SPI Target IP

IP Version: v2.4.0

User Guide

FPGA-IPUG-02070-2.2

July 2025

Disclaimers

Inclusive Language

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

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AHB	Advanced High-performance Bus
AHB-Lite	Advanced High-performance Bus Lite
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
CPHA	Clock Phase
CPOL	Clock Polarity
DUT	Device Under Test
EBR	Embedded Block RAM
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GPIO	General Purpose I/O
LMMI	Lattice Memory Mapped Interface
LUT	Look-Up Table
MC	Microcontroller
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, full-duplex interface that allows a serial bit stream of configured length (8, 16, 24, and 32 bits) to be shifted into and out of the device at a programmed bit-transfer rate.

1.2. Quick Facts

Table 1.1. Summary of the SPI Target 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 Target IP Release Notes (FPGA-RN-02014).
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 Target IP Support Readiness

Device Family	IP	System Clock Frequency (MHz)	Chip Select	SPI Clock	Features Supported	Radiant Timing Model	Hardware Validated
Avant	SPI Target	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Preliminary	Yes
MachXO5-NX	SPI Target	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
Certus-NX	SPI Target	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 Target	50	Pulse Mode, Low Polarity	Low Polarity, No Phase Shift	SCLK at 0.1 MHz: Data Width = 32 Read and Write	Final	No
CrossLink-NX	SPI Target	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 Target	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 Target	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 Target	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 Target 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
- Support for all SPI clocking modes (combination of clock polarity and clock phase)
- Sending of defined static value when Transmit FIFO is empty
- 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 Target 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

The 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 Target IP Core allows the host inside the FPGA to communicate with an external SPI Controller device. The data size of the SPI transaction can be configured to be 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. SPI Target 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 Target 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 Target IP has a standard SPI bus interface.

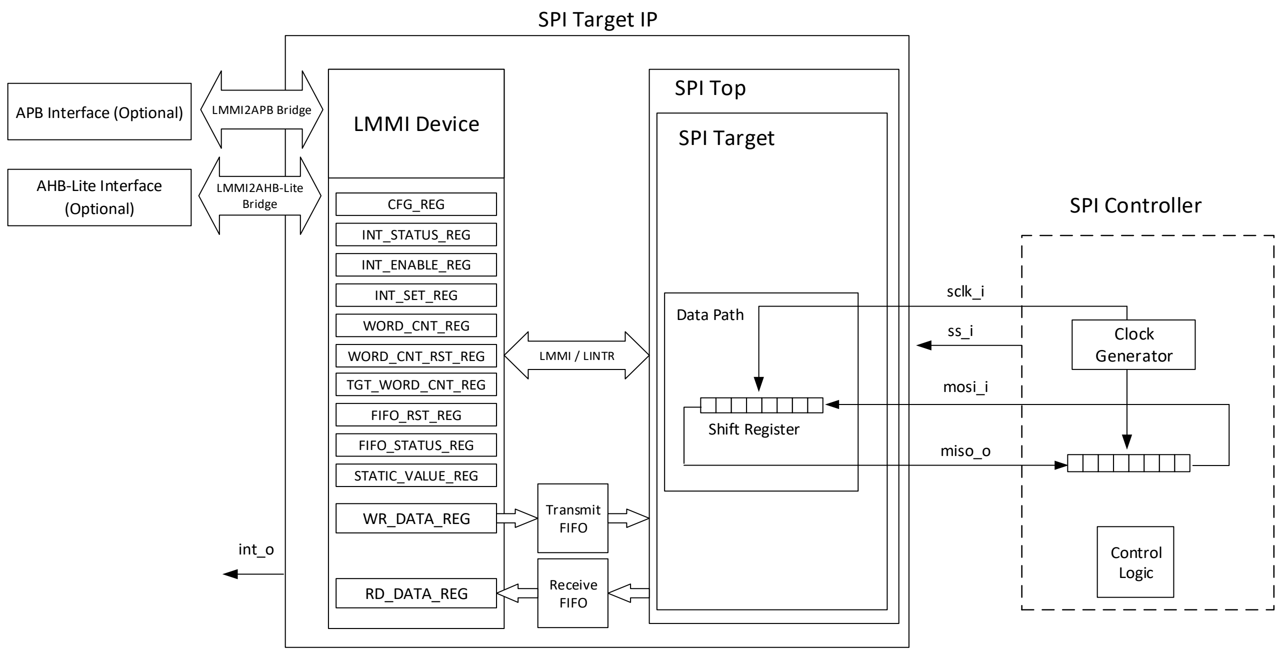


Figure 2.1. SPI Target IP Core Functional Diagram

The SPI Target IP includes the following layers:

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

2.2. Clocking

There are three clocks for the SPI Target IP.

- clk_i: system clock , used to drive the entire IP.
- lmmi_clk: manage clock, used to configure registers.
- sclk_i: input clk, used to receive data from peripherals.

2.2.1. Clocking Overview

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

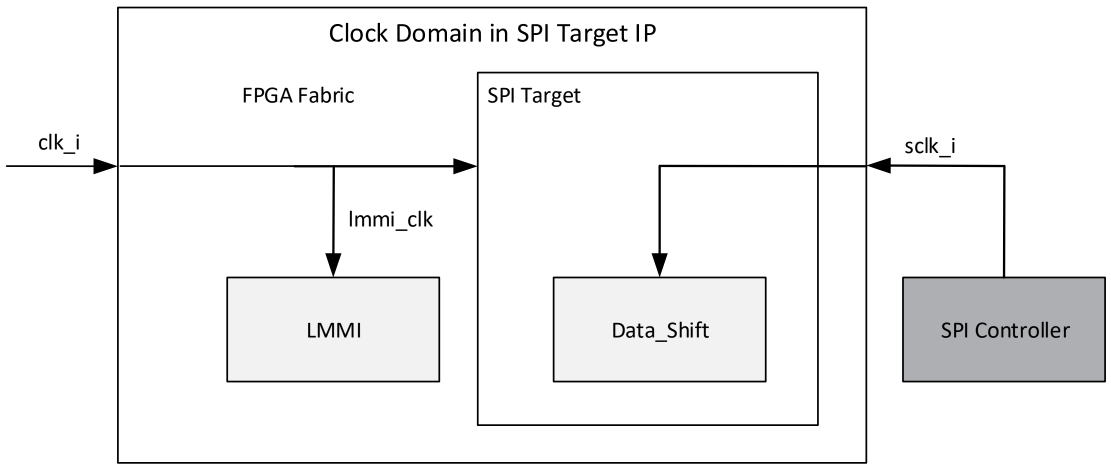


Figure 2.2. SPI Target IP Clock Domain Block Diagram

Note: clk_i is limited to 10-200 MHz and should be at least four times of sclk_i.

2.3. Reset

There is one reset for the SPI Target 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 Target 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. • lmmi_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 Target IP core and external SPI controller 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. Operations Details

2.5.1. Data Transmission

To begin SPI transaction, the external SPI Controller Device sends the clock signal and selects the SPI Target by asserting the ss_i bit for the corresponding 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 via the MOSI and MISO lines respectively. During SPI transaction, the data is simultaneously transmitted (shifted out serially onto the miso_o signal) and received (the mosi_i is sampled/shifted-in). The serial clock (sclk_i) edge synchronizes the shifting and sampling of the data. For more information, see the Clocking Modes: Clock Polarity and Clock Phase section. The first sclk_i edge after ss_i assertion shall occur after at least four clk_i cycles to ensure that data to be transmitted through miso_o is ready.

The bit size of each data word transferred during SPI transaction is set by the *Data Width* attribute.

2.5.2. Clocking Modes: Clock Polarity and Clock Phase

The external SPI Controller Device 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 – ss_i bits are in inactive logic based on Chip Select Polarity Register. Transitioning of ss_i bit(s) from inactive logic to active logic marks the start of the transmission and 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 external SPI Controller Device 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, this is shown in Table 2.2.

Table 2.2. Clocking Modes (CPOL and CPHA)

Clocking Mode	CPOL	CPHA	Action
0	0	0	The ss_i bit transition to active logic shifts out a data bit. The first clock transition samples the data. Data is sampled on the sclk_i 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_i falling edge and is shifted out on the rising edge.
2	1	0	The ss_i bit transition to active logic shifts out a data bit. The first clock transition samples the data. Data is sampled on the sclk_i 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 captures the data. Data is sampled on the sclk_i 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.

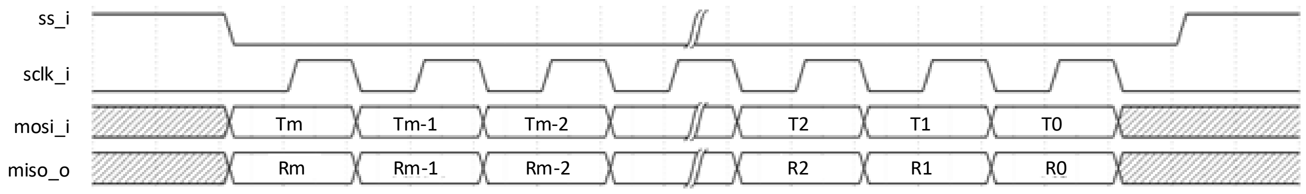


Figure 2.3.Clocking Mode 0 (CPOL=0, CPHA=0)

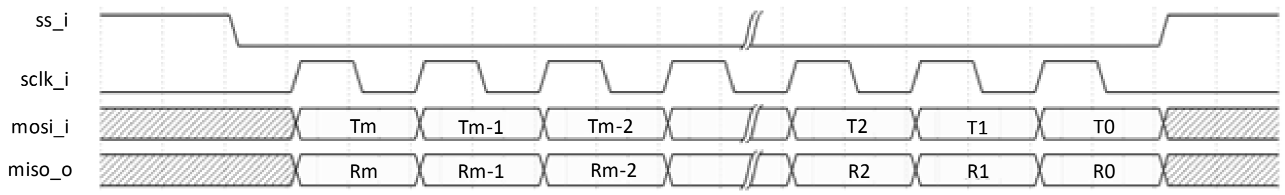


Figure 2.4. Clocking Mode 1 (CPOL=0, CPHA=1)

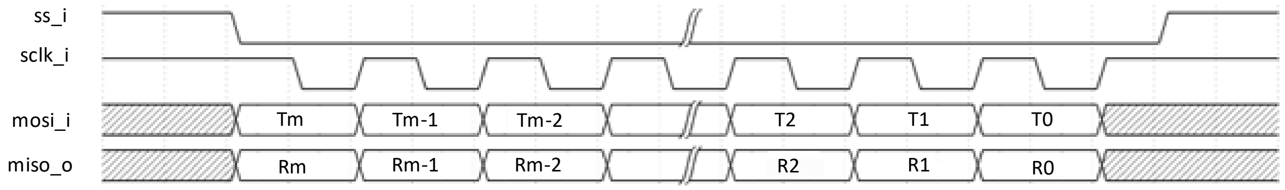


Figure 2.5. Clocking Mode 2 (CPOL=1, CPHA=0)

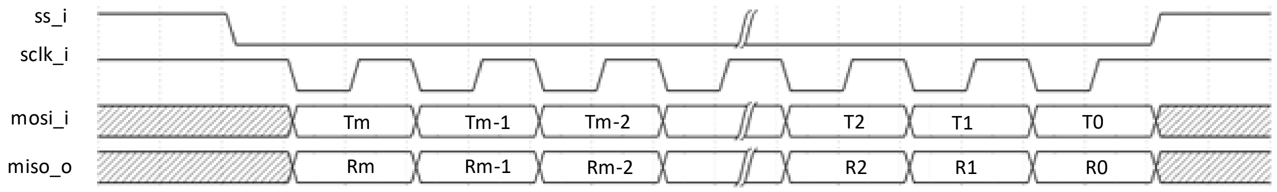


Figure 2.6. Clocking Mode 3 (CPOL=1, CPHA=1)

2.6. Independent Target Configuration

In the independent target configuration, there is an independent chip select line for each target, as shown in Figure 2.7. This is the way SPI is normally used. The external SPI Controller asserts only one chip select at a time.

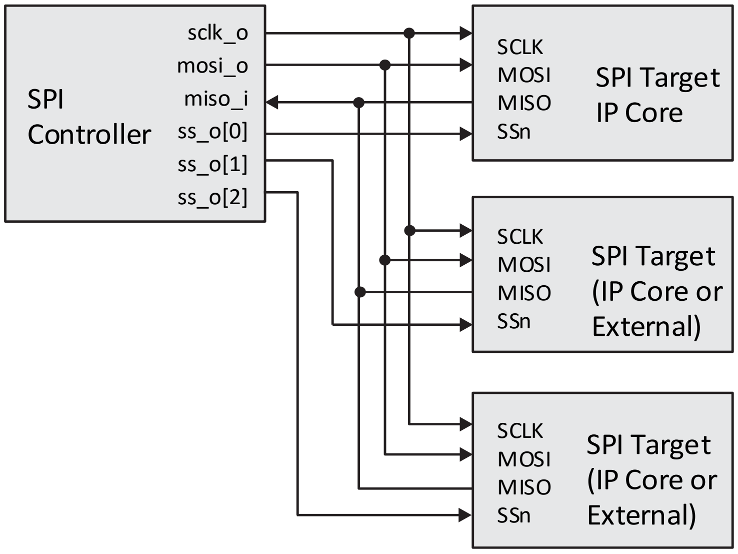


Figure 2.7. Independent Target Configuration

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

The SPI Target IP Core also supports daisy chain configuration, as shown in Figure 2.8. The first target output is connected to the second target input and so on. 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.

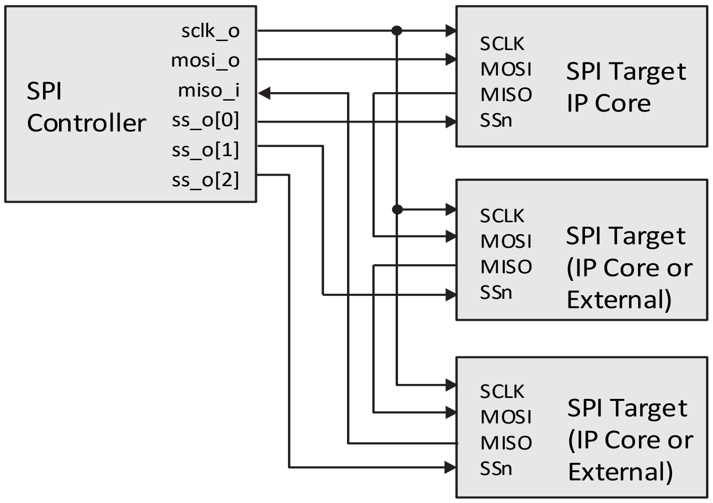


Figure 2.8. Daisy-Chain Configuration

Ensure that the following conditions are met in the SPI Target IP:

- SPI Target IP Transmit FIFO is empty.
- MISO Static Value is disabled.

2.8. Programming Flow

Below are the recommended steps for performing the target SPI transaction shown in the Figure 2.9 using interrupt or polling mode. The SPI transaction is composed of command (1 byte), address (3 bytes), dummy (4 bytes), and data phase
(n words, 1 word = 4 bytes). The host gets the command and address bytes from the SPI Target IP core to determine the
n-word data that is provided. The total data bytes are both known for the SPI controller and target. Dummy phase is necessary here to provide the host enough time to configure the SPI Target IP core and to write at least one data word to WR_DATA_REG.

The data width of the SPI Target IP core is set to 32 bits.

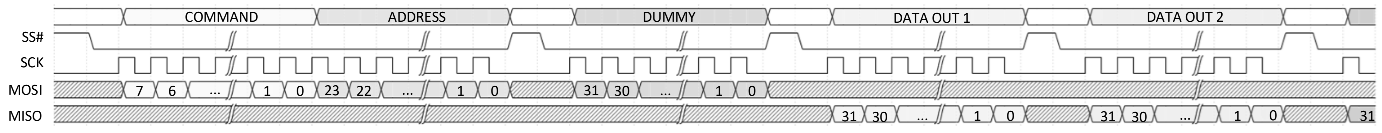


Figure 2.9. Target SPI Transaction

2.8.1. Transmit/Receive Operation Interrupt Mode

1. Set the CFG_REG fields according to the target SPI transaction.
2. Enable target interrupts in INT_ENABLE_REG.
3. Wait for interrupt, int_o, then read INT_STATUS_REG. The expected pending interrupt here is rx_fifo_ready_int, which indicates that the SPI target has received the first word for SPI command and address.
4. Clear the pending interrupt in INT_STATUS_REG.

5. Reset WORD_CNT_REG by writing 8’hFF to WORD_CNT_RST_REG.
6. Set the target n-word data in TGT_WORD_CNT_REG based on the SPI command and address.
7. 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.

   During SPI dummy phase, the host must write at least one data word to WR_DATA_REG. Writing of n-word can overlap until the SPI data phase.
8. Clear the pending interrupts as needed.
9. 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.
10. Clear the pending interrupt in INT_STATUS_REG.
11. Use FIFO_RST_REG.rx_fifo_rst to reset the Receive FIFO since the receive data for this SPI transaction can be discarded.

2.8.2. Transmit/Receive Operation Polling Mode

1. Set the CFG_REG fields according to the target SPI transaction.
2. Poll FIFO_STATUS_REG.rx_fifo_empty to check if SPI target has received the first data word for SPI command and address.
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.
   a. If target n-word is greater than the Transmit FIFO depth, read FIFO_STATUS_REG.tx_fifo_full before writing data to WR_DATA_REG.
   b. During SPI dummy phase, the host must write at least one data word to WR_DATA_REG. Writing of n-word can overlap until the SPI data phase.
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. Use FIFO_RST_REG.rx_fifo_rst to reset the Receive FIFO since the receive data for this SPI transaction can be discarded.

3. IP Parameter Description

The configurable attributes of the SPI Target 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.
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.
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
SPI Clock Polarity	**0**, 1	Specifies the reset value of CFG_REG.cpol.
SPI Clock Phase	**0**, 1	Specifies the reset value of CFG_REG.cpha.
Enable Daisy Chain	Checked, **Unchecked**	Specifies the reset value of CFG_REG.daisy_chain.
Chip Select Polarity	**Active Low**, Active High	Specifies the polarity of ss_i. Active Low – 1'b0 Active High – 1'b1
Enable MISO Static Value	Checked, **Unchecked**	Specifies the reset value of CFG_REG.sval_en.
MISO Static Value (Hex)	**00000000**	Specifies the reset value of STATIC_VALUE_REG.sval. Available only if *Enable MISO Static Value* is checked.
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.
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 *Data Width*.
FIFO Depth	**16**, 32, 64, 128, 256, 512	Specifies the number of FIFO levels. Only power of 2 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–(*FIFO Depth* – 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–(*FIFO Depth* – 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.

4. Signal Description

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

Table 4.1. SPI Target IP Signal Description

Port	Width	Type	Output Reset Value	Description
Clock and Reset				
clk_i	1	Input	—	System clock
rst_n_i	1	Input	—	Reset signal Resets the core, LMMI/APB/AHB-Lite interfaces and sets registers to their default values.
Interrupt Port				
int_o	1	Output	1'b0	Interrupt signal
SPI Interface				
sclk_i	1	Input	—	Serial clock Generated by the SPI Controller to synchronize data transfers. In order for the SPI Target to operate properly, the system clock frequency should be at least four times that of the serial clock.
mosi_i	1	Input	—	Controller Out, Target In (data output from controller) signal Transfers data going from the Controller to the Target.
ss_i	1	Input	—	Chip Select signal Asserted by the Controller to begin data transfer. Polarity is determined by the *Chip Select Polarity* attribute.
miso_o	1	Output	High impedance	Controller In, Target Out (data output from target) signal Transfers data going to the Controller from the Target.
LMMI Interface[1]				
lmmi_request_i	1	Input	—	Start 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	*Data Width*[2]	Input	—	Output data bus
lmmi_rdata_o	*Data Width*[2]	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 Interface[1]				
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, halfword, or word.
ahbl_hmastlock_i	1	Input	—	AHB-Lite Lock signal This signal is unused.

Port	Width	Type	Output Reset Value	Description
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 Interface[1]				
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 Bits [31:8] are not used.
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 available registers. 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 Target 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	CFG_REG	RW	Configuration Register
0x2	0x08	INT_STATUS_REG	RW1C	Interrupt Status Register
0x3	0x0C	INT_ENABLE_REG	RW	Interrupt Enable Register
0x4	0x10	INT_SET_REG	WO	Interrupt Set Register
0x5	0x14	WORD_CNT_REG	RO	Word Count Register
0x6	0x18	WORD_CNT_RST_REG	WO	Word Count Reset Register
0x7	0x1C	TGT_WORD_CNT_REG	RW	Target Word Count Register
0x8	0x20	FIFO_RST_REG	WO	FIFO Reset Register
0x9	0x24	FIFO_STATUS_REG	RO	FIFO Status Register
0xA	0x28	STATIC_VALUE_REG	RW	MISO Static Value Register

Note:

1. 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 register bit clears the bit to 1’b0. Writing 1’b0 on 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
[*Data Width*–1:0]	tx_fifo	Writing to WR_DATA_REG pushes a word to the Transmit FIFO. When the SPI Target IP Core is selected and a transfer is initiated by an SPI controller, the IP core pops one data word and sends it via the SPI interface. If Transmit FIFO is empty during the SPI transfer, the transmitted data in MISO depends on CFG_REG.sval_en. The timing of writing data to WR_DATA_REG should be determined by the requirements of the SPI transfer. The frequency difference between the system clock and SPI clock should be considered. 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 is 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
[*Data Width*—1:0]	rx_fifo	Reading from RD_DATA_REG pops a word from the Receive FIFO. During an SPI transaction, the received data in MOSI 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 Receive FIFO empty condition. On the other hand, if Receive FIFO is full and this register is not read but the IP core 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. 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 5.5. Configuration Register

Field	Name	Description	Access	Default[1]
[7]	sval_en	MISO Static Value Enable Enables the assigning of static value, STATIC_VALUE_REG, in MISO when Transmit FIFO is empty. 1'b0 – Data from MOSI (previous word) loops back to MISO when the Transmit FIFO is empty. 1'b1 – STATIC_VALUE_REG is assigned to MISO when Transmit FIFO is empty.	RW	*Enable MISO Static Value*
[6]	ss_pol	Chip Select polarity Specifies the active level of chip select signal, ss_i. 1'b0 – Active Low 1'b1 – Active High	RW	*Chip Select Polarity*
[5:4]	ds	Data Size The *Data Width* 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 16 bits. 2'b10 – Data size 24 bits. 2'b11 – Data size 32 bits.	RO	*Data Width*
[3]	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	*First Transmitted Bit*
[2]	daisy_chain	Daisy Chain Defines the daisy chain operation mode. 1'b0 – Enables independent target configuration mode. Refer to the Independent Target Configuration section for more information. 1'b1 – Enables Daisy Chain configuration mode. Refer to Daisy Chain Configuration section for more information.	RW	*Enable Daisy Chain*
[1]	cpol	Clock Polarity Specifies the logic level of sclk_i during idle. 1'b0 – sclk_i is set to 1'b0 when idle. 1'b1 – sclk_i is set to 1'b1 when idle.	RW	*SPI Clock Polarity*
[0]	cpha	Clock Phase Specifies clock edge for shifting out data in miso_o and sampling data in mosi_i. 1'b0 – The first clock transition is the first data capture edge. 1'b1 – The second clock transition is the first data capture edge.	RW	*SPI Clock Phase*

Note:

1. Refer to Table 3.1 for the attribute values.

5.5. Interrupt Status Register (INT_STATUS_REG)

The Interrupt Status Register contains all the interrupts currently pending in the SPI Target 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 5.6. 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 via 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 TX FIFO Almost Empty Flag – 1 to TX FIFO Almost Empty Flag. 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 RX FIFO Almost Full Flag – 1 to RX FIFO Almost Full Flag. 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 the SPI interface. 1'b0 – No interrupt 1'b1 – Interrupt pending	RW1C	1'b0

5.6. 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 bit asserts and the corresponding INT_ENABLE_REG bit is 1'b1, the interrupt signal int_o asserts.

Table 5.7. 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

5.7. Interrupt Set Register (INT_SET_REG)

The Interrupt Set Register is intended for testing purpose 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 5.8. 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 corresponding 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 corresponding to Receive FIFO Ready Interrupt Status. 0 – No action 1 – Asserts INT_STATUS_REG.rx_fifo_ready_int	WO	1’b0

5.8. Word Count Register (WORD_CNT_REG)

Table 5.9. Word Count Register

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

5.9. Word Count Reset Register (WORD_CNT_RST_REG)

Table 5.10. 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

5.10. Target Word Count Register (TGT_WORD_CNT_REG)

Table 5.11. 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

5.11. FIFO Reset Register (FIFO_RST_REG)

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

Table 5.12. FIFO Reset Register

Field	Name	Description	Access	Default
[7:2]	reserved	Reserved	RSVD	—
[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

5.12. FIFO Status Register (FIFO_STATUS_REG)

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

Table 5.13. 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 the *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 Almost 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 the *RX FIFO Almost Full Flag* attribute.	RO	1'b0
[0]	rx_fifo_empty	Receive FIFO Empty 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

5.13. MISO Static Value Register (STATIC_VALUE_REG)

Table 5.14. MISO Static Value Register

Field	Name	Description	Access	Default[1]
[*Data Width*–1:0]	sval	The MISO Static Value Register is used for assigning a specific value to MISO when Transmit is empty.	RW	*MISO Static Value*

Note:

1. Refer to Table 3.1 for the attribute value.

6. Example Design

The SPI Target example design allows you to compile, simulate, and test the SPI Target 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 Target IP example design.

Table 6.1. SPI Target IP Configuration Used on Example Design

SPI Target IP GUI Parameter	SPI Target IP Configuration
General	
Interface	APB
First Transmitted Bit	MSB
Data Width	32
SPI Clock Polarity	0
SPI Clock Phase	0
Enable Daisy Chain	×
Chip Select Polarity	Active Low
Enable MISO Static Value	✓
MISO Static Value (Hex)	A5A5A5A5
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

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 Target IP to external SPI controller or target devices.

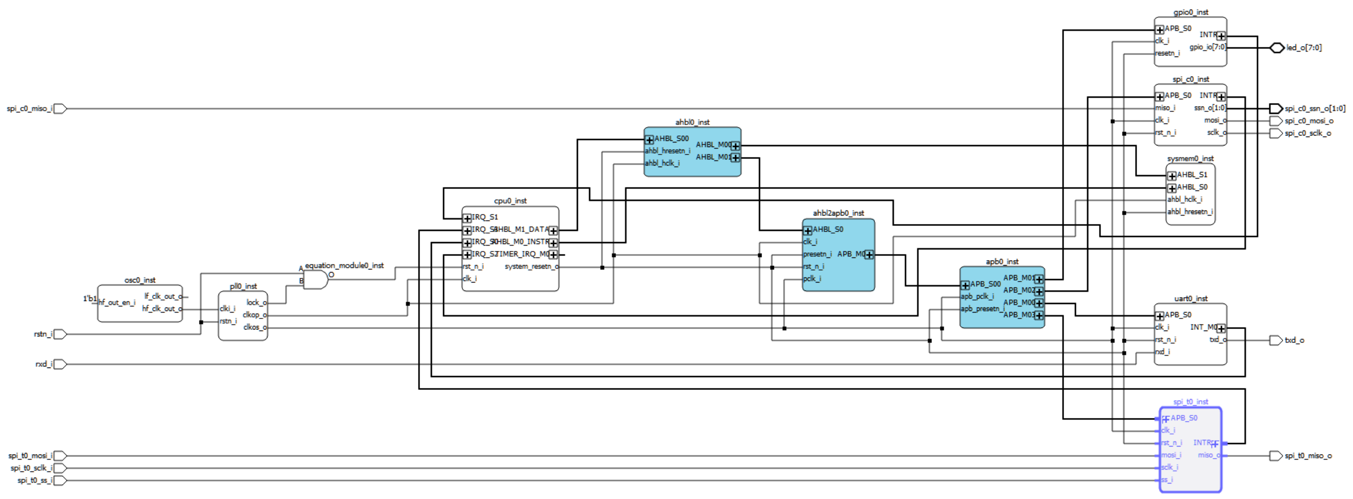
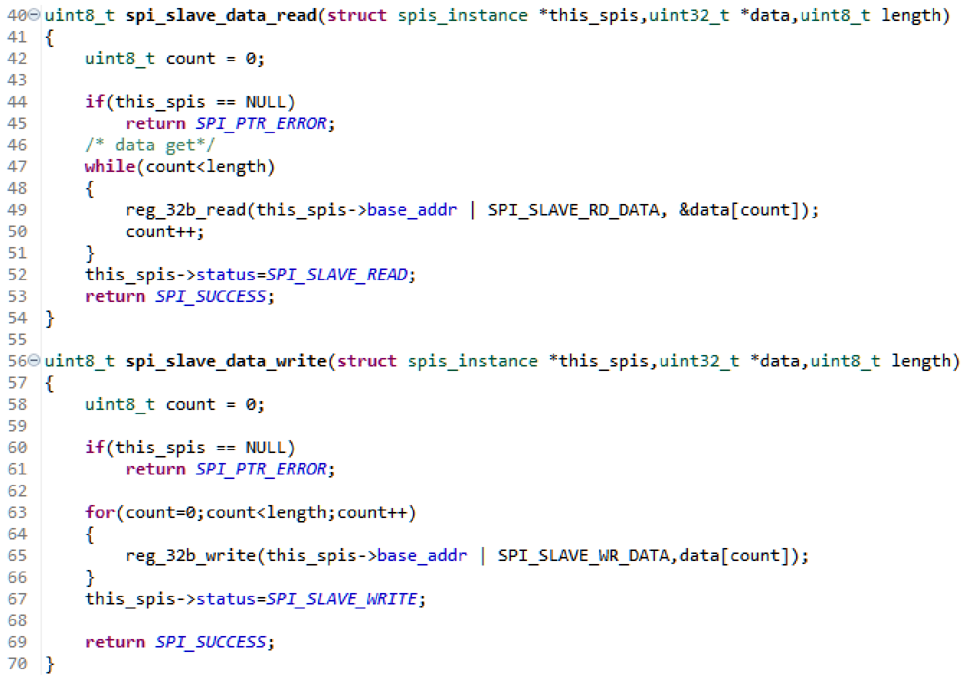


Figure 6.1. SPI Target IP Example Design in Propel SoC Project

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 Target 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 write to *WR_DATA_REG* and read from *RD_DATA_REG*. This sample test routine can be found in the driver file included in the generated IP.



```
uint8_t spi_slave_data_read(struct spis_instance *this_spis,uint32_t *data,uint8_t length)
{
    uint8_t count = 0;

    if(this_spis == NULL)
        return SPI_PTR_ERROR;
    /* data get*/
    while(count<length)
    {
        reg_32b_read(this_spis->base_addr | SPI_SLAVE_RD_DATA, &data[count]);
        count++;
    }
    this_spis->status=SPI_SLAVE_READ;
    return SPI_SUCCESS;
}

uint8_t spi_slave_data_write(struct spis_instance *this_spis,uint32_t *data,uint8_t length)
{
    uint8_t count = 0;

    if(this_spis == NULL)
        return SPI_PTR_ERROR;

    for(count=0;count<length;count++)
    {
        reg_32b_write(this_spis->base_addr | SPI_SLAVE_WR_DATA,data[count]);
    }
    this_spis->status=SPI_SLAVE_WRITE;

    return SPI_SUCCESS;
}
```

Figure 6.2. Sample C-Code Test Routine

6.3. Design Components Example

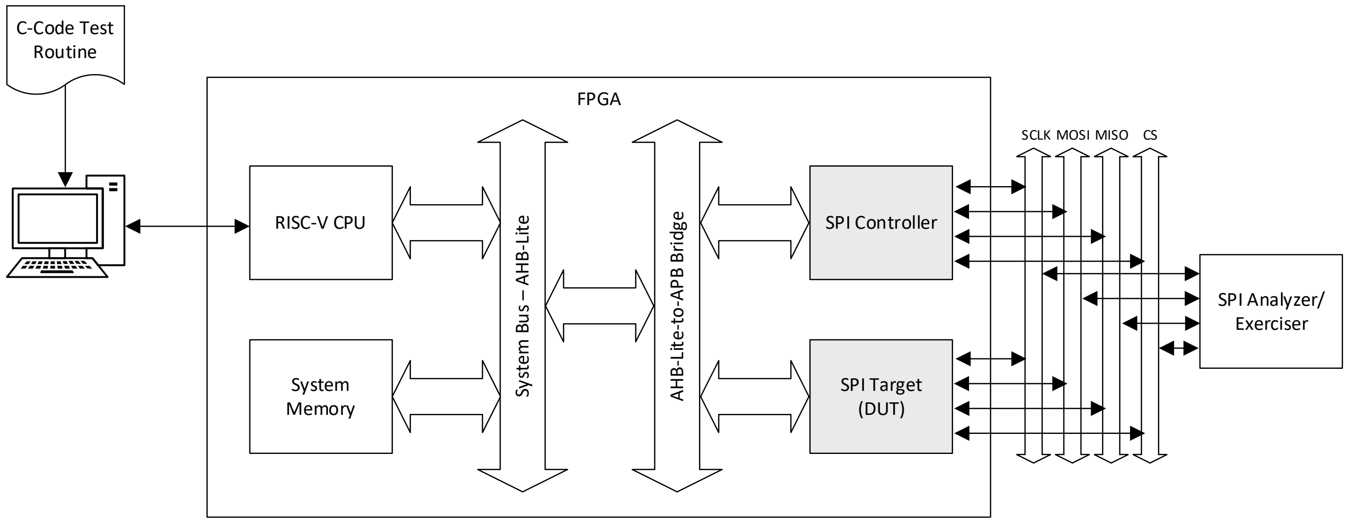


Figure 6.3. SPI Controller Example Design Block Diagram

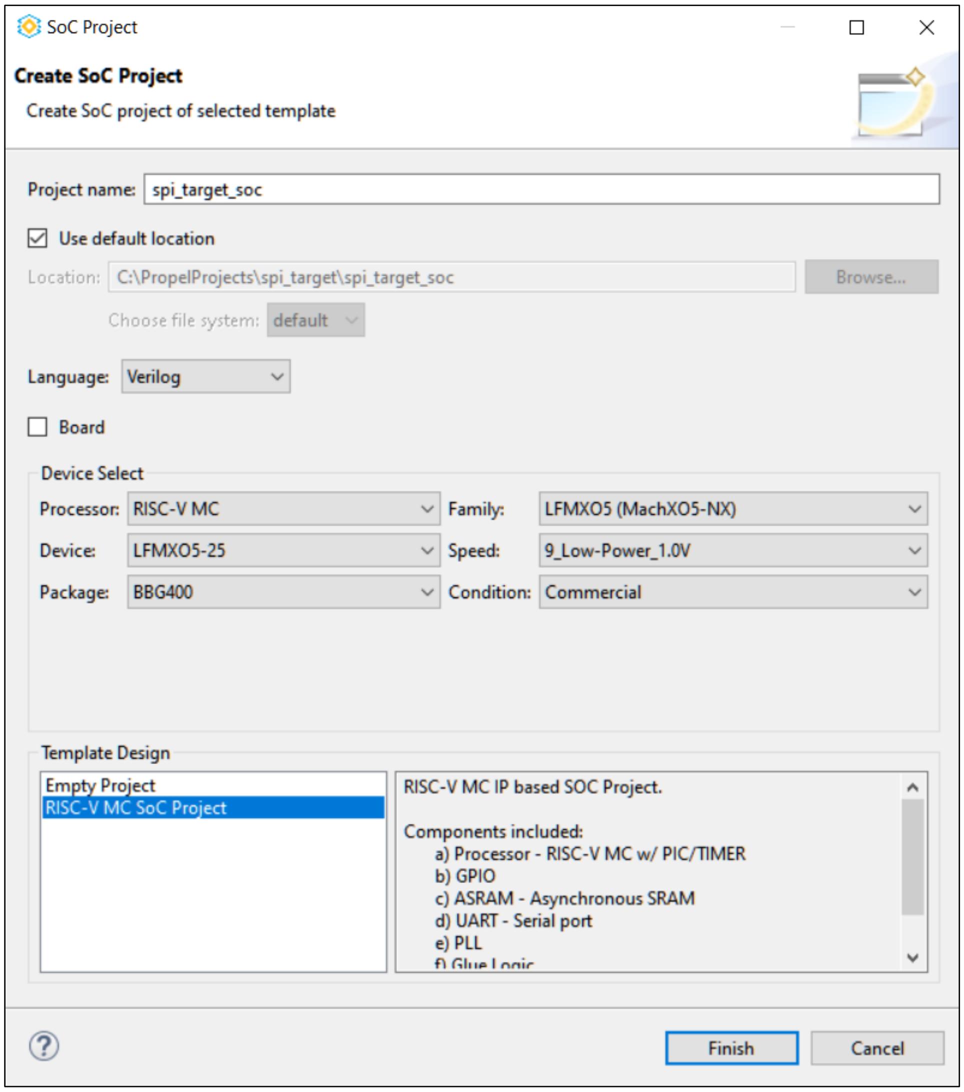
The SPI Controller example design includes the following blocks:

- 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 system bus for transfers between memory and IP
- SPI Target IP – IP instance connected to SPI bus (MISO, MOSI, CS, SCLK)
- SPI controller/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 the 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**.



Project name: spi_target_soc
Use default location [✓]
Location: C:\PropelProjects\spi_target\spi_target_soc
Choose file system: default
Language: Verilog
Board [ ]

Device Select

Processor:	RISC-V MC	Family:	LFMXO5 (MachXO5-NX)
Device:	LFMXO5-25	Speed:	9_Low-Power_1.0V
Package:	BBG400	Condition:	Commercial

Template Design
Empty Project
RISC-V MC SoC Project

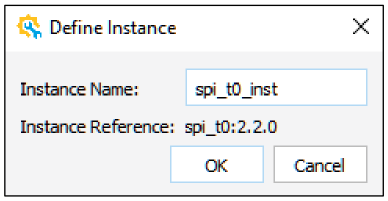
RISC-V MC IP based SOC Project.

Components included:
a) Processor - RISC-V MC w/ PIC/TIMER
b) GPIO
c) ASRAM - Asynchronous SRAM
d) UART - Serial port
e) PLL
f) Glue Logic

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 Target 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.

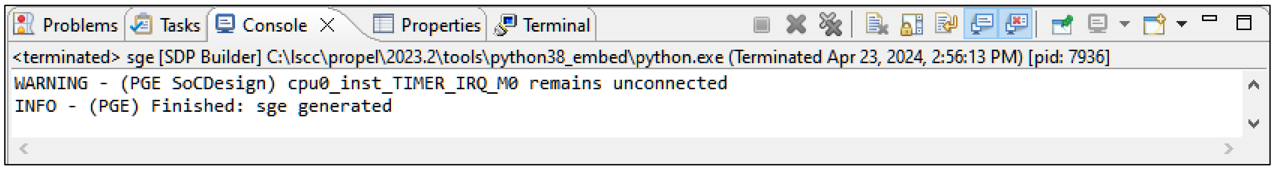
8. After generating the IP, the **Define Instance** window opens. Modify the instance name if needed, then click **OK.**



Instance Name: spi_t0_inst
Instance Reference: spi_t0:2.2.0

Figure 6.5. Define Instance

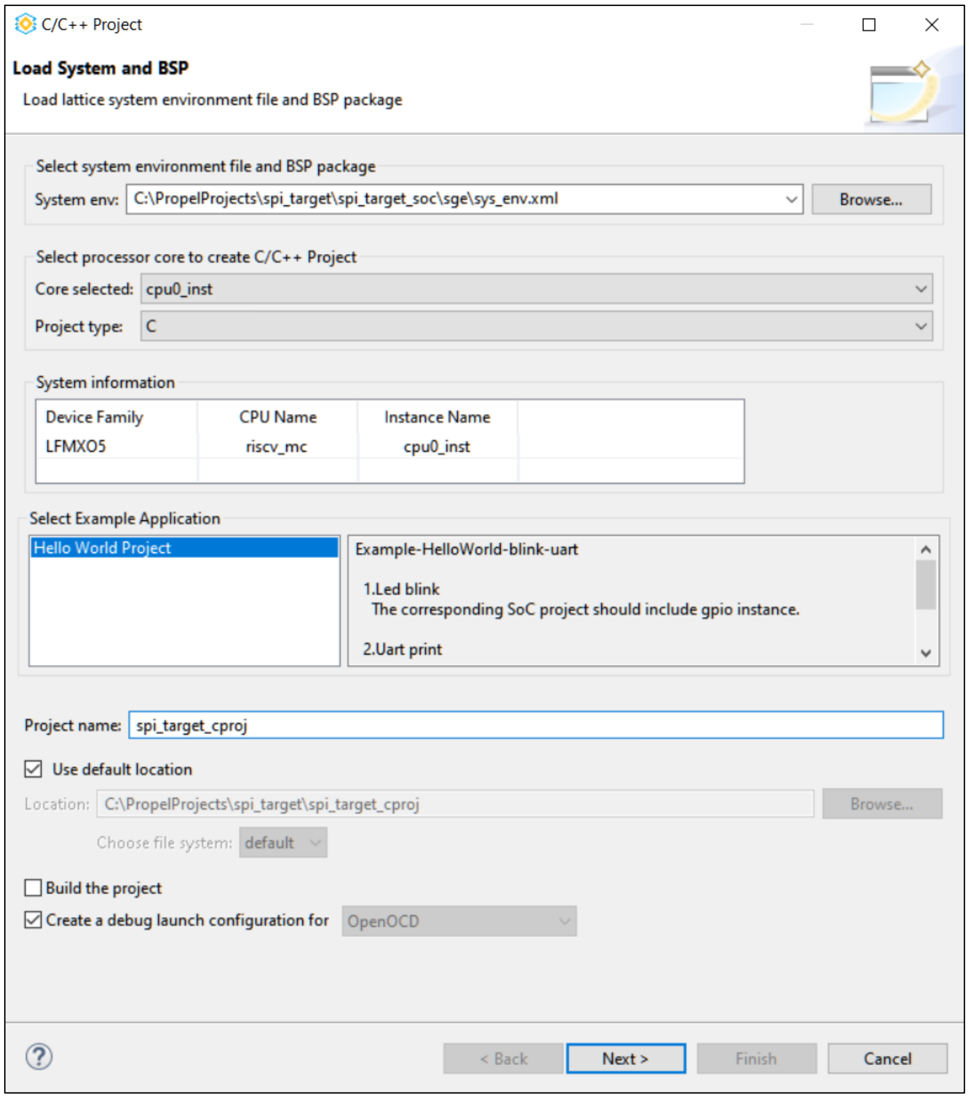
9. Connect the instantiated IPs to the system. Refer to Figure 6.1 for the connections used in this IP. You will need to update other components of the system for clock and reset sources, interrupt, and bus interface.
10. Click the icon or navigate to **Design** > **Run Radiant** to launch the Lattice 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.



```
<terminated> sge [SDP Builder] C:\lscc\propel\2023.2\tools\python38_embed\python.exe (Terminated Apr 23, 2024, 2:56:13 PM) [pid: 7936]
WARNING - (PGE SoCDesign) cpu0_inst_TIMER_IRQ_M0 remains unconnected
INFO - (PGE) Finished: sge generated
```

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**.


C/C++ Project
Load System and BSP
Load lattice system environment file and BSP package
Select system environment file and BSP package
System env: C:\PropelProjects\spi_target\spi_target_soc\sge\sys_env.xml
Browse...
Select processor core to create C/C++ Project
Core selected: cpu0_inst
Project type: C
System information
Select Example Application
Hello World Project
Project name: spi_target_cproj
Use default location
Location: C:\PropelProjects\spi_target\spi_target_cproj
Choose file system: default
Build the project
Create a debug launch configuration for OpenOCD
< Back
Next >
Finish
Cancel

Device Family	CPU Name	Instance Name
LFMXO5	riscv_mc	cpu0_inst

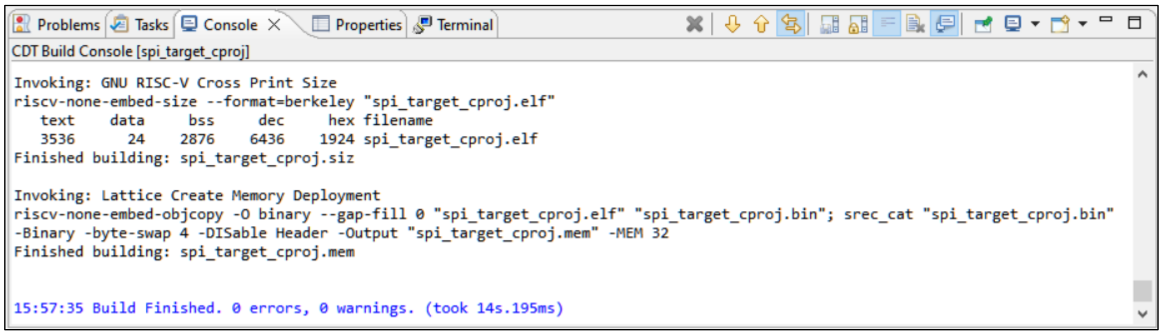
Example-HelloWorld-blink-uart

1.Led blink
The corresponding SoC project should include gpio instance.

2.Uart print

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.



```
Invoking: GNU RISC-V Cross Print Size
riscv-none-embed-size --format=berkeley "spi_target_cproj.elf"
   text    data     bss     dec     hex filename
   3536      24    2876    6436    1924 spi_target_cproj.elf
Finished building: spi_target_cproj.siz

Invoking: Lattice Create Memory Deployment
riscv-none-embed-objcopy -O binary --gap-fill 0 "spi_target_cproj.elf" "spi_target_cproj.bin"; srec_cat "spi_target_cproj.bin"
-Binary -byte-swap 4 -DISable Header -Output "spi_target_cproj.mem" -MEM 32
Finished building: spi_target_cproj.mem


15:57:35 Build Finished. 0 errors, 0 warnings. (took 14s.195ms)
```

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 a 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.

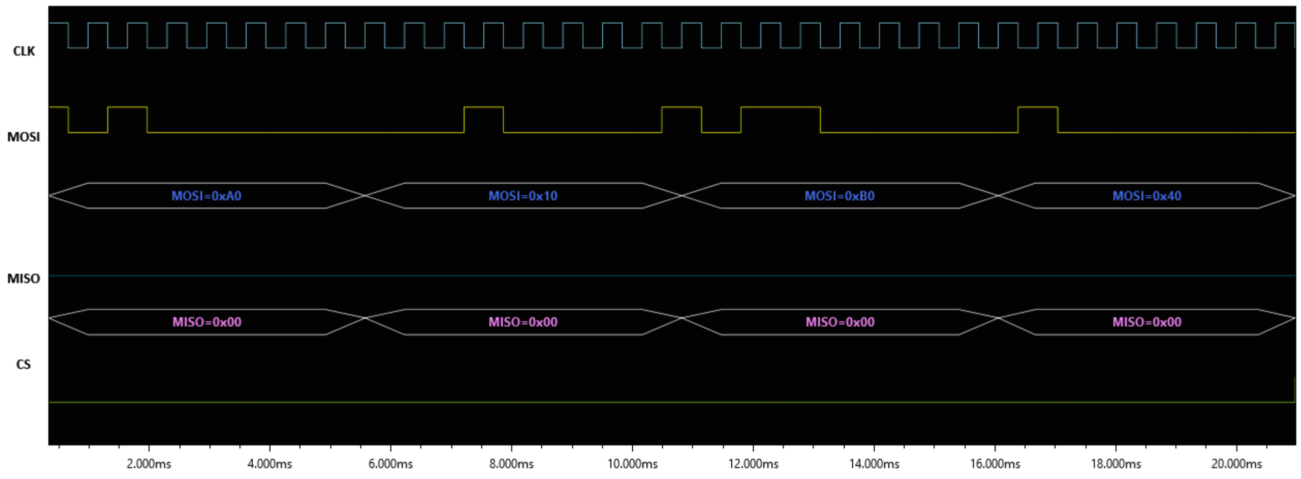


Figure 6.9. Sample SPI Transaction between Controller and Target

7. Designing with the IP

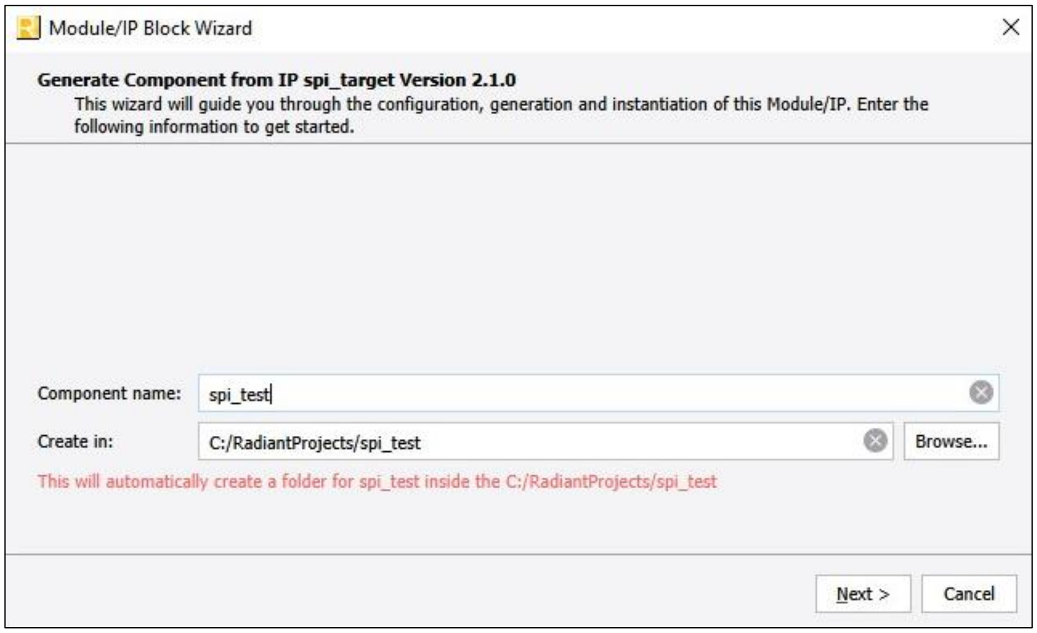
This section provides information on how to generate the IP Core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

7.1. Generating and Instantiating the IP

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

To generate the SPI Target IP:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **SPI Target** 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**.



Module/IP Block Wizard

Generate Component from IP spi_target Version 2.1.0

This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Enter the following information to get started.

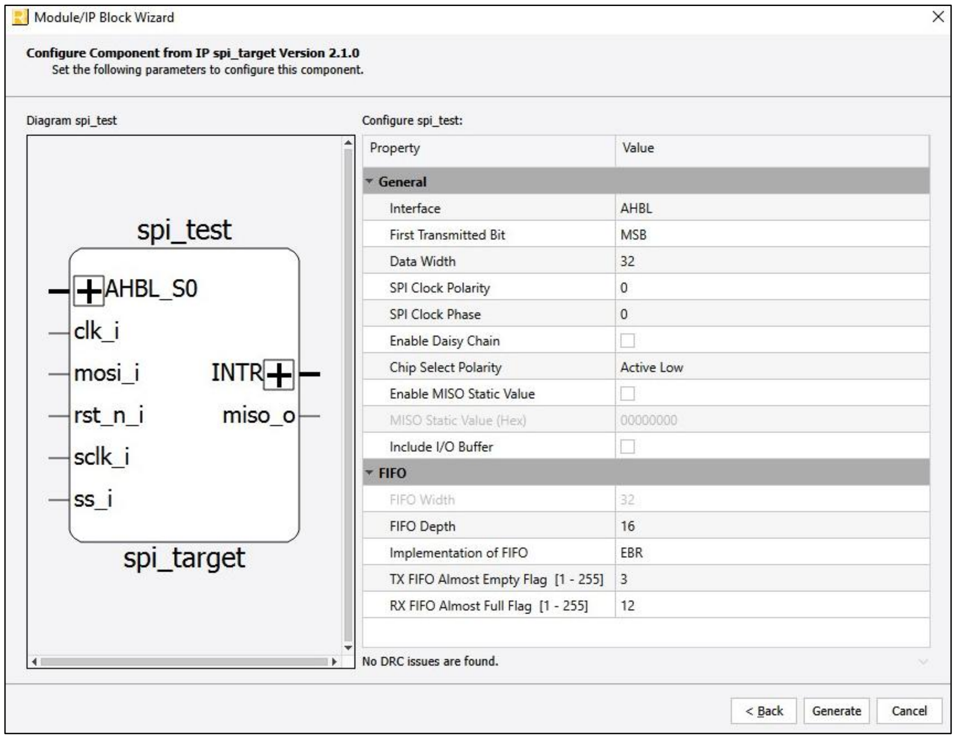
Component name: spi_test

Create in: C:/RadiantProjects/spi_test

This will automatically create a folder for spi_test inside the C:/RadiantProjects/spi_test

Figure 7.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected SPI Target IP using drop-down lists and check boxes. Figure 7.2 shows an example configuration of the SPI Target IP. For details on the configuration options, refer to the IP Parameter Description section.



Configure Component from IP spi_target Version 2.1.0
Set the following parameters to configure this component.

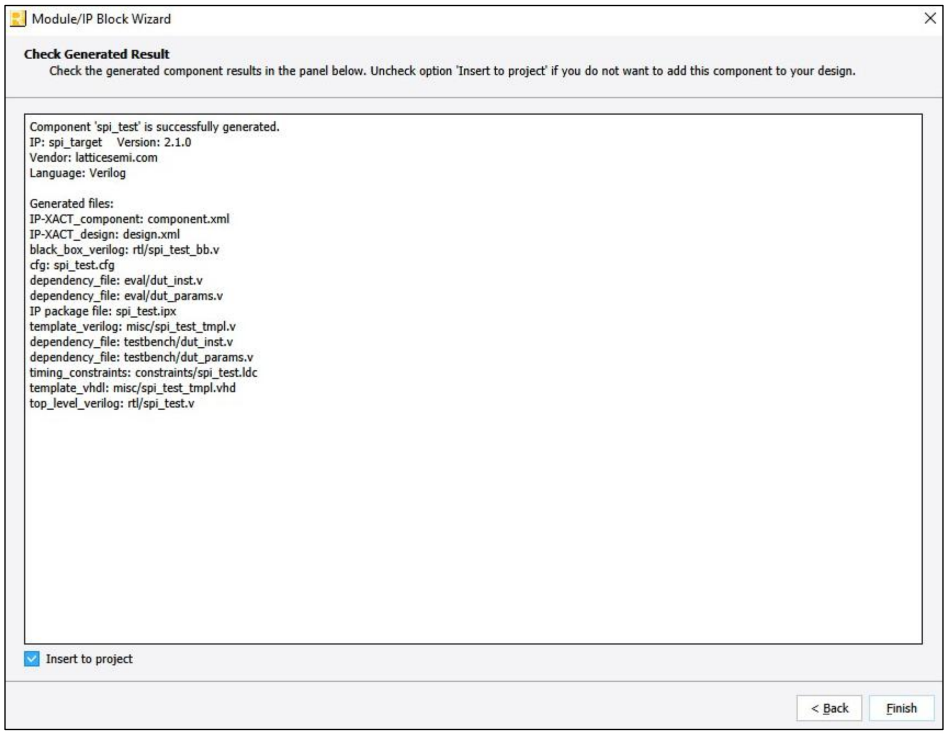
Configure spi_test:

Property	Value
General	
Interface	AHBL
First Transmitted Bit	MSB
Data Width	32
SPI Clock Polarity	0
SPI Clock Phase	0
Enable Daisy Chain	☐
Chip Select Polarity	Active Low
Enable MISO Static Value	☐
MISO Static Value (Hex)	00000000
Include I/O Buffer	☐
FIFO	
FIFO Width	32
FIFO Depth	16
Implementation of FIFO	EBR
TX FIFO Almost Empty Flag [1 - 255]	3
RX FIFO Almost Full Flag [1 - 255]	12

No DRC issues are found.

Figure 7.2. IP Configuration

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



Check Generated Result
Check the generated component results in the panel below. Uncheck option 'Insert to project' if you do not want to add this component to your design.

```
Component 'spi_test' is successfully generated.
IP: spi_target    Version: 2.1.0
Vendor: latticesemi.com
Language: Verilog

Generated files:
IP-XACT_component: component.xml
IP-XACT_design: design.xml
black_box_verilog: rtl/spi_test_bb.v
cfg: spi_test.cfg
dependency_file: eval/dut_inst.v
dependency_file: eval/dut_params.v
IP package file: spi_test.ipx
template_verilog: misc/spi_test_tmpl.v
dependency_file: testbench/dut_inst.v
dependency_file: testbench/dut_params.v
timing_constraints: constraints/spi_test.ldc
template_vhdl: misc/spi_test_tmpl.vhd
top_level_verilog: rtl/spi_test.v
```

☑ Insert to project

Figure 7.3. Check Generated Result

5. 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 Target IP 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 top-level for their complete 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.

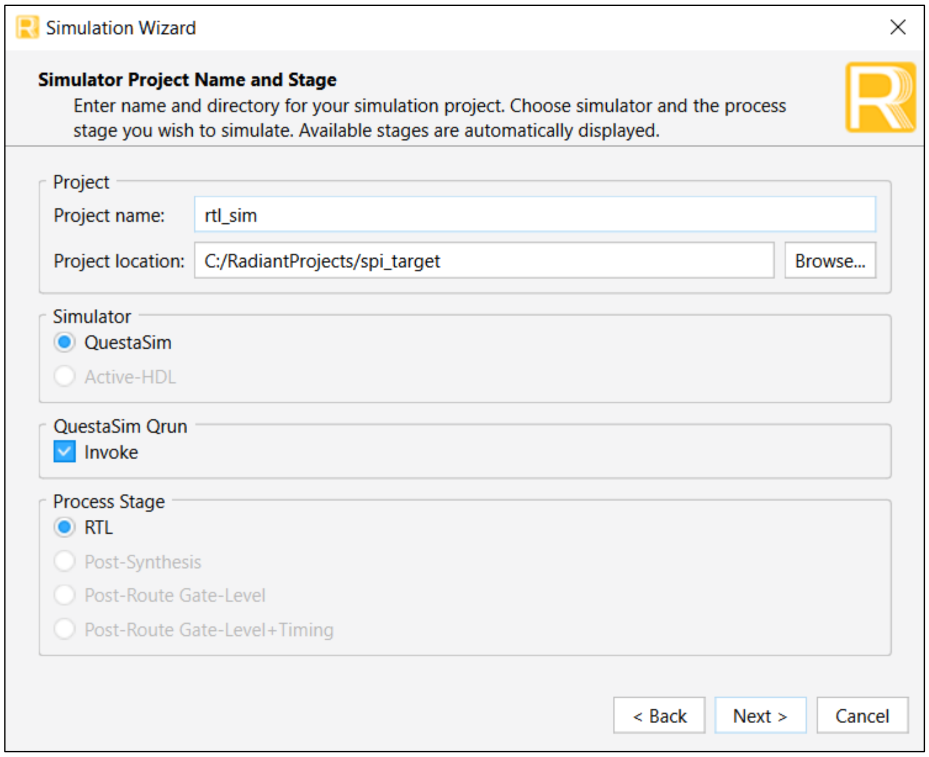
Refer to the relevant sections in the Lattice Radiant Software User Guide for more information on how to create or edit constraints and how to use the Device Constraint Editor.

7.3. Running Functional Simulation

You can run functional simulation after the IP is generated.

To run functional simulation:

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


Simulation Wizard

Simulator Project Name and Stage
Enter name and directory for your simulation project. Choose simulator and the process stage you wish to simulate. Available stages are automatically displayed.

Project
Project name: rtl_sim
Project location: C:/RadiantProjects/spi_target

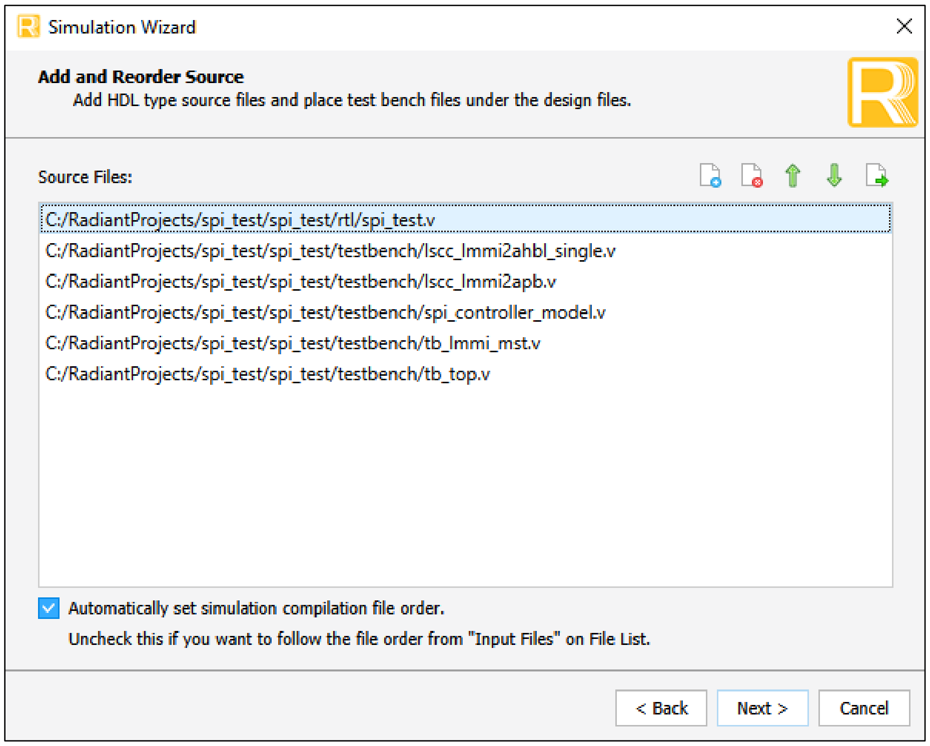
Simulator
[✓ QuestaSim]
Active-HDL

QuestaSim Qrun
[✓ Invoke]

Process Stage
[✓ RTL]
Post-Synthesis
Post-Route Gate-Level
Post-Route Gate-Level+Timing

Figure 7.4. Simulation Wizard

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


Simulation Wizard

Add and Reorder Source
Add HDL type source files and place test bench files under the design files.

Source Files:
C:/RadiantProjects/spi_test/spi_test/rtl/spi_test.v
C:/RadiantProjects/spi_test/spi_test/testbench/lscc_lmmi2ahbl_single.v
C:/RadiantProjects/spi_test/spi_test/testbench/lscc_lmmi2apb.v
C:/RadiantProjects/spi_test/spi_test/testbench/spi_controller_model.v
C:/RadiantProjects/spi_test/spi_test/testbench/tb_lmmi_mst.v
C:/RadiantProjects/spi_test/spi_test/testbench/tb_top.v

[✓ Automatically set simulation compilation file order.]
Uncheck this if you want to follow the file order from "Input Files" on File List.

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 simulation result.

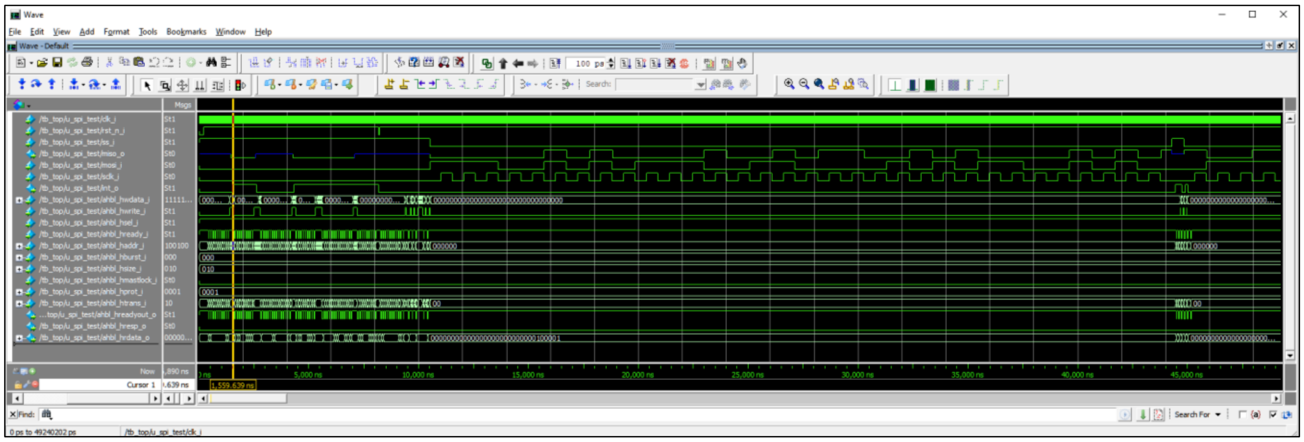


Figure 7.6. Simulation Waveform

7.3.1. Simulation Results

For Example:

Figure 7.7 shows AHB-Lite write operation, write the data (32’h3b23f176) to address (32’h00000004).

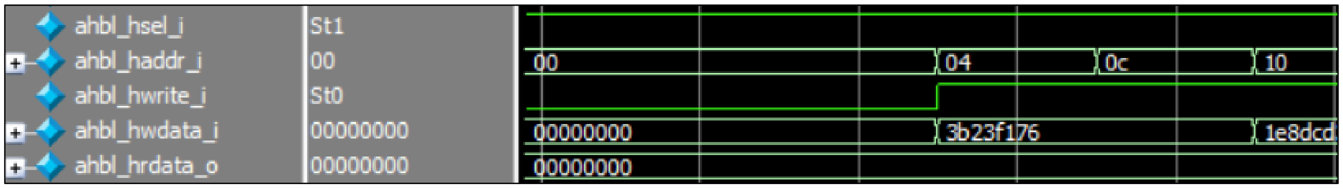


Figure 7.7. AHB-Lite Write Operation

Figure 7.8 shows AHB-Lite read operation, read the data (32’h00000019) to address (32’h00000028).

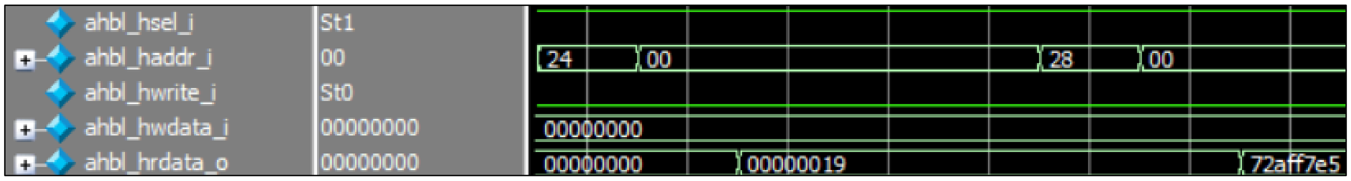


Figure 7.8. AHB-Lite Read Operation

Figure 7.9 shows SPI Target Transaction.

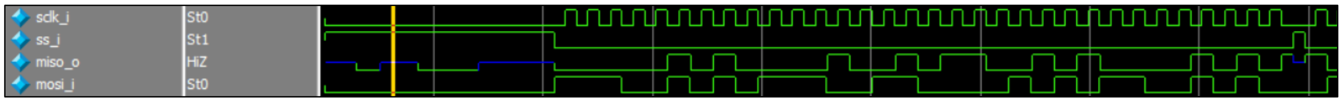


Figure 7.9. SPI Target Transaction

7.4. Constraining the IP

It is your responsibility to provide proper timing and physical design constraints to ensure that the design meets the desired performance goals on the FPGA. The content of the following IP constraint file may be added to the user design constraints: <IP_Instance_Path>/<IP_Instance_Name>/eval/constraint.pdc

The above constraint file has been verified during IP evaluation with the IP instantiated directly at the top-level module. The constraints in this file can be modified given a complete understanding of the effect of each constraint.

To use this constraint file:

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

Refer to Lattice Radiant Timing Constraints Methodology (FPGA-AN-02059) for details on how to constrain the user design.

Appendix A. Resource Utilization

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

Table A.1. LAV-AT-E70-3LFG1156I Resource Utilization

Configuration	Clk f_{MAX} (MHz)[1]	Slice Registers	LUTs	EBRs
Default	250.00	479	569	2
Interface: APB, Others = Default	228.89	397	414	2
Interface: LMMI, Others = Default	250.00	354	427	2
TX FIFO Almost Empty Flag: 1, *RX FIFO Almost Full Flag*: 1, Others = Default	250.00	479	564	2
FIFO Depth: 256, *TX FIFO Almost Empty Flag*: 255, *RX FIFO Almost Full Flag*: 255, Others = Default	250.00	591	664	2

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Target 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 Target IP Core for the LFMXO5-25-9BBG400I device, using Lattice Synthesis Engine of Lattice Radiant software. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.2. LFMXO5-25-9BBG400I Resource Utilization

Configuration	Clk f_{MAX} (MHz)[1]	Slice Registers	LUTs	EBRs
Default	179.57	476	601	2
Interface: APB, Others = Default	158.76	393	421	2
Interface: LMMI, Others = Default	186.08	352	436	2
TX FIFO Almost Empty Flag: 1, *RX FIFO Almost Full Flag*: 1, Others = Default	162.02	476	602	2
FIFO Depth: 256, *TX FIFO Almost Empty Flag*: 255, *RX FIFO Almost Full Flag*: 255, Others = Default	143.72	588	743	2

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Target 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 Target IP Core for the LFMXO5-25-7BBG400I device using Synplify Pro of Lattice Radiant software 3.1. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.3. LFMXO5-25-7BBG400I Resource Utilization

Configuration	Clk f_{MAX} (MHz)[1]	Slice Registers	LUTs	EBRs
Default	125.24	476	601	2
Interface: APB, Others = Default	138.97	393	421	2
Interface: LMMI, Others = Default	170.74	352	436	2
TX FIFO Almost Empty Flag: 1, *RX FIFO Almost Full Flag*: 1, Others = Default	150.04	476	602	2
FIFO Depth: 256, *TX FIFO Almost Empty Flag*: 255, *RX FIFO Almost Full Flag*: 255, Others = Default	131.49	588	743	2

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Target 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 Target 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.4. LN2-CT-20-1CBG484C Resource Utilization

Configuration	Clk f_{MAX} (MHz)[1]	Slice Registers	LUTs	EBRs
Default	250	476	634	2
Interface: APB, Others = Default	250	393	440	2
Interface: LMMI, Others = Default	250	352	457	2
TX FIFO Almost Empty Flag: 1, *RX FIFO Almost Full Flag*: 1, Others = Default	247.097	476	633	2
FIFO Depth: 256, *TX FIFO Almost Empty Flag*: 1, *RX FIFO Almost Full Flag*: 255, Others = Default	250	588	789	2

Note:

1. f_{MAX} is generated when the FPGA design only contains SPI Target 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 Target IP Release Notes (FPGA-RN-02014)
- 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 Target 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 2.2, IP v2.4.0, July 2025

Section	Change Summary
Introduction	• In Table 1.1. Summary of the SPI Target IP: ◦ Renamed Supported FPGA Family to Supported Devices and rearranged items. ◦ Removed Targeted Devices row. ◦ Added IP core version to Lattice Implementation. • In Table 1.2. SPI Target IP Support Readiness: ◦ Updated header to *System Clock Frequency*. ◦ Made minor editorial changes.
References	• Added SPI Target IP Core web page. • Updated listing name to Lattice Radiant Software web page.

Revision 2.1, 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 *AHB-Lite*, *DUT*, *EBR*, *GPIO*, *MC*, *PIC*, *SRAM*, and *UART*.

Introduction	• In Table 1.1. Summary of the SPI Target IP:
	  ◦ Added Certus-NX-RT, CertusPro-NX-RT, and Certus-N2 to *Supported FPGA Family*.
	  ◦ Added IP Changes row.
	  ◦ Removed IP Version row.
	  ◦ Added LFD2NX-9, LFD2NX-28, and LN2-CT-20 to *Targeted Devices*.
	  ◦ Added Resources row.
	  ◦ Added IP Core v2.3.0 to *Lattice Implementation*.
	• 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 description on conditions to meet in the SPI Target IP in the Daisy Chain Configuration section. • Updated description in the Data Transmission section under the Operations Details section.
IP Parameter Description	• Added statement regarding default values in bold. • In Table 3.1. General Attributes: ◦ 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 value range for TX FIFO Almost Empty Flag and RX FIFO Almost Full Flag. • Removed Table 3.2. Attributes Descriptions.
Signal Description	In Table 4.1. SPI Target 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:

Section	Change Summary
	◦ Merged register bit/field information into existing tables. ◦ Restructured tables to include Description column and removed Width column. • 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.4. LN2-CT-20-1CBG484C Resource Utilization.
Appendix B. Limitations	Removed section.
References	• Removed statement about Lattice Radiant Software User Guide. • Added SPI Target 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 2.0, January 2024

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 *Overview* to *IP Architecture Overview*; ◦ newly added *the SPI Target IP includes the following layers: APB/AHB-Lite Interface, LMMI Device, SPI Top, SPI Target.* • 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. General Attributes and Table 3.2. Attributes Descriptions.
Signal Description	Moved the original subsection 2.2 Signal Description from Functional Description to this section.
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 *IP Generation and Evaluation* to *Designing with the IP*. • Changed the subsection name from *Generation and Synthesis* to *Generating and Instantiating the IP*. • 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 1.9, December 2023

Section	Change Summary
All	• Updated title from *SPI Target IP Core - Lattice Radiant Software* to *SPI Target IP Core*. • Updated instances of *LAV-AT-500E* to *LAV-AT-E70*.
Disclaimers	Updated this section.

Section	Change Summary
Introduction	Added new devices *LAV-AT-G70*, and *LAV-AT-X70* to Table 1.1. Quick Facts.
Functional Description	Updated *FIFO Depth* in Table 2.2. Attributes Table.
IP Generation and Evaluation	• Added *eval/constraint.pdc* to Table 3.1. Generated File List. • Added section 3.4. Constraining the IP. • Updated section 3.6. Hardware Validation.
Resource Utilization	Updated the *Clk Fmax*, *Slice Registers*, and *LUTs* values of all configurations in Table A.1. LAV-AT-E70-3LFG1156I Resource Utilization.
Limitation	Updated this section.
References	Updated this section.

Revision 1.8, June 2023

Section	Change Summary
Inclusive Language	Added this section.
All	• Updated terminologies to replace with the following: • Master is replaced with Controller • Slave is replaced with Target • Slave Select is replaced with Chip Select
Introduction	• Updated Table 1.1. Quick Facts for below: • Updated Target devices with *LIFCL-40, LIFCL-33, LIFCL-17, LFD2NX-40, LFD2NX-17, LFCPNX-50, LFCPNX-100, LFMXO5-25, LFMXO5-55T, LFMXO5-100T, UT24C40, UT24CP100, LAV-AT-500E*. • Updated IP Core version to 2.0.0 and Lattice Radiant Software version to 2022.1. • Update with below features in Features section: • *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* • *Back-end LMMI Interface* • *Supports all SPI Clocking Modes (combination of Clock Polarity and Clock Phase)* • *Sending of defined static value when Transmit FIFO is empty* • *Independent target configuration and daisy chain configuration* • *Interrupt capability* • *Selectable memory-mapped Interface: AHB-Lite, APB or LMMI*
Functional Description	• Updated below figures to replace *slave* with *target*: • Figure 2.1. SPI Target IP Core Functional Diagram • Figure 2.6. Independent Target Configuration • Figure 2.7. Daisy-Chain Configuration • Figure 2.8. Target SPI Transaction
IP Generation and Evaluation	• Updated below figures for updating as per IP spi_target version 2.0.0: • Figure 3.1. Module/IP Block Wizard • Figure 3.2. Configure User Interface of SPI Target IP Core • Figure 3.3. Check Generating Result • Updated the Simulator wizard in Figure 3.4. Simulation Wizard and Figure 3.5. Adding and Reordering Source. • Updated simulation waveform in Figure 3.6. Simulation Waveform.
Reference	Added links for Crosslink-NX, Certus-NX, CertusPro-NX, MachXO5-NX, and Lattice Avant-E.

Revision 1.7, November 2022

Section	Change Summary
IP Generation and Evaluation	• Changed the title of Section 3.4 from Hardware Evaluation to IP Evaluation. • Added Section 3.5.

Section	Change Summary
Appendix A. Resource Utilization	• Added Table A.1. LAV-AT-500E-3LFG1156I Resource Utilization. • Updated values of Clk Fmax (MHz) and Slice Registers in Table A.2. LFMXO5-25-9BBG400I Resource Utilization and Table A.3. LFMXO5-25-7BBG400I Resource Utilization.

Revision 1.6, April 2022

Section	Change Summary
Introduction	Added Sending of define static value when Transmit FIFO is empty in Features.
Functional Description	• Updated TX FIFO Almost Empty Flag and RX FIFO Almost Full Flag minimum value from 0 to 1 in Table 2.2. Attributes Table. • Added Enable MISO Static Value and MISO Static Value (Hex) to both Table 2.2. Attributes Table and Table 2.3. Attributes Descriptions. • Added STATIC_VALUE_REG in Table 2.4. Summary of SPI Slave IP Core Registers. • Updated Write Data Register description. • Updated Read Data Register description. • Added sval_en as bit [7] in Configuration Register description. • Added MISO Static Value Register description. • Updated Programming Flow details and separate the contents to Transmit/Receive Operation Interrupt Mode and Transmit/Receive Operation Polling Mode. • Added Figure 2.8. Target SPI Transaction.
IP Generation and Evaluation	Updated Figure 3.1. Module/IP Block Wizard, Figure 3.2. Configure User Interface of SPI Slave IP Core and Figure 3.3. Check Generating Result.
Appendix A. Resource Utilization	Updated resource utilization information.

Revision 1.5, November 2021

Section	Change Summary
Functional Description	Updated Table 2.2. Attributes Table. Changed Selectable Values and Dependency on Other Attributes of TX FIFO Almost Empty Flag and RX FIFO Almost Full Flag attributes
References	Updated reference to Lattice Radiant software user guide.

Revision 1.4, June 2021

Section	Change Summary
Introduction	• Remove second paragraph. • Updated Table 1.1. Quick Facts. • Revised Supported FPGA Families • Revised Targeted Devices • Revised Lattice Implementation • Revised reference to Lattice Radiant Software User Guide
Appendix B. Limitations	Updated *Devices Affected*.
References	Revised reference to Lattice Radiant Software User Guide and removed link.

Revision 1.3, October 2020

Section	Change Summary
Acronyms in This Document	Updated the table content.
Functional Description	Added missing AHB-Lite and APB signals in Table 2.1.

Revision 1.2, June 2020

Section	Change Summary
Introduction	Updated Table 1.1.
Functional Description	Updated AHB-Lite and APB Interfaces in Selectable Memory-Mapped Interface section.
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.
All	Minor formatting changes

Revision 0.80, October 2019

Section	Change Summary
All	Preliminary release.

www.latticesemi.com