introduction	- Remove second paragraph. - Updated Table 1.1. - Revised Supported FPGA Families - Revised Targeted Devices - Revised Lattice Implementation - Revised reference to Lattice Radiant Software User Guide
IP Generation and Evaluation	Updated supported devices in the Hardware Evaluation section.
Appendix B. Limitations	Changed <i>Devices Affected</i> to All.
References	Revised reference to Lattice Radiant Software User Guide and removed link.

Revision 1.3, August 2020

Section	Change Summary
Functional Description	Updated Table 2.3.
Appendix C. Read Data Path Optimization by Synthesis Tool	Added this section.

Revision 1.2, June 2020

Section	Change Summary
Introduction	Updated Table 1.1.
Appendix B. Limitations	Added this section.

Revision 1.1, February 2020

Section	Change Summary
Introduction	Updated Table 1.1 to add LIFCL-17 as targeted device.
IP Generation and Evaluation	Updated user interface item to IP, Processors_Controllers_and_Peripherals category.

Revision 1.0, December 2019

Section	Change Summary
All	Changed document status from Preliminary to final.
Introduction	Updated Lattice Implementation information in Table 1.1. Quick Facts.
Appendix A. Resource Utilization	Updated resource utilization information.

Revision 0.80, October 2019

Section	Change Summary
All	Preliminary release.



www.latticesemi.com