LogiCORE IP AXI Quad SPI v3.1

Product Guide for Vivado Design Suite

PG153 December 18, 2013

XILINX®

Table of Contents

IP Facts

Chapter 1: Overview

Feature Summary 12
Unsupported Features 15
Licensing and Ordering Information 15

Chapter 2: Product Specification

Standards 16
Performance 16
Resource Utilization 18
Port Descriptions 21
Register Space 37
Specification Exceptions 58

Chapter 3: Designing with the Core

General Design Guidelines 61
Clocking 72
Resets 73
Protocol Description 73

Chapter 4: Customizing and Generating the Core

Vivado Integrated Design Environment 89
Output Generation 91

Chapter 5: Constraining the Core

Chapter 6: Simulation

Chapter 7: Synthesis and Implementation

Chapter 8: Example Design

Overview 95
Implementing the Example Design 96
Testing the Example Design on a KC705 Board 98

Chapter 9: Test Bench

Overview 100
Simulating the Example Design 101

Appendix A: Migrating and Upgrading

Migrating to the Vivado Design Suite 102
Upgrading in the Vivado Design Suite 102

Appendix B: Debugging

Finding Help on Xilinx.com 103
Debug Tools 104
Hardware Debug 105
Interface Debug 105

Appendix C: Additional Resources

Xilinx Resources 108
References 108
Revision History 109
Notice of Disclaimer 110

Introduction

The LogiCORE™ IP AXI Quad Serial Peripheral Interface (SPI) core connects the AXI4 interface to those SPI slave devices that support the Standard, Dual or Quad SPI protocol instruction set. This core provides a serial interface to SPI slave devices. The Dual/Quad SPI is an enhancement to the Standard SPI protocol (described in the Motorola M68HC11 data sheet) and provides a simple method for data exchange between a master and a slave.

Features

- Configurable AXI4 interface; when configured with an AXI4-Lite interface the core is backward compatible with version 1.00 of the core (Legacy mode)
- Configurable AXI4 interface for burst mode operation for Data Receive Register (DRR) and Data Transmit Register (DTR) FIFO
- Configurable eXecute In Place (XIP) mode of operation
- Connects as a 32-bit slave on either AXI4-Lite or AXI4 memory mapped interface
- Configurable SPI modes:
  - Standard SPI mode
  - Dual SPI mode
  - Quad SPI mode
- Standard SPI interfaces:
  - IO0 (MOSI) - Standard SPI mode
  - IO1 (MISO) - Standard SPI mode
  - SCK - Standard and Dual SPI mode
  - x $\overline{SS}$ - Standard and Dual SPI mode
- In Quad SPI mode supports six signal interfaces: IO0, IO1, IO2, IO3, SCK and $\overline{SS}$
- Up to 32 configurable SPI slaves
- Programmable SPI clock phase and polarity
- Configurable FIFO depth (16 or 256 element deep in Dual/Quad/Standard SPI mode) and fixed FIFO depth of 64 in XIP mode

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale™ Architecture, Zynq®-7000 All Programmable SoC, 7 Series FPGAs
Supported User Interfaces	AXI4, AXI4-Lite
Resources	See Table 2-3 through Table 2-5
Provided with Core	
Design Files	RTL
Example Design	VHDL
Test Bench	VHDL
Constraints File	XDC
Simulation Model	Not Provided
Supported S/W Driver[2]	Standalone and Linux
Tested Design Flows[3]	
Design Entry	Vivado® Design Suite Vivado IP integrator
Simulation	For a list of supported simulators, see the Xilinx Design Tools: Release Notes Guide
Synthesis	Vivado synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete list of supported devices, see Vivado IP catalog.
2. Standalone driver details can be found in the SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from wiki.xilinx.com.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The top-level block diagram for the AXI Quad SPI core when configured with the AXI4-Lite interface option is shown in Figure 1-1.

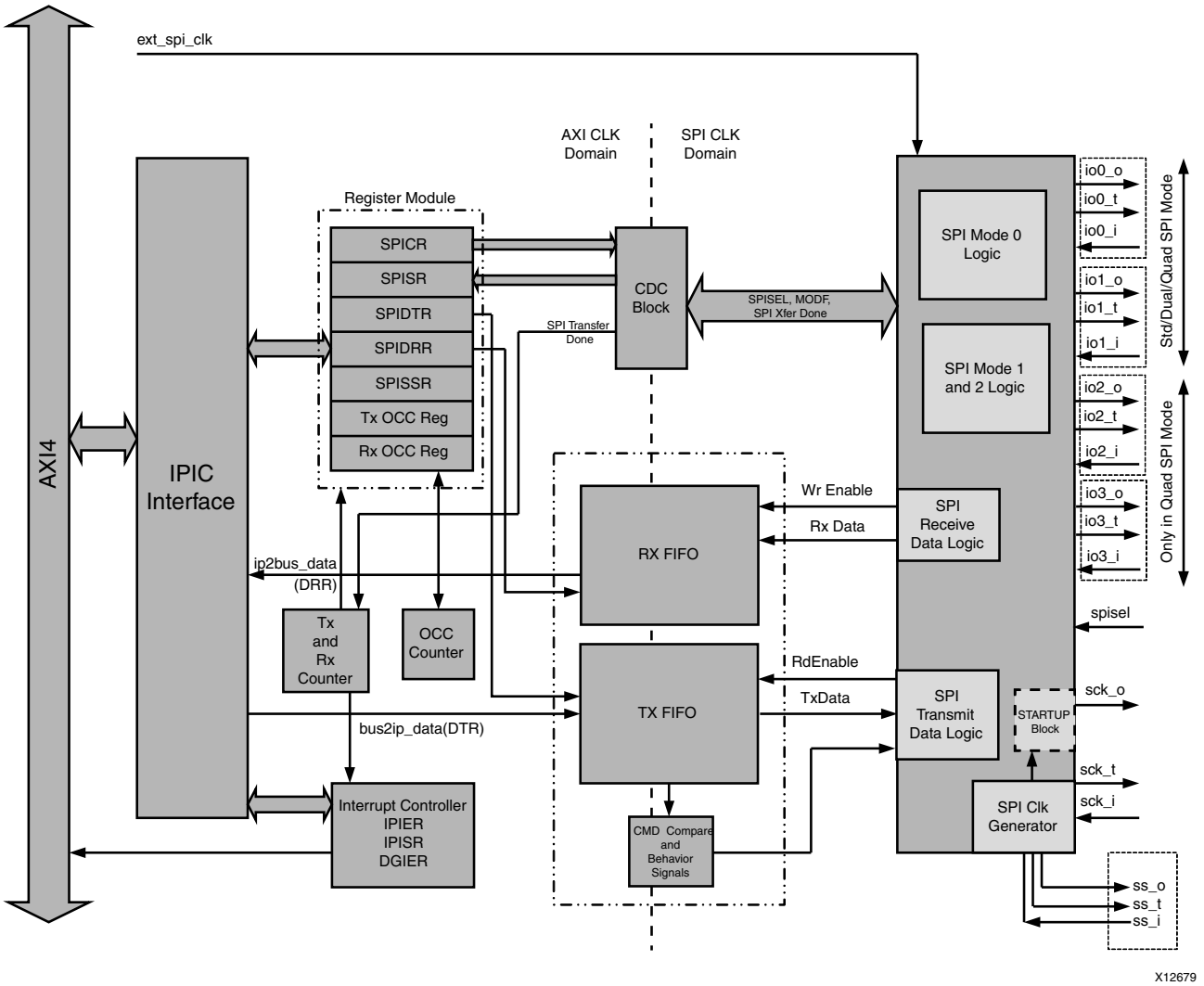

Figure 1-1: **AXI Quad SPI Core Top-Level Block Diagram**

The choice of either AXI4-Lite or AXI4 memory-mapped interface is based on the **Enable Performance Mode** option in the Vivado Integrated Design Environment (IDE) (see Chapter 4). Performance mode is disabled by default which selects the AXI4-Lite interface. The core always operates as a slave IP when the AXI4 interface is selected.

Legacy Mode

Legacy mode is selected when the **Enable Performance Mode** option in the Vivado IDE is disabled. Performance mode uses the AXI4-Lite interface and is fully backward compatible with all the older versions of AXI Quad SPI core in terms of functionality, register bit placement and register access.

The AXI Quad SPI IP core, when configured in standard SPI mode, is a full-duplex synchronous channel which supports a four-wire interface (receive, transmit, clock and slave-select) between a master and a selected slave. When configured in Dual/Quad SPI mode, this core supports additional pins for interfacing with external memory. These additional pins are used while transmitting the command, address, and data based on the control register settings and command used.

The core supports the manual slave select mode as the default mode of operation for slave select mode. This mode allows manual control of the slave select line with the data written to the slave select register, thereby allowing transfers of an arbitrary number of elements without toggling the slave select line between elements. However, before starting a new transfer, the slave select line must be toggled.

The other mode of operation related to slave select is automatic slave select mode. In this mode, the slave select line is toggled automatically after each element transfer. This mode, which is supported only in standard SPI mode, is described in more detail in SPI Protocol Slave Select Assertion Modes in Chapter 3.

The core functionality is divided into standard SPI mode and dual and quad SPI mode. The functionality for each mode differs in the way the slave memory works.

Standard SPI Mode

Standard SPI mode is selected when the **Mode** option in the Vivado IDE is set to **Standard**. The relevant parameters in this mode are:

- Mode
- Slave Device
- Enable STARTUPE2 Primitive
- Transaction Width
- No. of Slaves
- Frequency Ratio
- FIFO Depth

The properties of the core in standard SPI mode, including or excluding a FIFO, are described as:

1. The choice of inclusion of the FIFO is based on the **FIFO Depth** option which, if the FIFO is included in the design, limits the transmit and receive FIFO depth to 16 or 256. A FIFO depth of 256 should be used because this is the most suitable depth in relation to the flash memory page size.
2. The valid values for the **FIFO Depth** option in this mode are **0**, **16** or **256**.

When **FIFO Depth** is **0**, no FIFO is included in the core. Data transmission occurs through the single transmit and receive register. When **FIFO Depth** is **16** or **256**, the transmit or receive FIFO is included in the design with a depth of 16 or 256 elements. The width of the transmit and receive FIFO is configured with the **Transaction Width** option.

The AXI Quad SPI core supports continuous transfer mode. When configured as master, the transfer continues until the data is available in the transmit register/FIFO. This capability is provided in both manual and automatic slave select modes. As an example, during the page read command, the command, address, and number of data beats in the DTR must be set equal to the same number of data bytes intended to be read by the SPI memory.

When the core is configured as a slave, if the slave select line (SPISEL) inadvertently goes High (inactive state) during the data element transfer, the current transfer is aborted. If the slave select line goes Low, the aborted data element is transmitted again. The slave mode of the core is allowed only in the standard SPI mode.

Dual/Quad SPI Mode

Dual SPI mode is selected when **Mode** option in the Vivado IDE is set to **Dual**. The relevant parameters in this mode are:

- Mode
- Slave Device
- Enable STARTUPE2 Primitive
- Transaction Width
- No. of Slaves
- FIFO Depth

The properties associated with the FIFO are:

- The depth of the FIFO is based on the **FIFO Depth** option which has valid values of **16** or **256**
- The width of the FIFO is 8 bits because the page size of the SPI slave memories is always 8 bits

The behavior of the ports in dual mode is:

- For standard SPI mode instructions, the IO0 and IO1 pins are unidirectional (the same as the MOSI and MISO pins).
- For dual mode SPI instructions, the IO0 and IO1 pins are bidirectional — depending on the type of command and memory chosen.

The quad SPI mode is selected when the **Mode** option is set to **Quad**. The behavior of the ports in quad SPI mode is:

- For standard mode SPI instructions, the IO0 and IO1 pins are unidirectional and function the same as in standard SPI mode.
- For dual mode SPI instructions, the IO0 and IO1 pins are unidirectional or bidirectional depending on the type of instruction and memory selected by setting the control register bits. The IO2 and IO3 bits are 3-state.
- For quad mode SPI instructions, the IO0, IO1, IO2, and IO3 pins are unidirectional or bidirectional depending on the type of memory used while transmitting the command, address, and data.

When the **Mode** option is **Dual** or **Quad**, the core is forced to operate in dual or quad SPI mode, respectively, while the core continues to support the standard SPI commands and interface. The internal command logic guides the core I/O behavior depending on the command loaded in the DTR FIFO (SPI DTR). The **Mode** option settings also determine the I/O pin availability.

Common Information for Both SPI Modes

The core permits additional slaves to be added with automatic generation of the required decoding logic for individual slave select outputs by the master. Additional masters can also be added. However, detection of all possible conflicts is not implemented with this interface standard. To eliminate conflicts, the system software is required to arbitrate bus control.

The core can communicate with both off-chip and on-chip masters and slaves. The number of slaves is limited to 32 by the size of the slave select register. However, the number of slaves and masters affects the achievable performance in terms of frequency and resource utilization. All of the SPI core and interrupt registers are 32 bits wide. The core supports only 32-bit access to all SPI and interrupt register modules.

AXI4 Memory Mapped Interface

The AXI4 memory-mapped interface is included when the **Enable Performance Mode** option is selected. In this mode, the core can be operated in enhanced mode (**Enable XIP Mode** is not selected) or XIP mode (**Enable XIP Mode** is selected). In performance mode, the AXI4 memory mapped interface is used for burst transactions at the DTR and DRR locations.

Enhanced Mode - Non-XIP Mode

In this mode, the AXI4-Lite interface for the core is replaced with the AXI4 interface. This mode also supports standard, dual and quad modes depending on the **Mode** option setting. The target slave memory can be chosen by setting the **Slave Device** option to **Mixed**, **Winbond** or **Numonyx**. All of the registers are mapped to the same offset as with the AXI4-Lite interface. The AXI4 interface is allowed to do burst transactions at the data transmit register and data receive register only. All other registers should be single access only. This should be noted while designing the application for the core.

The DTR and DRR FIFOs are configurable to 16 or 256 beat depth. The core supports the same functionality as the AXI4-Lite interface. The added advantage for this mode is burst capability at the DTR and DRR locations, reducing the overhead of reading and writing data to and from the core at the AXI4 interface side.

XIP Mode

In XIP mode, the core has an AXI4-Lite as well as an AXI4 memory-mapped interface (see Figure 1-2). The AXI4-Lite interface is chosen for accessing the configuration register and the status register. The AXI4 interface is used only for reading. The AXI4 interface supports only the read channel. No write transactions are allowed. The AXI4-Lite interface can access the configuration register to change the clock polarity (CPOL) or clock phase (CPHA) configuration.

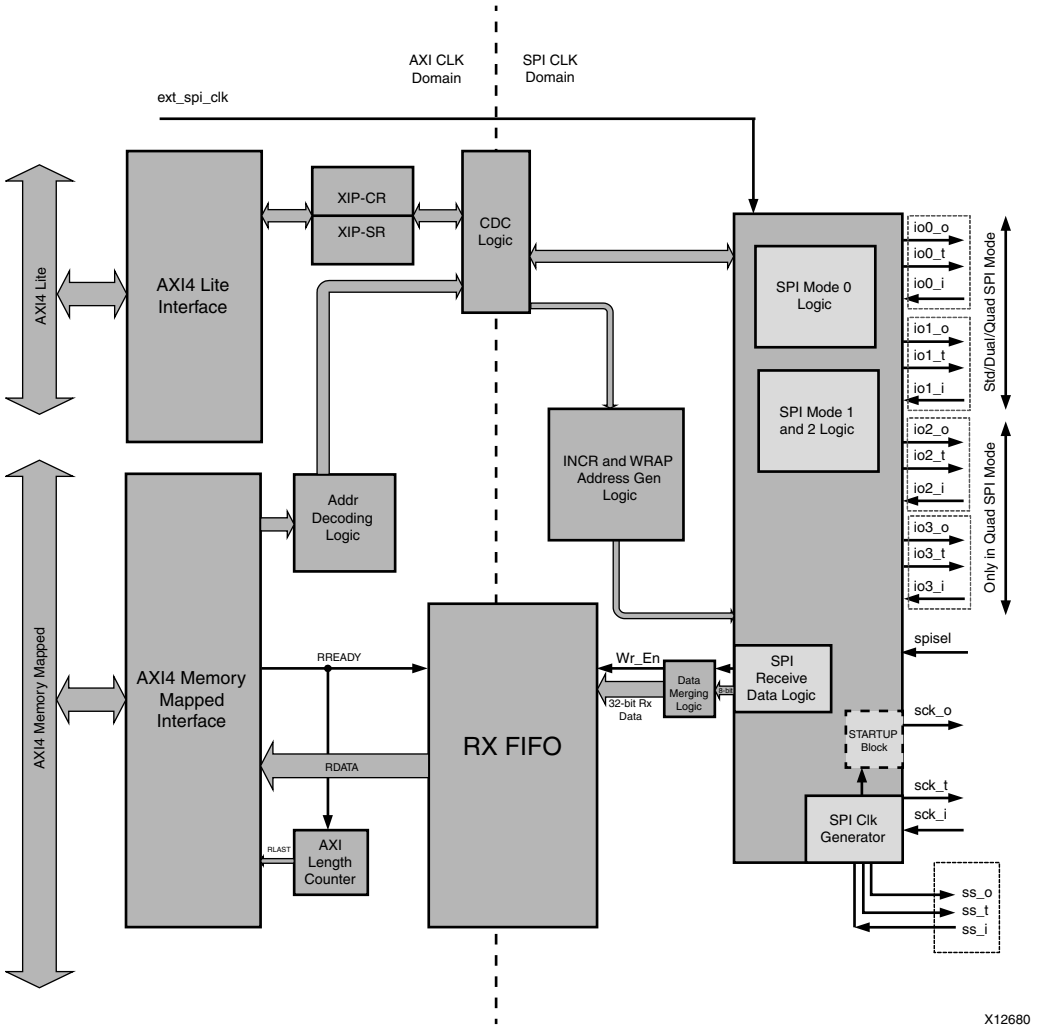

Figure 1-2: **Block Diagram of AXI Quad SPI in XIP Mode**

This mode is suitable for boot operation. In this mode, the core supports INCR and WRAP read transactions only.

In this mode, the core considers the SPI flash memory as read-only memory. The three read commands are provided with the configuration mode which is used while reading the SPI flash. The core functionality is verified by assigning the same frequency to both the AXI4-Lite and AXI4 memory mapped interface. The three main read commands which are built into the core are fast read (`0x0Bh`), DIOFR (`0xBBh`) and QIOFR (`0xEBh`).

Core Internal Submodules

The AXI Quad SPI core submodules are described in these sections:

Enable Performance Mode Not Selected

In this mode, the core is backward-compatible with all earlier versions of the AXI Quad SPI core. The core includes these submodules:

AXI4-Lite Interface Module

The AXI4-Lite interface module provides the interface to the AXI4-Lite protocol and IPIC. The read and write transactions at the AXI4-Lite interface are translated into equivalent IP interconnect (IPIC) transactions. This is the default combination for the core.

SPI Register Module

The SPI register module includes all the memory-mapped registers shown in Figure 1-1. This module connects to the AXI4-Lite interface. and consists of status register, control register, N-bit slave select register (N ≤ 32), and a pair of transmit and receive registers.

Interrupt Controller Register Set Module

The interrupt controller register set module consists of the interrupt-related registers:

- Device global interrupt enable register (DGIER)
- IP interrupt enable register (IPIER)
- IP interrupt status register (IPISR).

SPI Module

The SPI module consists of a shift register, a parameterized baud rate generator (BRG), and a control unit. It provides the SPI interface, including the control logic and initialization logic. In standard SPI mode, this module is the center of the SPI operation.

Optional FIFOs

When enabled by the parameter **FIFO Depth**, the transmit FIFO and receive FIFO are implemented on both the transmit and receive paths. The width of the transmit FIFO and receive FIFO are the same and depend on the **Transaction Width** parameter. When the FIFOs are enabled, their depth is variable at 0, 16 or 256 in standard SPI mode. In dual and quad SPI modes, the FIFO depth is 16 or 256 locations (bytes).

Start-Up Module

STARTUPE2 is a primitive in the Xilinx FPGA. This primitive can be used after the FPGA configuration in the design. For more understanding on the use of this primitive, read the targeted FPGA user guide. This primitive can be included in the design by selecting the **Enable STARTUPE2 Primitive** parameter. This primitive has a dedicated clock pin which can be used to provide the SPI clock to the slave memory.

QSPI Control Logic Module

This module is responsible for generation of control signals which are used in dual or quad SPI modes. This module contains logic for a shift register, SPI clock generator, and state machines for various memory configurations.

Feature Summary

- Legacy mode — AXI4-Lite interface based design
  - AXI interface
    - Supports AXI4-Lite interface — legacy mode
    - All registers in the core should be accessed as 32-bit access and only through single length AXI4-Lite transaction
  - Configurable SPI modes
    - Supports standard, dual and quad SPI mode of operation
    - Standard mode supports:
      - Master and slave SPI mode
      - MSB/LSB first transactions
      - Local loopback capability for testing
      - Multiple master and multiple slave environment
      - Optional 0 or 16 or 256 element deep (an element is a byte, a half-word or a word) transmit and receive FIFOs
    - Dual/quad SPI mode supports:
      - Master mode only
      - MSB transfer only
      - SPI transfer length of 8-bit only
      - Multiple master and multiple slave environment

    - Optional 16 or 256 deep transmit and receive FIFO.
    - Optional support of 6 pin SPI interface mode (In Quad mode only)

- AXI4 memory mapped interface mode
  - AXI4 interface
    - Supports AXI4 memory mapped interface
  - Configurable SPI interface supports:
    - Standard, dual and quad mode of SPI configuration
    - Master mode only
    - 16 or 256 element deep transmit and receive FIFO
    - MSB-only transfer of 8-bit length at SPI
    - Multiple SPI slaves - configurable up to 32
    - Configurable XIP (execute in place) and non-XIP modes
  - Enhanced mode — Non-XIP mode (burst mode access) — AXI4 memory mapped design:
    - SPI read and write commands provided by master
    - Only fixed-burst transfer at DTR and DRR FIFO locations.
    - Only one read or one write transaction is acceptable at a time from AXI4 memory mapped interface
    - All registers in the core should be accessed as 32-bit access and only through single length AXI4 transaction
    - WRAP transactions not supported
  - Read only XIP mode — AXI4-Lite + AXI4 memory mapped interface based design:
    - AXI4-Lite mode used for setting the configuration register and reading the status/debug register
    - Read commands only at SPI interface
    - INCR and WRAP read transactions at memory mapped address
    - 64 beat deep fixed internal FIFO with 32-bit data width
    - FIXED transactions unsupported

Table 1-1 defines the parameters used to configure the core.

Table 1-1: **Core Operation Mode and Design Parameter Values**

Core Operation	Parameters							
	Enable Performance Mode	Enable XIP Mode	FIFO Depth	Mode	Frequency Ratio	No. of Slaves	Slave Device	Enable STARTUPE2 Primitive
Legacy mode[3]			0, 16, 256	Standard	2, 4, 8, Nx16 for N = 1, 2, 3,...,128[1]	1 - 32	(any)	(either)
			16, 256	Dual	2[2]	1 - 32	(any)	(either)
			16, 256	Quad	2[2]	1 - 32	(any)	(either)
Enhanced mode[3]	x		0, 16, 256	Standard	2, 4, 8, Nx16 for N = 1, 2, 3,...,128[1]	1 - 32	(any)	(either)
	x		16, 256	Dual	2[2]	1 - 32	(any)	(either)
	x		16, 256	Quad	2[2]	1 - 32	(any)	(either)
XIP mode[3]	x	x	64	Standard	2[2]	1[4]	(any)	(either)
	x	x	64	Dual	2[2]	1[4]	(any)	(either)
	x	x	64	Quad	2[2]	1[4]	(any)	(either)

Notes:

1. In this mode the `ext_spi_clk` can be the same as the `axi_aclk` or `axi4_aclk`, but it should not be less than the AXI CLK or AXI4 ACLK. This mode is specifically for slow operating devices which work on SPI protocol. Examples of these devices are EEPROMs and SPI interface based DACs.
2. In this mode, set `ext_spi_clk` to double the intended SPI clock. The Frequency Ratio parameter divides this clock by 2 to generate the SPI clock. The maximum clock on the `ext_spi_clk` port should be 100 MHz.
3. Most of the SPI flash memory commands operate at a higher SPI clock rate with the exception of some commands which operate at a lesser frequency. It is your responsibility to configure the core with the correct `ext_spi_clk` and Frequency Ratio parameter and use the appropriate commands. If slower operating commands are executed on higher SPI clock ratios, the core does not generate an error and passes these commands to the external flash, but at the same time, the operation of the flash is not guaranteed.
4. In XIP mode, No. of Slaves is always equal to 1 and this parameter cannot be updated.

Unsupported Features

These features relate to the AXI4 memory-mapped interface and are not supported by the core.

- INCR of length more than 1 and WRAP bursts in enhanced mode
- FIXED burst in XIP mode
- Narrow bursts in enhanced mode (only the last 8 bits from the 32 burst bits are valid in enhanced mode)
- Write channel and transactions in XIP mode
- Atomic, locked and cache transactions
- Debug/secure and user signals
- Out-of-order transactions
- Region signals
- QOS signals
- Holes in byte strobes
- Barrier transactions
- Write interleaving
- User signals
- AXI TrustZone and low power state
- Simultaneous read and write transactions in enhanced mode
- Un-aligned address when the core is configured in read-only XIP mode
- Byte access in XIP mode

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Standards

The AXI Quad SPI core connects the AXI4 and AXI4-Lite interfaces to SPI slave devices that support the standard, dual or quad SPI protocol instruction set. The core provides a serial interface to SPI slave devices such as SPI serial flash from Winbond and Numonyx. The dual/quad SPI is an enhancement to the standard SPI protocol (described in the Motorola M68HC11 data sheet). See Specification Exceptions for differences between the core protocol and the Motorola M68HC11-Rev. 4.0 reference manual.

Performance

The performance characterization of this core was compiled using the margin system methodology. The details of the margin system characterization methodology are described in the "Vivado IP Optimization (Fmax Characterization)" appendix of the *Vivado Design Suite User Guide: Designing With IP (UG896)* [Ref 1].

Note: Performance for Zynq®-7000 devices is similar to 7 series devices.

The performance characterization was compiled for these families:

- Virtex®-7
- Kintex®-7
- Artix®-7

Maximum Frequencies

The maximum frequencies for the AXI Quad SPI core are shown in Table 2-1.

Table 2-1: **AXI Quad SPI Maximum Frequencies**

Family	Speed Grade	F_{MAX} (MHz)
Virtex-7	-1	180
	-2	200
	-3	200
Kintex-7	-1	180
	-2	200
	-3	200
Artix-7	-1	180
	-2	200
	-3	200

Performance Comparison

The commands in Table 2-2 are used to provide a comparison between the AXI Quad SPI v1.00 core and the current AXI Quad SPI core. These commands are randomly chosen and the time taken for completion is calculated using simulation waveforms for writing 256 bytes. The time specified here is taken across these events:

1. Assert slave select.
2. Fill the DTR of the master.
3. Enable the transaction.
4. Wait until the transaction completes.
5. Deassert the slave select line.

In reality this time might be further reduced based on how the DTR is filled.

Table 2-2: **Performance Comparison**

Command	From	To	AXI Quad v1.00	Current AXI Quad SPI
DIOFR	Slave Select assertion	Slave Select de-assertion	58080 ns	21810 ns
QOFR			22400 ns	6580 ns

Resource Utilization

Because the AXI Quad SPI core is used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When the core is combined with other designs in the system, the utilization of FPGA resources and core timing varies from the results reported here.

The AXI Quad SPI core resource utilization for various parameter combinations measured with an Artix-7 FPGA as the target device are detailed in Table 2-3. The utilization figures should be considered as reference only.

Note: Performance for Zynq-7000 and UltraScale™ devices is similar to 7 series devices.

Table 2-3: **Resource Utilization for an Artix-7 FPGA (XC7A100T-3)**

Parameter Values (other parameters at default values)								Device Resources			
Enable Performance Mode	Enable XIP Mode	Mode	Slave Device	FIFO Depth	Frequency Ratio	No. of Slaves	Transaction Width	LUTs	Flip-Flops	LUT-Flip-Flop Pairs	Slices
		Standard	Mixed	16	2	2	8	301	483	469	167
		Dual	Winbond	256	2	2	8	498	612	667	234
		Quad	Winbond	256	2	2	8	524	619	701	238
		Standard	Mixed	256	2	2	8	481	619	651	210
		Dual	Numonyx	256	2	2	8	535	614	706	243
		Quad	Numonyx	16	2	2	8	350	493	509	177
		Dual	Numonyx	16	2	2	8	320	486	473	164
		Dual	Winbond	16	2	2	8	326	484	468	156
		Quad	Numonyx	256	2	2	8	531	621	686	226
		Quad	Mixed	16	2	2	8	324	489	493	175
		Quad	Winbond	16	2	2	8	346	491	514	179
x	x	Standard	Numonyx	16	2	1	8	480	570	718	286
x	x	Dual	Numonyx	16	2	1	8	492	574	731	275
x	x	Dual	Winbond	16	2	1	8	508	579	735	276
x	x	Quad	Winbond	16	2	1	8	509	581	741	273

Table 2-3: **Resource Utilization for an Artix-7 FPGA (XC7A100T-3) *(Cont'd)***

Parameter Values (other parameters at default values)								Device Resources			
Enable Performance Mode	Enable XIP Mode	Mode	Slave Device	FIFO Depth	Frequency Ratio	No. of Slaves	Transaction Width	LUTs	Flip-Flops	LUT-Flip-Flop Pairs	Slices
x	x	Quad	Numonyx	16	2	1	8	494	578	712	259
x	x	Standard	Winbond	16	2	1	8	481	567	665	252

The AXI Quad SPI core resource utilization for various parameter combinations in legacy mode measured with a Virtex-7 FPGA as the target device are detailed in Table 2-4. The utilization figures should be considered as reference only.

Table 2-4: **Resource Utilization for a Virtex-7 FPGA (xc7v285tffg784-3)**

Parameter Values (other parameters at default values)								Device Resources			
Enable Performance Mode	Enable XIP Mode	Mode	Slave Device	FIFO Depth	Frequency Ratio	No. of Slaves	Transaction Width	LUTs	Flip-Flops	LUT-Flip-Flop Pairs	Slices
x	x	Standard	Numonyx	16	2	1	8	480	570	700	258
x	x	Quad	Numonyx	16	2	1	8	493	578	709	269
		Standard	Mixed	16	2	2	8	300	483	426	131
x	x	Quad	Winbond	16	2	1	8	509	581	704	259
		Quad	Mixed	16	2	2	8	324	489	482	174
		Dual	Winbond	16	2	2	8	327	484	477	164
		Quad	Winbond	16	2	2	8	345	491	506	178
		Quad	Numonyx	16	2	2	8	349	493	491	166
		Quad	Numonyx	256	2	2	8	532	621	702	245
		Quad	Winbond	256	2	2	8	524	619	702	239
		Dual	Winbond	256	2	2	8	497	612	652	205
x	x	Standard	Winbond	16	2	1	8	481	567	714	271

Table 2-4: **Resource Utilization for a Virtex-7 FPGA (xc7v285tffg784-3) *(Cont'd)***

Parameter Values (other parameters at default values)								Device Resources			
Enable Performance Mode	Enable XIP Mode	Mode	Slave Device	FIFO Depth	Frequency Ratio	No. of Slaves	Transaction Width	LUTs	Flip-Flops	LUT-Flip-Flop Pairs	Slices
		Dual	Numonyx	16	2	2	8	320	486	458	154
x	x	Dual	Numonyx	16	2	1	8	492	574	682	233
		Standard	Mixed	256	2	2	8	481	619	656	208
		Dual	Numonyx	256	2	2	8	536	614	694	234
x	x	Dual	Winbond	16	2	1	8	507	579	706	255

The AXI Quad SPI core resource utilization for various parameter combinations in legacy mode measured with a Kintex-7 FPGA as the target device are detailed in Table 2-5. These utilization figures should be considered as reference only.

Table 2-5: **Resource Utilization for a Kintex-7 FPGA (xc7k325tffg900-3)**

Parameter Values (other parameters at default values)								Device Resources			
Enable Performance Mode	Enable XIP Mode	Mode	Slave Device	FIFO Depth	Frequency Ratio	No. of Slaves	Transaction Width	LUTs	Flip-Flops	LUT-Flip-Flop Pairs	Slices
x	x	Dual	Winbond	16	2	1	8	508	579	727	269
		Quad	Winbond	256	2	2	8	525	619	685	229
x	x	Dual	Numonyx	16	2	1	8	492	574	694	244
		Dual	Numonyx	16	2	2	8	320	486	472	153
x	x	Standard	Numonyx	16	2	1	8	480	570	710	264
		Dual	Winbond	16	2	2	8	327	484	456	148
x	x	Quad	Numonyx	16	2	1	8	494	578	689	243
		Dual	Winbond	256	2	2	8	498	612	648	205
		Standard	Mixed	16	2	2	8	299	483	433	145

Table 2-5: **Resource Utilization for a Kintex-7 FPGA (xc7k325tffg900-3) *(Cont'd)***

Parameter Values (other parameters at default values)								Device Resources			
Enable Performance Mode	Enable XIP Mode	Mode	Slave Device	FIFO Depth	Frequency Ratio	No. of Slaves	Transaction Width	LUTs	Flip-Flops	LUT-Flip-Flop Pairs	Slices
x		Quad	Winbond	16	2	1	8	509	581	710	251
		Quad	Numonyx	16	2	2	8	350	493	502	174
		Quad	Numonyx	256	2	2	8	531	621	700	233
x	x	Standard	Winbond	16	2	1	8	481	567	677	244
		Standard	Mixed	256	2	2	8	481	619	665	218
		Quad	Mixed	16	2	2	8	324	489	456	156
		Dual	Numonyx	256	2	2	8	535	614	687	233
		Quad	Winbond	16	2	2	8	346	491	500	168

Port Descriptions

Legacy Mode

The I/O signals when **Enable Performance Mode** is not selected are described in Table 2-6.

Table 2-6: **I/O Signal for Legacy Mode (AXI4-Lite Interface)**

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Global System Signals					
P1	s_axi_aclk	AXI	I	-	AXI Clock.
P2	s_axi_aresetn	AXI	I	-	AXI Reset. Active-Low.
P3	ext_spi_clk	-	I	-	This clock is used for SPI interface. This clock should be double of the maximum SPI frequency intended at the SPI interface.
AXI4-Lite Write Address Channel Signals					
P4	s_axi_awaddr[31:0]	AXI	I	-	AXI Write address. The write address bus gives the address of the write transaction.

Table 2-6: **I/O Signal for Legacy Mode (AXI4-Lite Interface) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P5	s_axi_awvalid	AXI	I	-	Write address valid. Indicates that a valid write address and control information are available.
P6	s_axi_awready	AXI	O	0	Write address ready. Indicates that the slave is ready to accept an address and associated control signals.
AXI4-Lite Write Channel Signals					
P7	s_axi_wdata[31:0]	AXI	I	-	Write data.
P8	s_axi_wstb[3:0]	AXI	I	-	Write strobes. Indicates which byte lanes to update in memory.
P9	s_axi_wvalid	AXI	I	-	Write valid. Indicates that valid write data and strobes are available.
P10	s_axi_wready	AXI	O	0	Write ready. Indicates that the slave can accept the write data.
AXI4-Lite Write Response Channel Signals					
P11	s_axi_bresp[1:0]	AXI	O	0	Write response. Indicates the status of the write transaction. 00 - OKAY (normal response). 10 - SLVERR (error response). 11 - DECERR (not issued by core).
P12	s_axi_bvalid	AXI	O	0	Write response valid. Indicates that a valid write response is available.
P13	s_axi_bready	AXI	I	-	Response ready. Indicates that the master can accept the response information.
AXI4-Lite Read Address Channel Signals					
P14	s_axi_araddr[31:0]	AXI	I	-	Read address. The read address bus gives the address of a read transaction.
P15	s_axi_arvalid	AXI	I	-	Read address valid. This signal indicates, when High, that the read address and control information is valid and remains stable until the address acknowledgement signal, `s_axi_arready`, is High.
P16	s_axi_arready	AXI	O	1	Read address ready. Indicates that the slave is ready to accept an address and associated control signals.
AXI4-Lite Read Data Channel Signals					
P17	s_axi_rdata[31:0]	AXI	O	0	Read data.

Table 2-6: **I/O Signal for Legacy Mode (AXI4-Lite Interface) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P18	s_axi_rresp[1:0]	AXI	O	0	Read response. Indicates the status of the read transfer. 00 - OKAY (normal response). 10 - SLVERR (error condition). 11 - DECERR (not issued by core).
P19	s_axi_rvalid	AXI	O	0	Read valid. Indicates that the required read data is available and the read transfer can complete.
P20	s_axi_rready	AXI	I	-	Read ready. Indicates that the master can accept the read data and response information.
SPI Slave Device Interface Signals					
Global System Signals From the Core					
P21	ip2intc_irpt	SPI	O	0	Interrupt control signal from SPI.
SPI Interface Signals					
P22	sck_i	SPI	I	-	SPI bus clock input. This signal is available only when the core is configured in Standard SPI slave mode.
P23	sck_o	SPI	O	0	SPI bus clock output.
P24	sck_t	SPI	O	1	3-state enable for SPI bus clock. Active-Low.
P25	ss_i[(No. of Slaves – 1):0]	SPI	I	-	Input one-hot encoded. This signal is a dummy signal and is not used in the design as a chip select input.
P26	ss_o[(No. of Slaves – 1):0]	SPI	O	1	Output one-hot encoded, active-Low slave select vector of length n.
P27	ss_t	SPI	O	1	3-state enable for slave select. Active-Low.
Standard (and Dual) SPI Mode Signals					
P28	io0_i	SPI	I	-	Behaves similar to master output slave input (MOSI) input pin.
P29	io0_o	SPI	O	-	Behaves similar to master output slave input (MOSI) output pin. This is available only in standard SPI mode. In dual SPI mode, this signal acts as a bidirectional signal based on certain instructions.
P30	io0_t	SPI	O	1	3-state enable master output slave input. Active-Low.

Table 2-6: **I/O Signal for Legacy Mode (AXI4-Lite Interface) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P31	io1_i	SPI	I	-	Behaves similar to master input slave output (MISO) input. This signal can also be considered as IO1_I port in dual or quad SPI mode.
P32	io1_o	SPI	O	-	Behaves similar to master input slave output (MISO) output. This is available only in standard SPI mode. In dual SPI mode, this signal acts as a bidirectional signal based on certain instructions.
P33	io1_t	SPI	O	1	3-state enable master input slave output. Active-Low.
P34	spisel[(1)]	SPI	I	1	Local SPI slave select active-Low input. This is input signal when the core is configured in standard SPI slave mode. Must be set to 1 (along with the master bit in the SPICR) in master mode.
Quad Mode SPI Signals [(2)]					
P35	io2_i	SPI	I	-	IO2 input based on commands used. This signal is available only in quad SPI mode.
P36	io2_o	SPI	O	-	IO2 output based on commands used. This signal is available only in quad SPI mode.
P37	io2_t	SPI	O	1	3-state enable IO2. This signal is available only in quad SPI mode. Active-Low.
P38	io3_i	SPI	I	-	IO3 input based on commands used. This signal is available only in quad SPI mode.
P39	io3_o	SPI	O	-	IO3 output based on commands used. This signal is available only in quad SPI mode.
P40	io3_t	SPI	O	1	3-state enable IO3. This signal is available only in quad SPI mode. Active-Low.

Notes:

1. The `spisel` signal is used as a slave select line when AXI Quad SPI is configured as a slave in Standard SPI mode.
2. These signals are applicable only when the core is configured in Quad SPI mode.

Enhanced Non-XIP Mode

In this mode, the AXI4-Lite interface is replaced with the AXI4 memory-mapped interface and the I/O list is as described in Table 2-7.

Table 2-7: **I/O Signal for Enhanced Mode (AXI4 Memory Mapped)**

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Global System Signals					
P1	s_axi4_aclk	AXI	I	-	AXI4 Clock.
P2	s_axi4_aresetn	AXI	I	-	AXI4 Reset. Active-Low.
P3	ext_spi_clk	-	I	-	This clock is used for SPI interface. This clock should be double of the maximum SPI frequency intended at the SPI interface.
AXI4 Memory Mapped Write Address Channel Signals					
P4	s_axi4_awid[(auto update)]	AXI	I	-	Write address ID: This signal is the identification tag for the write address group of signals.
P5	s_axi4_awaddr[31:0]	AXI	I	-	AXI4 Write address: The write address bus gives the address of the first transfer in a write burst transaction.
P6	s_axi4_awlen[7:0]	AXI	I	-	Burst length: This signal gives the exact number of transfers in a burst. 00000000 - 11111111 indicates burst length 1 - 256.
P7	s_axi4_awsize[2:0]	AXI	I	-	Burst size: Indicates the size of each transfer in the burst. 000 - 1 byte. 001 - 2 byte (half word). 010 - 4 byte (word). 011 - 8 byte (double word). others - NA.
P8	s_axi4_awburst[1:0]	AXI	I	-	Burst type: This signal coupled with the size information, details how the address for each transfer within the burst is calculated. 00 - FIXED. 01 - INCR. 10 - WRAP. 11 - Reserved.
P9	s_axi4_awlock(1)	AXI	I	-	Lock type:(1). This signal provides additional information about the atomic characteristics of the transfer. This signal is not supported in the design.

Table 2-7: **I/O Signal for Enhanced Mode (AXI4 Memory Mapped) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P10	s_axi4_awcache[3:0](1)	AXI	I	-	Cache type:(1). Indicates the bufferable, cacheable, write-through, write-back and allocate attributes of the transaction. Bit-0: Bufferable (B). Bit-1: Cacheable (C). Bit-2: Read Allocate (RA). Bit-3: Write Allocate (WA). This signal is not supported in the design.
P11	s_axi4_awprot[2:0](1)	AXI	I	-	Protection type:(1). Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. Bit-0: 0=Normal access, 1=Privileged access. Bit-1: 0=Secure access, 1=Non-secure access. Bit- 2: 0=Data access; 1=Instruction access. This signal is not supported in the design.
P12	s_axi4_awvalid	AXI	I	-	Write address valid: Indicates that valid write address and control information are available.
P13	s_axi4_awready	AXI	O	0	Write address ready: Indicates that the slave is ready to accept an address and associated control signals.
AXI4 Memory Mapped Write Channel Signals					
P14	s_axi4_wdata[31:0]	AXI	I	-	Write data bus.
P15	s_axi4_wstb[3:0]	AXI	I	-	Write strobes: Indicates which byte lanes in `s_axi_wdata` are/is valid.
P16	s_axi4_wlast	AXI	I	-	Write last: Indicates the last transfer in a write burst.
P17	s_axi4_wvalid	AXI	I	-	Write valid: Indicates that valid write data and strobes are available.
P18	s_axi4_wready	AXI	O	0	Write ready: Indicates that the slave can accept the write data.
AXI4 Memory Mapped Write Response Channel Signals					
P19	s_axi4_bid[(auto update)]	AXI	O	0	Write response ID: This signal is the identification tag of the write response. The BID value must match the AWID value of the write transaction to which the slave is responding.

Table 2-7: **I/O Signal for Enhanced Mode (AXI4 Memory Mapped) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P20	s_axi4_bresp[1:0]	AXI	O	0	Write response: Indicates the status of the write transaction. 00 - OKAY. 01 - EXOKAY - NA. 10 - SLVERR. 11 - DECERR - NA.
P21	s_axi4_bvalid	AXI	O	0	Write response valid: Indicates that a valid write response is available.
P22	s_axi4_bready	AXI	I	-	Response ready: Indicates that the master can accept the response information.
AXI4 Memory Mapped Read Address Channel Signals					
P23	s_axi4_arid[(auto update)]	AXI	I	-	Read address ID: This signal is the identification tag for the read address group of signals.
P24	s_axi4_araddr[31:0]	AXI	I	-	Read address: The read address bus gives the initial address of a read burst transaction.
P25	s_axi4_arlen[7:0]	AXI	I	-	Burst length: This signal gives the exact number of transfers in a burst. 00000000 - 11111111 indicates burst length 1 - 256.
P26	s_axi4_arsize[2:0]	AXI	I	-	Burst size: Indicates the size of each transfer in the burst. 000 - 1 byte. 001 - 2 byte (Half word). 010 - 4 byte (word). 011 - 8 byte (double word). others - NA.
P27	s_axi4_arburst[1:0]	AXI	I	-	Burst type: The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. 00 - FIXED. 01 - INCR. 10 - WRAP. 11 - Reserved. Only fixed burst is supported.
P28	s_axi4_arlock(1)	AXI	I	-	Lock type:(1). This signal provides additional information about the atomic characteristics of the transfer. This signal is not supported in the design.

Table 2-7: **I/O Signal for Enhanced Mode (AXI4 Memory Mapped) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P29	s_axi4_arcache[3:0](1)	AXI	I	-	Cache type:(1). This signal provides additional information about the cacheable characteristics of the transfer. Bit-0: Bufferable (B). Bit-1: Cacheable (C). Bit-2: Read Allocate (RA). Bit-3: Write Allocate (WA). This signal is not supported.
P30	s_axi4_arprot[2:0](1)	AXI	I	-	Protection type:(1). This signal provides protection unit information for the transaction.
P31	s_axi4_arvalid	AXI	O	0	Read address valid: Indicates, when High, that the read address and control information is valid and remains stable until the address acknowledgement signal, `s_axi_arready`, is high.
P32	s_axi4_arready	AXI	I	-	Read address ready: Indicates that the slave is ready to accept an address and associated control signals.
AXI4 Memory Mapped Read Data Channel Signals					
P33	s_axi4_rid[(auto update)]	AXI	O	0	Read ID tag: This signal is the ID tag of the read data group of signals. The `s_axi_rid` value is generated by the slave and must match the `arid` value of the read transaction to which it is responding.
P34	s_axi4_rdata[31:0]	AXI	O	0	Read data bus.
P35	s_axi4_rresp[1:0]	AXI	O	0	Read response: Indicates the status of the read transfer. 00 - OKAY. 01 - EXOKAY - NA. 10 - SLVERR. 11 - DECERR - NA.
P36	s_axi4_rlast	AXI	O	0	Read last: Indicates the last transfer in a read burst.
P37	s_axi4_rvalid	AXI	O	0	Read valid: Indicates that the required read data is available and the read transfer can complete.
P38	s_axi4_rready	AXI	I	-	Read ready: Indicates that the master can accept the read data and response information.

Table 2-7: **I/O Signal for Enhanced Mode (AXI4 Memory Mapped) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
SPI Slave Device Interface Signals					
Global System Signals From the Core					
P39	ip2intc_irpt	SPI	O	0	Interrupt control signal from SPI.
SPI Interface Signals					
P40	sck_i	SPI	I	-	SPI bus clock input. This signal is available only when the core is configured in standard SPI slave mode.
P41	sck_o	SPI	O	0	SPI bus clock output.
P42	sck_t	SPI	O	1	3-state enable for SPI bus clock. Active-Low.
P43	ss_i[(No. of Slaves – 1):0]	SPI	I	-	Input one-hot encoded. This signal is a dummy signal and is not used in the design as a chip select input.
P44	ss_o[(No. of Slaves – 1):0]	SPI	O	1	Output one-hot encoded, active-Low slave select vector of length n.
P45	ss_t	SPI	O	1	3-state enable for slave select. Active-Low.
Standard (and Dual) SPI Mode Signals					
P46	io0_i	SPI	I	-	Behaves similar to master output slave input (MOSI) input.
P47	io0_o	SPI	O	-	Behaves similar to master output slave input (MOSI) output pin. This is available only in standard SPI mode. In dual SPI mode, this signal acts as a bidirectional signal based on certain instructions.
P48	io1_i	SPI	I	-	Behaves similar to master input slave output (MISO) input. This signal can also be considered as IO1_I port in dual or quad SPI modes.
P49	io1_o	SPI	O	-	Behaves similar to master input slave output (MISO) output. This is available only in standard SPI mode. In dual SPI mode, this signal acts as a bidirectional signal based on certain instructions.
P50	io1_t	SPI	O	1	3-state enable master input slave output. Active-Low.
P51	spisel	SPI	I	1	Local SPI slave select active-Low input. This is input signal when the core is configured in standard SPI slave mode. Must be set to 1 in master mode.

Table 2-7: **I/O Signal for Enhanced Mode (AXI4 Memory Mapped) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
Quad Mode SPI Signals(2)					
P52	io2_i	SPI	I	-	IO2 input based on commands used. This signal is available only in quad SPI mode.
P53	io2_o	SPI	O	-	IO2 output based on commands used. This signal is available only in quad SPI mode.
P54	io2_t	SPI	O	1	3-state enable IO2. This signal is available only in quad SPI mode. Active-Low.
P55	io3_i	SPI	I	-	IO3 input based on commands used. This signal is available only in quad SPI mode.
P56	io3_o	SPI	O	-	IO3 output based on commands used. This signal is available only in quad SPI mode.
P54	io3_t	SPI	O	1	3-state enable IO3. This signal is available only in quad SPI mode. Active-Low.

Notes:

1. These ports and associated functionality are not supported in this core.
2. These ports are available only when the core is configured in Quad mode.

Enhanced XIP Mode

When **Enable XIP Mode** is selected, the design has the I/O ports described in Table 2-8.

Table 2-8: **I/O Signal Description in XIP Mode**

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Global System Signals					
P1	s_axi_aclk	AXI	I	-	AXI clock.
P2	s_axi_aresetn	AXI	I	-	AXI reset. Active-Low.
P3	s_axi4_aclk	AXI4	I	-	AXI4 clock.
P4	s_axi4_aresetn	AXI4	I	-	AXI4 reset. Active-Low.
P5	ext_spi_clk	-	I	-	This clock is used for SPI interface. This clock should be double of the maximum SPI frequency intended at the SPI interface.
AXI4-Lite Write Address Channel Signals					
P6	s_axi_awaddr[31:0]	AXI	I	-	AXI Write address. The write address bus gives the address of the write transaction.

Table 2-8: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P7	s_axi_awvalid	AXI	I	-	Write address valid. Indicates that a valid write address and control information are available.
P8	s_axi_awready	AXI	O	0	Write address ready. Indicates that the slave is ready to accept an address and associated control signals.
AXI4-Lite Write Channel Signals					
P9	s_axi4_wdata[31:0]	AXI	I	-	Write data.
P10	s_axi_wstb[3:0]	AXI	I	-	Write strobes. Indicates which byte lanes to update in memory.
P11	s_axi_wvalid	AXI	I	-	Write valid. Indicates that valid write data and strobes are available.
P12	s_axi_wready	AXI	O	0	Write ready. Indicates that the slave can accept the write data.
AXI4-Lite Write Response Channel Signals					
P13	s_axi_bresp[1:0]	AXI	O	0	Write response. Indicates the status of the write transaction. 00 - OKAY (normal response). 10 - SLVERR (error response). 11 - DECERR (not issued by core).
P14	s_axi_bvalid	AXI	O	0	Write response valid. Indicates that a valid write response is available.
P15	s_axi_bready	AXI	I	-	Response ready. Indicates that the master can accept the response information.
AXI4-Lite Read Address Channel Signals					
P16	s_axi_araddr[31:0]	AXI	I	-	Read address. The read address bus gives the address of a read transaction.
P17	s_axi_arvalid	AXI	I	-	Read address valid. This signal indicates, when High, that the read address and control information is valid and remains stable until the address acknowledgement signal, `s_axi_aready`, is High.
P18	s_axi_aready	AXI	O	1	Read address ready. Indicates that the slave is ready to accept an address and associated control signals.

Table 2-8: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
AXI4-Lite Read Data Channel Signals					
P19	s_axi_rdata[31:0]	AXI	O	0	Read data.
P20	s_axi_rresp[1:0]	AXI	O	0	Read response. Indicates the status of the read transfer. 00 - OKAY (normal response). 10 - SLVERR (error condition). 11 - DECERR (not issued by core).
P21	s_axi_rvalid	AXI	O	0	Read valid. Indicates that the required read data is available and the read transfer can complete.
P22	s_axi_rready	AXI	I	-	Read ready. Indicates that the master can accept the read data and response information.
AXI4 Memory Mapped Write Address Channel Signals - Unused Signals					
P23	s_axi4_awid[(auto update)]	AXI	I	-	Write address ID: This signal is the identification tag for the write address group of signals.
P24	s_axi4_awaddr[31:0]	AXI	I	-	AXI4 write address: The write address bus gives the address of the first transfer in a write burst transaction.
P25	s_axi4_awlen[7:0]	AXI	I	-	Burst length: This signal gives the exact number of transfers in a burst. 00000000 - 11111111indicates burst length 1 - 256.
P26	s_axi4_awsize[2:0]	AXI	I	-	Burst size: Indicates the size of each transfer in the burst. 000 - 1 byte. 001 - 2 byte (half word). 010 - 4 byte (word). 011 - 8 byte (double word). others - NA.
P27	s_axi4_awburst[1:0]	AXI	I	-	Burst type: This signal coupled with the size information, details how the address for each transfer within the burst is calculated. 00 - FIXED. 01 - INCR. 10 - WRAP. 11 - Reserved.
P28	s_axi4_awlock[1](1)	AXI	I	-	Lock type: This signal provides additional information about the atomic characteristics of the transfer. This signal is not supported in the design.

Table 2-8: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P29	s_axi4_awcache[3:0][1][(1)]	AXI	I	-	Cache type: Indicates the bufferable, cacheable, write-through, write-back and allocate attributes of the transaction. Bit-0: Bufferable (B). Bit-1: Cacheable (C). Bit-2: Read allocate (RA). Bit-3: Write allocate (WA). This signal is not supported in the design.
P30	s_axi4_awprot[2:0][1][(1)]	AXI	I	-	Protection type: Indicates the normal, privileged, or secure protection level of the transaction and whether the transaction is a data access or an instruction access. Bit-0: 0=Normal access, 1=Privileged access. Bit-1: 0=Secure access, 1=Non-secure access. Bit- 2: 0=Data access, 1=Instruction access. This signal is not supported in the design.
P31	s_axi4_awvalid	AXI	I	-	Write address valid: Indicates that valid write address and control information are available.
P32	s_axi4_awready	AXI	O	0	Write address ready: Indicates that the slave is ready to accept an address and associated control signals.
AXI4 Memory Mapped Write Channel Signals - Unused Signals					
P33	s_axi4_wdata[31:0]	AXI	I	-	Write data bus.
P34	s_axi4_wstb[3:0]	AXI	I	-	Write strobes: Indicates which byte lanes in `s_axi_wdata` are/is valid.
P35	s_axi4_wlast	AXI	I	-	Write last: Indicates the last transfer in a write burst.
P36	s_axi4_wvalid	AXI	I	-	Write valid: Indicates that valid write data and strobes are available.
P37	s_axi4_wready	AXI	O	0	Write ready: Indicates that the slave can accept the write data.
AXI4 Memory Mapped Write Response Channel Signals - Unused Signals					
P38	s_axi4_bid[(auto update)]	AXI	O	0	Write response ID: This signal is the identification tag of the write response. The BID value must match the `awid` value of the write transaction to which the slave is responding.

Table 2-8: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P39	s_axi4_bresp[1:0]	AXI	O	0	Write response: Indicates the status of the write transaction. 00 - OKAY. 01 - EXOKAY - NA. 10 - SLVERR. 11 - DECERR - NA.
P40	s_axi4_bvalid	AXI	O	0	Write response valid: Indicates that a valid write response is available.
P41	s_axi4_bready	AXI	I	-	Response ready: Indicates that the master can accept the response information.
AXI4 Memory Mapped Read Address Channel Signals					
P42	s_axi4_arid[(auto update)]	AXI	I	-	Read address ID: This signal is the identification tag for the read address group of signals.
P43	s_axi4_araddr[31:0]	AXI	I	-	Read address: The read address bus gives the initial address of a read burst transaction.
P44	s_axi4_arlen[7:0]	AXI	I	-	Burst length: This signal gives the exact number of transfers in a burst. 00000000 - 11111111indicates burst length 1 - 256.
P45	s_axi4_arsize[2:0]	AXI	I	-	Burst size: Indicates the size of each transfer in the burst. 000 - 1 byte. 001 - 2 byte (Half word). 010 - 4 byte (word). 011 - 8 byte (double word). others - NA.
P46	s_axi4_arburst[1:0]	AXI	I	-	Burst type: The burst type, coupled with the size information, details how the address for each transfer within the burst is calculated. 00 - FIXED. 01 - INCR. 10 - WRAP. 11 - Reserved.
P47	s_axi4_arlock(1)	AXI	I	-	Lock type: This signal provides additional information about the atomic characteristics of the transfer. This signal is not supported in the design.

Table 2-8: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P48	s_axi4_arcache[3:0][(1)]	AXI	I	-	Cache type: This signal provides additional information about the cacheable characteristics of the transfer. Bit-0: Bufferable (B). Bit-1: Cacheable (C). Bit-2: Read Allocate (RA). Bit-3: Write Allocate (WA). This signal is not supported in the design.
P49	s_axi4_arprot[2:0][(1)]	AXI	I	-	Protection type: This signal provides protection unit information for the transaction. This signal is not supported in the design.
P50	s_axi4_arvalid	AXI	O	0	Read address valid: This signal indicates, when High, that the read address and control information is valid and remains stable until the address acknowledgement signal, `s_axi_arready`, is High.
P51	s_axi4_arready	AXI	I	-	Read address ready: Indicates that the slave is ready to accept an address and associated control signals.
AXI4 Memory Mapped Read Data Channel Signals					
P52	s_axi4_rid[(auto update)]	AXI	O	0	Read ID tag: This signal is the ID tag of the read data group of signals. The `s_axi_rid` value is generated by the slave and must match the `arid` value of the read transaction to which it is responding.
P53	s_axi4_rdata[31:0]	AXI	O	0	Read data bus.
P54	s_axi4_rresp[1:0]	AXI	O	0	Read response: Indicates the status of the read transfer. 00 - OKAY. 01 - EXOKAY - NA. 10 - SLVERR. 11 - DECERR - NA.
P55	s_axi4_rlast	AXI	O	0	Read last: Indicates the last transfer in a read burst.
P56	s_axi4_rvalid	AXI	O	0	Read valid: Indicates that the required read data is available and the read transfer can complete.
P57	s_axi4_rready	AXI	I	-	Read ready: Indicates that the master can accept the read data and response information.

Table 2-8: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
SPI Slave Device Interface Signals					
SPI Interface Signals					
P58	sck_i	SPI	I	-	SPI bus clock input. This signal is available only when the core is configured in standard SPI slave mode.
P59	sck_o	SPI	O	0	SPI bus clock output.
P60	sck_t	SPI	O	1	3-state enable for SPI bus clock. Active-Low.
P61	ss_i[(No. of Slaves – 1):0][(2)]	SPI	I	-	Input one-hot encoded. This signal is a dummy signal and is not used in the design as chip select input.
P62	ss_o[(No. of Slaves – 1):0][(2)]	SPI	O	1	Output one-hot encoded, active-Low slave select vector of length n.
P63	ss_t	SPI	O	1	3-state enable for slave select. Active-Low.
Standard (and Dual) SPI Mode Signals					
P64	io0_i	SPI	I	-	Behaves similar to master output slave input (MOSI) input.
P65	io0_o	SPI	O	-	Behaves similar to master output slave input (MOSI) output pin. This is available only in standard SPI mode. In dual SPI mode, this signal acts as a bidirectional signal based on certain instructions.
P66	io0_t	SPI	O	1	3-state enable master output slave input. Active-Low.
P67	io1_i	SPI	I	-	Behaves similar to master input slave output (MISO) input. This signal can also be considered as IO1_I port in dual or quad SPI mode.
P68	io1_o	SPI	O	-	Behaves similar to master input slave output (MISO) output. This is available only in standard SPI mode. In dual SPI mode, this signal acts as a bidirectional signal based on certain instructions.
P69	io1_t	SPI	O	1	3-state enable master input slave output. Active-Low.

Notes:

1. These AXI4 ports and corresponding functionality are not supported in this mode.
2. In this mode, the parameter No. of Slaves is always 1 and this parameter cannot be changed.

Register Space

Legacy and Enhanced Non-XIP Mode

Table 2-9 shows the set of registers applicable whether or not **Enable Performance Mode** is selected and **Enable XIP Mode** is not selected. Some AXI Quad SPI core registers should be accessed individually. These registers are configurable and accessible through either the AXI4-Lite interface or the AXI4 memory-mapped interface (enhanced mode). All registers are accessed as 32 bit. If non-existent registers are accessed, they return an OKAY response. The reading of these registers will return 0, and write will not have any affect.

Table 2-9: **Core Registers in Legacy and Enhanced Mode**

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
Core Grouping				
Base Address + 40	SRR	Write	N/A	Software reset register
Base Address + 60	SPICR	R/W	0x180	SPI control register
Base Address + 64	SPISR	Read	0x0a5	SPI status register
Base Address + 68	SPI DTR	Write	0x0	SPI data transmit register A single register or a FIFO
Base Address + 6C	SPI DRR	Read	N/A(1)	SPI data receive register A single register or a FIFO
Base Address + 70	SPISSR	R/W	No slave is selected 0xffff	SPI Slave select register
Base Address + 74	SPI Transmit FIFO Occupancy Register(2)	Read	0x0	Transmit FIFO occupancy register
Base Address + 78	SPI Receive FIFO Occupancy Register(2)	Read	0x0	Receive FIFO occupancy register
Interrupt Controller Grouping				
Base Address + 1C	DGIER	R/W	0x0	Device global interrupt enable register
Base Address + 20	IPISR	R/TOW(3)	0x0	IP interrupt status register

Table 2-9: **Core Registers in Legacy and Enhanced Mode *(Cont'd)***

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
Base Address + 28	IPIER	R/W	0x0	IP interrupt enable register

Notes:

1. The power-on-reset data in the SPI DRR is unknown or all zeroes. This register should not be considered for power-on-reset conditions.
2. Exists only when FIFO Depth is set to 16 or 256.
3. TOW = Toggle on write. Writing a 1 to a bit position within the register causes the corresponding bit position in the register to toggle.
4. All these registers are present when the Enable Performance Mode parameter is not selected (Legacy mode) or selected (Enhanced mode). There is a separate set of registers for XIP mode.
5. All these registers have reserved bits which are not accessible for writing. These bits always return 0 on a read.
6. The DRR and DTR values are undefined on reset and return random values if read.
7. If read only registers are addressed with a write transaction, the register contents are not updated, but the transaction will be completed gracefully with OKAY as a slave response.

Register Details

Software Reset Register

The Software Reset Register (SRR) permits resetting the core independently of other cores in the system.

IMPORTANT: *To activate the software-generated reset, the value of 0x0000_000a must be written to the Software Reset Register.*

Writing 0x0000_000a to the SRR resets the core register for four AXI clock cycles. Any other write access generates undefined results and results in an error. The bit assignment in the software reset register is shown in Figure 2-1 and described in Table 2-10. Any attempt to read this register returns undefined data.

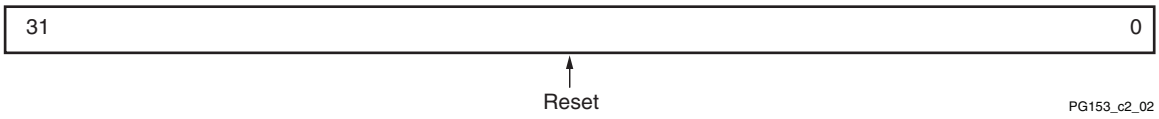

Figure 2-1: **Software Reset Register (Core Base Address + 0x40)**

Table 2-10: **Software Reset Register Description (Core Base Address + 0x40)**

Bit(s)	Name	Core Access	Reset Value	Description
31 – 0	Reset	Write only	N/A	The only allowed operation on this register is a write of 0x0000000a, which resets the AXI Quad SPI core.

SPI Control Register

The SPI Control Register (SPICR) allows programmer control over various aspects of the AXI Quad SPI core. The bit assignment in the SPICR is shown in Figure 2-2 and described in Table 2-11.

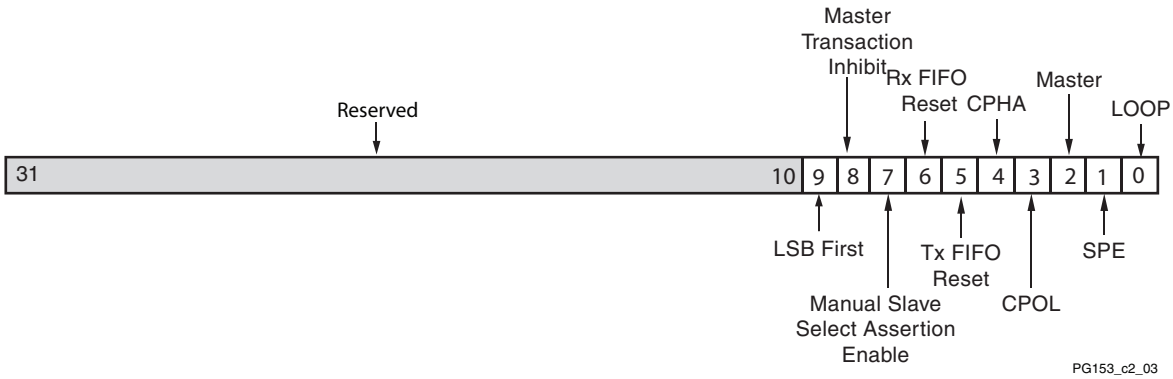

Figure 2-2: **SPI Control Register (Core Base Address + 0x60)**

Table 2-11: **SPI Control Register Description (Core Base Address + 0x60)**

Bit(s)	Name	Core Access	Reset Value	Description
31 – 10	Reserved	NA	NA	Reserved.
9	LSB First	R/W	0	LSB first:(1) This bit selects LSB first data transfer format. The default transfer format is MSB first. When set to: 0 = MSB first transfer format. 1 = LSB first transfer format. ***Note:*** In Dual/Quad SPI mode only the MSB first mode of the core is allowed.
8	Master Transaction Inhibit	R/W	1	Master transaction inhibit: This bit inhibits master transactions. This bit has no effect on slave operation. When set to: 0 = Master transactions enabled. 1 = Master transactions disabled.

Table 2-11: **SPI Control Register Description (Core Base Address + 0x60) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
7	Manual Slave Select Assertion Enable	R/W	1	Manual slave select assertion enable: This bit forces the data in the slave select register to be asserted on the slave select output anytime the device is configured as a master and the device is enabled (SPE asserted). This bit has no effect on slave operation. When set to: 0 = Slave select output asserted by master core logic. 1 = Slave select output follows data in slave select register. ***Note:*** The manual slave assertion mode is supported in standard SPI mode only.
6	RX FIFO Reset	R/W	0	Receive FIFO reset: When written to 1, this bit forces a reset of the receive FIFO to the empty condition. One AXI clock cycle after reset, this bit is again set to 0. When set to: 0 = Receive FIFO normal operation. 1 = Reset receive FIFO pointer.
5	TX FIFO Reset	R/W	0	Transmit FIFO reset: When written to 1, this bit forces a reset of the transmit FIFO to the empty condition. One AXI clock cycle after reset, this bit is again set to 0. When set to: 0 = Transmit FIFO normal operation. 1 = Reset transmit FIFO pointer.
4	CPHA	R/W	0	Clock phase:(2) Setting this bit selects one of two fundamentally different transfer formats. See SPI Clock Phase and Polarity Control in Chapter 3.
3	CPOL	R/W	0	Clock polarity:(2) Setting this bit defines clock polarity. When set to: 0 = Active-High clock; SCK idles Low. 1 = Active-Low clock; SCK idles High.
2	Master	R/W	0	Master (SPI master mode):(3) Setting this bit configures the SPI device as a master or a slave. When set to: 0 = Slave configuration. 1 = Master configuration. ***Note:*** In dual/quad SPI mode only the master mode of the core is allowed.

Table 2-11: **SPI Control Register Description (Core Base Address + 0x60) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
1	SPE	R/W	0	SPI system enable: Setting this bit to 1 enables the SPI devices as noted here. When set to: 0 = SPI system disabled. Both master and slave outputs are in 3-state and slave inputs are ignored. 1 = SPI system enabled. Master outputs active (for example, IO0 (MOSI) and SCK in idle state) and slave outputs become active if $\overline{SS}$ becomes asserted. The master starts transferring when transmit data is available.
0	LOOP	R/W	0	Local loopback mode:(4) Enables local loopback operation and is functional only in standard SPI master mode. When set to: 0 = Normal operation. 1 = Loopback mode. The transmitter output is internally connected to the receiver input. The receiver and transmitter operate normally, except that received data (from remote slave) is ignored. ***Note:*** The loopback mode is supported only when **Mode** is set to **Standard**.

Notes:

1. Setting of this bit (LSB First) is allowed only in Standard SPI mode. Dual/quad SPI modes support MSB first mode only. If in dual/quad SPI mode, if this bit is set then the corresponding error bit is set in SPISR and an interrupt is generated.
2. In dual and quad SPI mode, values for CPHA-CPOL of either 00 or 11 are allowed. Setting of other configurations causes a malfunction while communicating with memory. The setting of mode 0 or 3 is mandatory only when the targeted memory is either Winbond or Numonyx. If other values are set, then the corresponding error bit is set in the SPISR and an interrupt is generated if the corresponding bit is enabled in the SPI IPIER register.
3. The slave mode support is available only in standard SPI mode. In dual or quad SPI mode only the master mode of the core. If other values are set, then the corresponding error bit is set in SPISR and an interrupt is generated if the corresponding bit is enabled in the SPI IPIER register.
4. Loopback is allowed in standard SPI mode.

SPI Status Register

The SPI Status Register (SPISR) is a read-only register that provides the status of some aspects of the AXI Quad SPI core to the programmer. The bit assignment in the SPISR is shown in Figure 2-3 and described in Table 2-12. Writing to the SPISR does not modify the register contents.

Figure 2-3: **SPI Status Register (Core Base Address + 0x64)**

Table 2-12: **SPI Status Register Description (Core Base Address + 0x64)**

Bit(s)	Name	Core Access	Reset Value	Description
31 – 11	Reserved	N/A	N/A	Reserved
10	Command Error	Read	0	Command error flag. When set to: 0 = Default. 1 = When the core is configured in dual/quad SPI mode and the first entry in the SPI DTR FIFO (after reset) do not match with the supported command list for the particular memory, this bit is set. ***Note:*** Command error is only applicable when the core is configured either in dual or quad mode in legacy or enhanced mode AXI4 interface.
9	Loopback Error	Read	0	Loopback error flag. When set to: 0= Default. The loopback bit in the control register is at default state. 1 = When the SPI command, address, and data bits are set to be transferred in other than standard SPI protocol mode and this bit is set in control register (SPICR). ***Note:*** Loopback is only allowed when the core is configured in standard mode. Other modes setting of the bit causes an error and the interrupt bit is set in legacy or enhanced mode AXI4 interface.
8	MSB Error	Read	0	MSB error flag. When set to: 0= Default. 1 = This bit is set when the core is configured to transfer the SPI transactions in either dual or quad SPI mode and LSB first bit is set in the control register (SPICR). ***Note:*** In dual/quad SPI mode, only the MSB first mode of the core is allowed. MSB error flag is only applicable when the core is configured either in dual or quad mode in legacy or enhanced mode AXI4 interface.

Table 2-12: **SPI Status Register Description (Core Base Address + 0x64) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
7	Slave Mode Error	Read	1	Slave mode error flag. When set to: 1 = This bit is set when the core is configured with dual or quad SPI mode and master is set to 0 in the control register (SPICR). 0 = Master mode is set in the control register (SPICR). In Dual ***Note:*** Quad SPI mode, only the master mode of the core is allowed. Slave mode error flag is only applicable when the core is configured either in dual or quad mode in legacy or enhanced AXI4 mode interface.
6	CPOL_CPHA_ Error	Read	0	CPOL_CPHA_Error flag. When set to: 0 = Default. 1 = The CPOL and CPHA are set to 01 or 10. When the SPI memory is chosen as either Winbond or Numonyx, and CPOL=CPHA are configured as 01 or 10, this bit is set. These memories support CPOL=CPHA mode in 00 or in 11 mode. CPOL_CPHA_Error flag is only applicable when the core is configured either in dual or quad mode in legacy or enhanced mode AXI4 interface.
5	Slave_Mode_ Select	Read	1	Slave_Mode_Select flag. This flag is asserted when the core is configured in slave mode. Slave_Mode_Select is activated as soon as the master SPI core asserts the chip select pin for the core. 1 = Default in standard mode. 0 = Asserted when core configured in slave mode and selected by external SPI master.
4	MODF	Read	0	Mode-fault error flag. This flag is set if the $\overline{SS}$ signal goes active while the SPI device is configured as a master. MODF is automatically cleared by reading the SPISR. A Low-to-High MODF transition generates a single-cycle strobe interrupt. 0 = No error. 1 = Error condition detected.
3	Tx_Full	Read	0	Transmit full. When a transmit FIFO exists, this bit is set High when the transmit FIFO is full. ***Note:*** When FIFOs do not exist, this bit is set High when an AXI write to the transmit register has been made (this option is available only in standard SPI mode). This bit is cleared when the SPI transfer is completed.

Table 2-12: **SPI Status Register Description (Core Base Address + 0x64) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
2	Tx_Empty	Read	1	Transmit empty. When a transmit FIFO exists, this bit is set to High when the transmit FIFO is empty. The occupancy of the FIFO is decremented with the completion of each SPI transfer. ***Note:*** When FIFOs do not exist, this bit is set with the completion of an SPI transfer (this option is available only in standard SPI mode). Either with or without FIFOs, this bit is cleared on an AXI write to the FIFO or transmit register. For Dual/Quad SPI mode, the FIFO is always present in the core.
1	Rx_Full	Read	0	Receive full. When a receive FIFO exists, this bit is set High when the receive FIFO is full. The occupancy of the FIFO is incremented with the completion of each SPI transaction. ***Note:*** When FIFOs do not exist, this bit is set High when an SPI transfer has completed (this option is available only in standard SPI mode). Rx_Empty and Rx_Full are complements in this case.
0	Rx_Empty	Read	1	Receive Empty. When a receive FIFO exists, this bit is set High when the receive FIFO is empty. The occupancy of the FIFO is decremented with each FIFO read operation. ***Note:*** When FIFOs do not exist, this bit is set High when the receive register has been read (this option is available only in standard SPI mode). This bit is cleared at the end of a successful SPI transfer. For dual/quad SPI mode, the FIFO is always present in the core.

SPI Data Transmit Register

The SPI Data Transmit Register (SPI DTR) is written with the data to be transmitted on the SPI bus. After the SPE bit is set to 1 in master mode or `spisel` is active in the slave mode, the data is transferred from the SPI DTR to the shift register.

The SPI DTR should be in reset state before filling so that the DTR FIFO write pointer is pointing to the 0th location.

Dual/Quad Mode

The first write must always be a SPI command from AXI transactions, followed by the address (either 24-bit or 32-bit), then filled with the data to be transmitted. When reading the status register of the memory, then as per the command requirements, this register should be filled with dummy bytes along with a command and address (optional). In one of these modes, the dummy bytes are required to fill in the DTR which is used for the internal count of the data reception from memory.

In commands such as dual mode read, these bytes are not transferred on the data lines, but are used by the internal logic to keep track of the number of data bytes to be read from the SPI slave memory.

If a transfer is in progress, the data in the SPI DTR is loaded into the shift register as soon as the data in the shift register is transferred to the SPI DRR and a new transfer starts. The data is held in the SPI DTR until replaced by a subsequent write. The SPI DTR is shown in Figure 2-4 and Table 2-13 shows the data format.

When a transmit FIFO exists, data is written directly into the FIFO and the first location in the FIFO is treated as the SPI DTR. The pointer is decremented after completion of each SPI transfer. The choice of inclusion or exclusion of the FIFO in the design is available only when the core is configured in Standard SPI mode. If the core is configured in dual or quad SPI mode, the FIFO always exists. In this mode, the FIFO depth is defined with the parameter **FIFO Depth** (allowed values are 16 or 256).

This register cannot be read and can only be written when it is known that space for the data is available. If an attempt to write is made on a full register or FIFO, the AXI write transaction completes with an error condition. Reading the SPI DTR is not allowed and the read transaction results in undefined data.

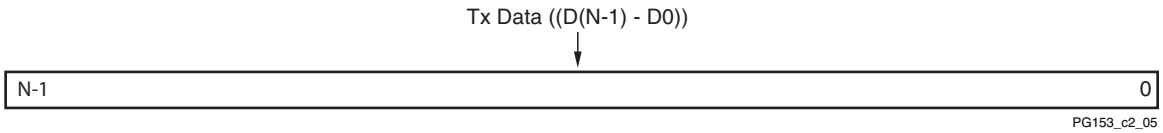

Figure 2-4: **SPI Data Transmit Register (Core Base Address + 0x68)**

Table 2-13: **SPI Data Transmit Register Description (Core Base Address + 0x68)**

Bit(s)	Name	Core Access	Reset Value	Description
[N–1]–0	TX Data[1] (D_{N-1} – D_0)	Write only	0	N-bit SPI transmit data. N can be 8, 16 or 32.[2] N = 8 when the Transfer Width parameter is 8. N = 16 when the Transfer Width parameter is 16. N = 32 when the Transfer Width parameter is 32.

Notes:

1. The D_{N-1} bit always represents the MSB bit irrespective of LSB first or MSB first transfer selection. When the Transfer Width parameter is 8 or 16, the unused upper bits ((AXI data width – 1) to N) are reserved.
2. In standard SPI mode, the width of this register can be 8 or 16 or 32 based on the core configuration. In dual or quad SPI mode, this register is 8 bits wide.

SPI Data Receive Register

The SPI Data Receive Register (SPI DRR) is used to read data that is received from the SPI bus. This is a double-buffered register. The received data is placed in this register after each complete transfer. The SPI architecture does not provide any means for a slave to throttle traffic on the bus; consequently, the SPI DRR is updated following each completed transaction only if the SPI DRR was read prior to the last SPI transfer.

If the SPI DRR was not read and is full, the most recently transferred data is lost and a receive overrun interrupt occurs. The same condition can also occur with a master SPI device.

The choice of inclusion (**FIFO Depth** = 16 or 256) or exclusion (**FIFO Depth** = 0) of the FIFO in the design is available only when the core is configured in standard SPI mode. When the core is configured in dual or quad SPI mode, the FIFO always exists. In this mode, the FIFO depth is defined with the parameter **FIFO Depth** (allowed values are 16 or 256). For both master and slave SPI configuration mode of the core (in Standard SPI mode only) with a receive FIFO, the data is buffered in the FIFO. The receive FIFO is a read-only buffer. If an attempt is made to read an empty receive register or FIFO, the AXI read transaction completes successfully with undefined data. Writes to the SPI DRR do not modify the register contents and return with a successful OK response.

The power-on-reset values for the SPI DRR are unknown. When known data has been written into the receive FIFO during core transactions, the data in this register can be considered for reading. The SPI DRR is shown in Figure 2-5, while the specifics of the data format is described Table 2-14.

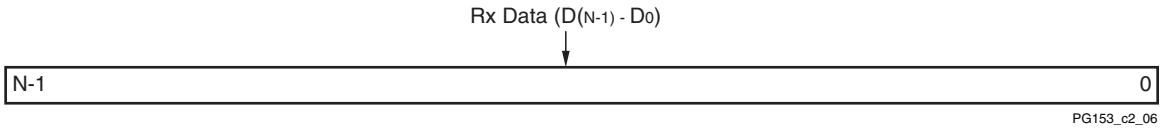

Figure 2-5: **SPI Data Receive Register (Core Base Address + 0x6C)**

Table 2-14: **SPI Data Receive Register Description (Core Base Address + 0x6C)**

Bit(s)	Name	Core Access	Reset Value	Description
[N–1]–0	RX Data[1] (D_{N-1} - D_0)	Read only	N/A	N-bit SPI receive data. N can be 8, 16 or 32.[2] N = 8 when the Transfer Width parameter is 8. N = 16 when the Transfer Width parameter is 16. N = 32 when the Transfer Width parameter is 32.

Notes:

1. The D_{N-1} bit always represents the MSB bit irrespective of LSB first or MSB first transfer selection. When the Transfer Width parameter is 8 or 16, the unused upper bits ((AXI data width – 1) to N) are reserved.
2. In standard SPI mode, the width of this register can be 8 or 16 or 32 based on the core configuration. In dual or quad SPI mode, this register is 8 bit wide.

SPI Slave Select Register

The SPI Slave Select Register (SPISSR) contains an active-Low, one-hot encoded slave select vector $\overline{SS}$ of length N, where N is the number of slaves set by the **No. of Slaves** parameter. The $\overline{SS}$ vector occupies the right-most bits of the register. At most, one bit can be asserted Low. This bit denotes the slave with which the local master communicates. The bit assignment in the SPISSR is shown in Figure 2-6 and described in Table 2-15.

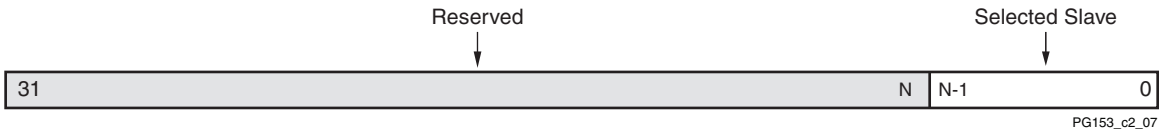

Figure 2-6: **SPI Slave Select Register (Core Base Address + 0x70)**

Table 2-15: **SPI Slave Select Register Description (Core Base Address + 0x70)**

Bit(s)	Name	Core Access	Reset Value	Description
31 – N	Reserved	N/A	N/A	Reserved
[N–1]–0	Selected Slave	R/W	1	Active-Low, one-hot encoded slave select vector of length N-bits. N must be less than or equal to the data bus width (32-bit). The slaves are numbered right to left starting at zero with the LSB. The slave numbers correspond to the indexes of signal $\overline{SS}$.

SPI Transmit FIFO Occupancy Register

The SPI Transmit FIFO Occupancy Register (TX_FIFO_OCY) is present only if the AXI Quad SPI core is configured with FIFOs (**FIFO Depth** = 16 or 256). If it is present and if the transmit FIFO is not empty, the register contains a four-bit, right-justified value that is one less than the number of elements in the FIFO (occupancy minus one).

This register is read-only. When written, or read when the FIFO is empty, the register contents are not affected. The only reliable way to determine that the transmit FIFO is empty is by reading the Tx_Empty status bit in the SPI Status Register or the DTR empty bit in the interrupt status register. The transmit FIFO occupancy register is shown in Figure 2-7, while the specifics of the data format are described in Table 2-16.

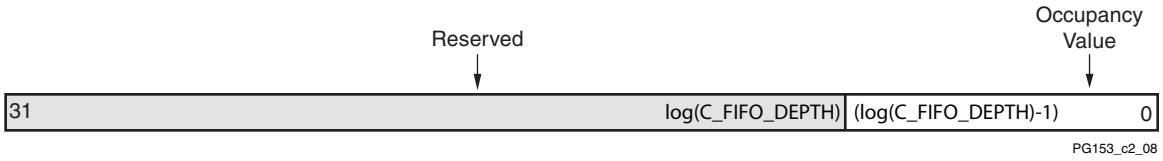

Figure 2-7: **SPI Transmit FIFO Occupancy Register (Core Base Address + 0x74)**

Table 2-16: **SPI Transmit FIFO Occupancy Register Description (Core Base Address + 0x74)**

Bit(s)	Name	Core Access	Reset Value (hex)	Description
31 – log(FIFO Depth)	Reserved	N/A	N/A	Reserved
(log(FIFO Depth)–1)–0	Occupancy Value	Read	0	The binary value plus 1 yields the occupancy.

Notes:

1. In standard SPI mode, only bits 3 – 0 of this register are valid as the FIFO depth is limited to 16.
2. In dual or quad SPI mode, the bit position is changed based on the FIFO depth. A maximum FIFO depth of 256 bytes is allowed in the design.

SPI Receive FIFO Occupancy Register

The SPI Receive FIFO Occupancy Register (RX_FIFO_OCY) is present only if the AXI Quad SPI core is configured with FIFOs (**FIFO Depth** = 16 or 256). If the register is present and if the receive FIFO is not empty, the register contains a four-bit, right-justified value that is one less than the number of elements in the FIFO (occupancy minus one).

This register is read-only. A write to it (or of a read when the FIFO is empty) does not affect the register contents. The only reliable way to determine that the receive FIFO is empty is by reading the `Rx_Empty` status bit in the SPI status register.

The receive FIFO occupancy register is shown in Figure 2-8, while the specifics of the data format are described in Table 2-17.

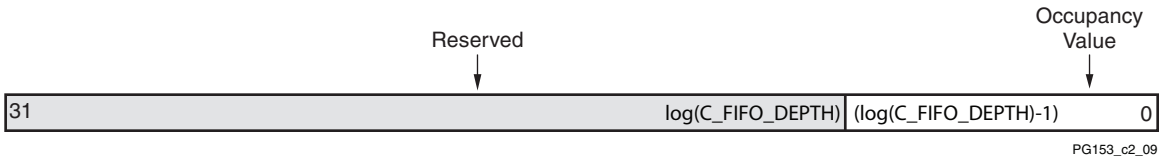

Figure 2-8: **SPI Receive FIFO Occupancy Register (Core Base Address + 0x78)**

Table 2-17: **SPI Receive FIFO Occupancy Register Description (Core Base Address + 0x78)**

Bit(s)	Name	Core Access	Reset Value (hex)	Description
31–log(FIFO Depth)	Reserved	N/A	N/A	Reserved
(log(FIFO Depth)–1)–0	Occupancy Value	Read	0	The binary value plus 1 yields the occupancy.

Notes:

1. In standard SPI mode, only bits 3 – 0 of this register are valid as the FIFO depth is limited to 16.
2. In dual or quad SPI mode, the bit position is changed based on the FIFO depth. A maximum FIFO depth of 256 bytes is allowed in the design.

Interrupt Register Set Description

The AXI Quad SPI core has many distinct interrupts that are sent to the interrupt controller. The core interrupt controller allows each interrupt to be enabled independently (using the IP interrupt enable register (IPIER)). The interrupt registers are contained within the interrupt controller. An interrupt strobe can be generated under multiple conditions or only after a transfer completion. In standard, dual or quad SPI mode, and master mode, when the parameter **FIFO Depth** is set to 16 or 256, almost all of the interrupts shown in Table 2-19 are available.

In Standard SPI mode, when **FIFO Depth** is set to 0, all of the interrupts are available except:

- Bit 6: Transmit FIFO half empty
- Bit 8: DRR not empty (not present in this mode)

Device Global Interrupt Enable Register

The Device Global Interrupt Enable Register (DGIER) is used to globally enable the final interrupt output from the interrupt controller as shown in Figure 2-9 and described in Table 2-18. This is a read/write bit and is cleared on reset.

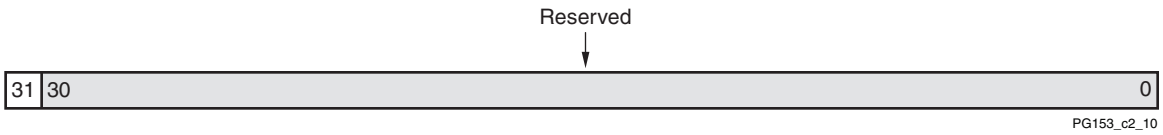

Figure 2-9: **Device Global Interrupt Enable Register (Core Base Address + 0x1C)**

Table 2-18: **Device Global Interrupt Enable Register Description (Core Base Address + 0x1C)**

Bit(s)	Name	Core Access	Reset Value	Description
31	GIE	R/W	0	Global Interrupt Enable. Allows passing all individually enabled interrupts to the interrupt controller. When set to: 0 = Disabled. 1 = Enabled.
30 – 0	Reserved	N/A	N/A	Reserved

IP Interrupt Status Register (IPISR)

Up to fourteen unique interrupt conditions are possible depending on whether the system is configured with FIFOs or not, as well as if it is configured in master mode or slave mode. A system without FIFOs has seven interrupts. The 32-bit interrupt status register within the interrupt controller can enable each interrupt independently. The IP Interrupt Status Register (IPISR) collects all of the interrupt events. Bit assignments are shown in Figure 2-10 and described in Table 2-19. The interrupt register is a read/toggle-on-write register. Writing a 1 to a bit position within the register causes the corresponding bit to *toggle*. All register bits are cleared on reset.

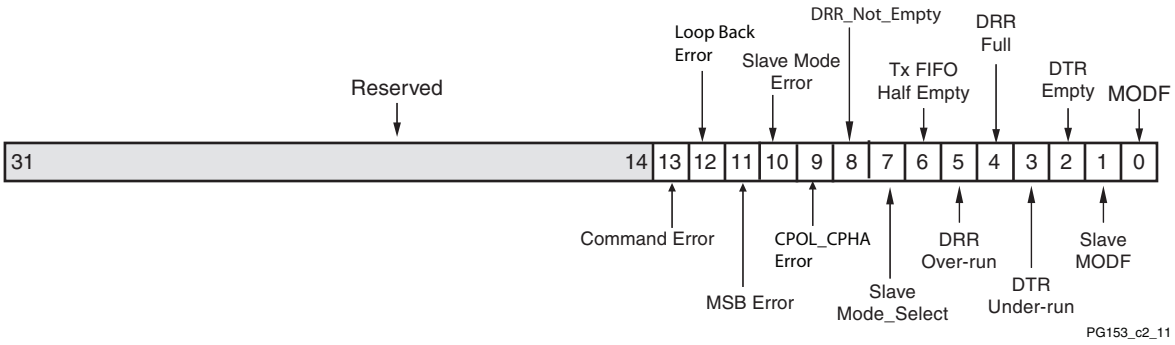

Figure 2-10: **IP Interrupt Status Register (Core Base Address + 0x20)**

Table 2-19: **IP Interrupt Status Register Description (Core Base Address + 0x20)**

Bit(s)	Name	Core Access	Reset Value	Description
31 – 14	Reserved	N/A	N/A	Reserved
13	Command Error	R/TOW(1)	0	Command error. IPISR bit(13) is the command error. When set to: 1 = This flag is asserted when: • The core is configured in dual/quad SPI mode and • The first entry in the SPI DTR FIFO (after reset) does not match with the supported command list for particular memory. When the SPI command in DTR FIFO does not match with the internal supported command list, the core completes the SPI transactions in standard SPI format. This bit is set to show this behavior of the core. In standard SPI mode this bit is always in default state.
12	Loopback Error	R/TOW(1)	0	Loopback error. IPISR bit(12) is the loopback error. When set to: 1 = This flag is asserted when: • The core is configured in dual or quad SPI transfer mode and • The LOOP bit is set in control register (SPICR(0)). In standard SPI mode, this bit is always in default state.
11	MSB Error	R/TOW(1)	0	MSB error. IPISR bit(11) is the MSB error. When set to: 1 = This flag is asserted when: • The core is configured in either dual or quad SPI mode and • The LSB First bit in the control register (SPICR) is set to 1. In standard SPI mode, this bit is always in default state.
10	Slave Mode Error	R/TOW(1)	1	I/O mode instruction error. IPISR bit(10) is the slave mode error. This flag is asserted when: • The core is configured in either dual or quad SPI mode and • The core is configured in master = 0 in control register (SPICR(2)). In standard SPI mode, this bit is always in default state.
9	CPOL_CPHA Error	R/TOW(1)	0	CPOL_CPHA error. IPISR bit(9) is the CPOL_CPHA error. This flag is asserted when: • The core is configured in either dual or quad SPI mode and • The CPOL - CPHA control register bits are set to 01 or 10. In standard SPI mode, this bit is always in default state.

Table 2-19: **IP Interrupt Status Register Description (Core Base Address + 0x20) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
8	DRR_Not_Empty	R/TOW[1]	0	DRR not empty. IPISR bit(8) is the DRR not empty bit. The assertion of this bit is applicable only in the case where **FIFO Depth** is 16 or 256 and the core is configured in slave mode and standard SPI mode. This bit is set when the DRR FIFO receives the first data value during the SPI transaction. This bit is set by a one-clock period strobe to the interrupt register when the core receives the first data beat. ***Note:*** The assertion of this bit is applicable only when the **FIFO Depth** parameter is 16 or 256 and the core is configured in slave mode in standard SPI mode. When **FIFO Depth** is set to 0, this bit always returns 0. This bit has no significance in dual/quad mode.
7	Slave_Select_Mode	R/TOW[1]	0	Slave select mode. IPISR bit(7) is the slave select mode bit. The assertion of this bit is applicable only when the core is configured in slave mode in standard SPI configuration. This bit is set when the other SPI master core selects the core by asserting the slave select line. This bit is set by a one-clock period strobe to the interrupt register. ***Note:*** This bit is applicable only in standard SPI slave mode.
6	TX FIFO Half Empty	R/TOW[1]	0	Transmit FIFO half empty. In standard SPI configuration, IPISR bit(6) is the transmit FIFO half-empty interrupt. For example, when FIFO depth = 16, this bit is set by a one-clock period strobe to the interrupt register when the occupancy value is decremented from `1000` to `0111`. Note that `0111` means there are 8 elements in the FIFO to be transmitted. In this mode, the FIFO depth is fixed to 16 only. The same logic applies when the FIFO depth is 256 where `10000000` changes to `01111111`. In dual or quad SPI configuration, based on the FIFO depth, this bit is set at half-empty condition. ***Note:*** This interrupt exists only if the AXI Quad SPI core is configured with FIFOs (In standard, dual or quad SPI mode).
5	DRR Overrun	R/TOW[1]	0	Data receive register/FIFO overrun. IPISR bit 5 is the data receive FIFO overrun interrupt. This bit is set by a one-clock period strobe to the interrupt register when an attempt to write data to a full receive register or FIFO is made by the SPI core logic to complete a SPI transfer. This can occur when the SPI device is in either master or slave mode (in standard SPI mode) or if the IP is configured in SPI master mode (dual or quad SPI mode).

Table 2-19: **IP Interrupt Status Register Description (Core Base Address + 0x20) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
4	DRR Full	R/TOW[1]	0	Data receive register/FIFO full. IPISR bit 4 is the data receive register full interrupt. Without FIFOs, this bit is set at the end of a SPI element transfer by a one-clock period strobe to the interrupt register (An element can be a byte, half-word, or word depending on the value of **Transfer Width**). With FIFOs, this bit is set at the end of the SPI element transfer, when the receive FIFO has been completely filled by a one-clock period strobe to the interrupt register.
3	DTR Underrun	R/TOW[1]	0	Data transmit register/FIFO underrun. IPISR bit 3 is the data transmit register/FIFO under-run interrupt. This bit is set at the end of a SPI element transfer by a one-clock period strobe to the interrupt register when data is requested from an empty transmit register/FIFO by the SPI core logic to perform a SPI transfer. This can occur only when the SPI device is configured as a slave in standard SPI configuration and is enabled by the SPE bit as set. All zeros are loaded in the shift register and transmitted by the slave in an under-run condition.
2	DTR Empty	R/TOW[1]	0	Data transmit register/FIFO empty. IPISR bit 2 is the data transmit register/FIFO empty interrupt. Without FIFOs, this bit is set at the end of a SPI element transfer by a one-clock period strobe to the interrupt register. With FIFOs, this bit is set at the end of the SPI element transfer when the transmit FIFO is emptied by a one-clock period strobe to the interrupt register. See Transfer End Period in Chapter 3. In the context of the M68HC11 reference manual, when configured without FIFOs, this interrupt is equivalent in information content to the complement of the SPI transfer complete flag ($\overline{\text{SPIF}}$) interrupt bit. In master mode if this bit is set to 1, no more SPI transfers are permitted.
1	Slave MODF	R/TOW[1]	0	Slave mode-fault error. IPISR bit 1 is the slave mode-fault error flag. This interrupt is generated if the $\overline{\text{SS}}$ signal goes active while the SPI device is configured as a slave, but is not enabled. This bit is set immediately on $\overline{\text{SS}}$ going active and continually set if $\overline{\text{SS}}$ is active and the device is not enabled.
0	MODF	R/TOW[1]	0	Mode-fault error. IPISR bit 0 is the mode-fault error flag. This interrupt is generated if the $\overline{\text{SS}}$ signal goes active while the SPI device is configured as a master. This bit is set immediately on $\overline{\text{SS}}$ going active.

Notes:

1. TOW = Toggle On Write. Writing a 1 to a bit position within the register causes the corresponding bit position in the register to toggle.

IP Interrupt Enable Register (IPIER)

The Interrupt Enable Register (IPIER) register allows the system interrupt output to be active. This interrupt is generated if an active bit in the IPISR register corresponds to an enabled bit in the IPIER register. The IPIER register has an enable bit for each defined bit of the IPISR as shown in Figure 2-11 and described in Table 2-20. All bits are cleared on reset.

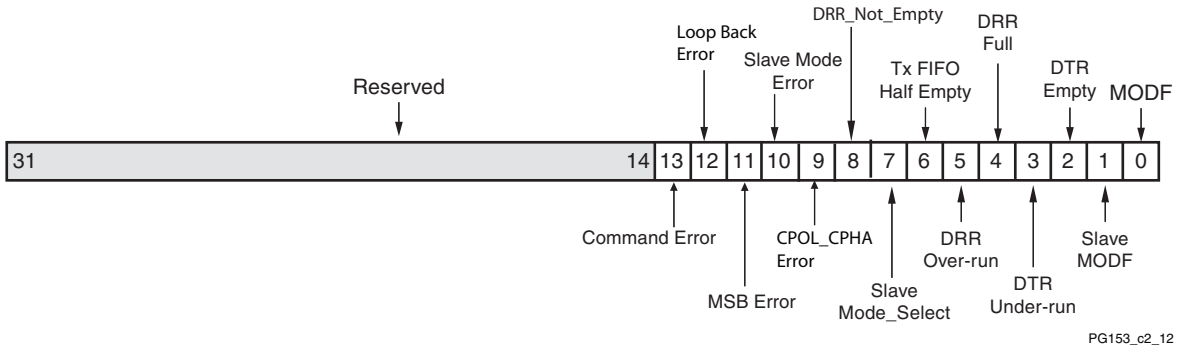

Figure 2-11: **IP Interrupt Enable Register (Core Base Address + 0x28)**

Table 2-20: **IP Interrupt Enable Register Description (Core Base Address + 0x28)**

Bit(s)	Name	Core Access	Reset Value	Description
31 – 14	Reserved	N/A	N/A	Reserved
13	Command Error	R/W	0	Command error. 0 = Disabled. 1 = Enabled. This bit is applicable only when the core is configured in dual or quad SPI mode.
12	Loopback Error	R/W	0	Loopback error. 0 = Disabled. 1 = Enabled. This bit is applicable only when the core is configured in dual or quad SPI mode.
11	MSB Error	R/W	0	MSB error. 0 = Disabled. 1 = Enabled. This bit is applicable only when the core is configured in dual or quad SPI mode.
10	Slave Mode Error	R/W	0	I/O mode instruction error. 0 = Disabled. 1 = Enabled. This bit is applicable only when the core is configured in dual or quad SPI mode.

Table 2-20: **IP Interrupt Enable Register Description (Core Base Address + 0x28) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
9	CPOL_CPHA Error	R/W	0	CPOL_CPHA error. 0 = Disabled. 1 = Enabled. This bit is applicable only when the core is configured in dual or quad SPI mode.
8	DRR_Not_ Empty	R/W	0	DRR_Not_Empty. 0 = Disabled. 1 = Enabled. ***Note:*** The setting of this bit is applicable only when **FIFO Depth** is set to 1 and the core is configured in slave mode of standard SPI mode. If **FIFO Depth** is set to 0, the setting of this bit has no effect. This is allowed only in standard SPI configuration. It means this bit is not set in the IPIER register. Therefore, this bit should only be used when **FIFO Depth** is set to 1 and when the core is configured in slave mode. This bit has no significance in dual or quad mode.
7	Slave_Select_ Mode	R/W	0	Slave_Select_Mode. 0 = Disabled. 1 = Enabled. This bit is applicable only when the core is configured in slave mode by selecting the active-Low status on `spisel`. In master mode, setting this bit has no effect.
6	TX FIFO Half Empty	R/W	0	Transmit FIFO half empty. 0 = Disabled. 1 = Enabled. ***Note:*** This bit is meaningful only if the AXI Quad SPI core is configured with FIFOs.
5	DRR Overrun	R/W	0	Receive FIFO overrun. 0 = Disabled. 1 = Enabled.
4	DRR Full	R/W	0	Data receive register/FIFO full. 0 = Disabled. 1 = Enabled.
3	DTR Underrun	R/W	0	Data transmit FIFO underrun. 0 = Disabled. 1 = Enabled.
2	DTR Empty	R/W	0	Data transmit register/FIFO empty. 0 = Disabled. 1 = Enabled.

Table 2-20: **IP Interrupt Enable Register Description (Core Base Address + 0x28) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
1	Slave MODF	R/W	0	Slave mode-fault error flag. 0 = Disabled. 1 = Enabled.
0	MODF	R/W	0	Mode-fault error flag. 0 = Disabled. 1 = Enabled.

XIP Mode

When the AXI Quad SPI core is configured in XIP mode, only these registers are available through the AXI4-Lite interface:

- XIP Configuration Register
- XIP Status Register

These 32-bit registers are configurable and accessible individually through the AXI4-Lite interface. Table 2-21 provides a summary of the AXI Quad SPI core registers in XIP mode.

Table 2-21: **Core Registers in Enhanced Mode XIP Mode**

Base Address Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
Core Grouping				
60	XIP Config_Reg (XIP-CR)	R/W	0x0	XIP configuration register
64	XIP Status_Reg (XIP-SR)	Read	0x0	XIP status register

XIP Mode Commands

The registers in XIP mode are accessed through the AXI4-Lite interface. Based on the core configuration in this mode, the core supports three read commands:

- Fast Read (0x0Bh)
- Fast Read Dual I/O (0xBBh)
- Fast Read Quad I/O (0xEBh)

Mode determines which one of these commands is supported by the core. These commands are built in and no other commands need to be configured. When **Mode** is set, the same command operates during the whole transaction.

XIP Configuration Register

The bit assignments for the XIP Configuration Register (XIP-CR) are shown in Figure 2-12. This register is used to configure the XIP (read-only) mode of operation. This is a read/write register used to configure the CPOL and CPHA modes. In XIP mode, either CPOL–CPHA = 00 or CPOL–CPHA = 11 is supported. If any other combination is selected, the error flag is set in the status register and the transaction on the AXI4 interface is not accepted by the core. Before the start of any new AXI4 transaction, the core checks the CPOL and CPHA settings.

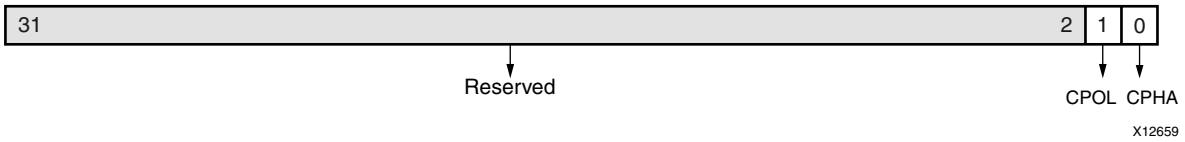

Figure 2-12: **XIP Configuration Register (Core Base Address+0x60)**

XIP Status Register

The bit assignments for the XIP Status Register (XIP-SR) are shown in Figure 2-13 and described in Table 2-22. This register is used to check the status of commands and other processes performed by the core. In the case where the core receives write commands or write transactions, an error is generated and this is indicated in the status register. This is a read-only register. Any attempt to write to this register completes with a standard acknowledge and the register contents are not updated. The contents of this register are reset as soon as it is read. As there is no timeout counter involved in the core, if the AXI4 transaction is not accepted, check the status registers for any errors. If there are errors, the core does not accept the AXI4 transaction.

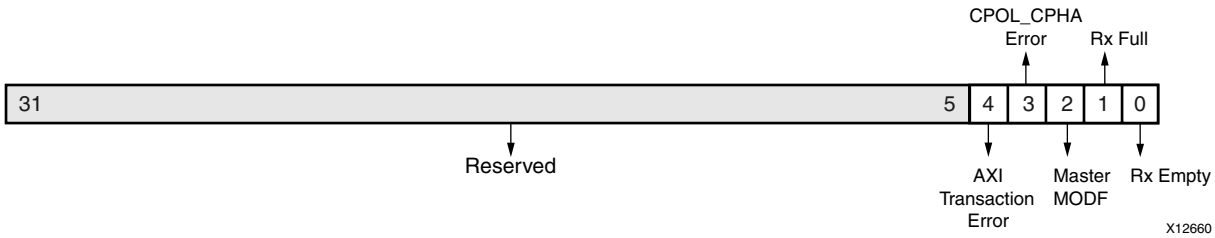

Figure 2-13: **XIP Status Register (Core Base Address+0x64)**

Table 2-22: **XIP Status Register Description (Core Base Address+0x64)**

Bit(s)	Name	Core Access	Reset Value	Description
31 – 5	Reserved	N/A	0	Reserved
4	AXI Transaction Error	R	0	AXI transaction error

Table 2-22: **XIP Status Register Description (Core Base Address+0x64) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
3	CPOL_CPHA Error	R	0	CPOL_CPHA error
2	Master MODF	R	0	Master mode fault. This bit is set to 1 if the `spisel` line is deasserted.
1	RX Full	R	0	Receiver full
0	RX Empty	R	1	Receiver empty

Specification Exceptions

Exceptions from the Motorola M68HC11-Rev. 4.0 Reference Manual

1. A slave mode-fault error is added to provide an interrupt if a SPI device is configured as a slave and is selected when not enabled.
2. In this design, the SPI DTR and SPI DRR registers have independent addresses. This is an exception to the M68HC11 specification which calls for two registers to have the same address.
3. All $\overline{SS}$ signals are required to be routed between SPI devices internally to the FPGA. This is because toggling of the $\overline{SS}$ signal is utilized in slaves to minimize FPGA resources.
4. Manual control of the $\overline{SS}$ signals is provided by setting bit 7 in the SPICR register. When the device is configured as a master and is enabled while bit 7 of the SPICR register is set, the vector in the SPISSR register is asserted. When this mode is enabled, multiple elements can be transferred without toggling the $\overline{SS}$ vector.
5. A control bit is provided to inhibit master transfers. This bit is effective in any master mode, but its primary intent is manual control of the $\overline{SS}$ signals.
6. In the M68HC11 implementation, the transmit register is transparent to the shift register which necessitates the write collision error (WCOL) detection hardware. This feature is not implemented in this design.
7. The interrupt enable bit (SPIE) defined by the M68HC11 specifications which resides in the M68HC11 control register has been moved to the IPIER register. In place of the SPIE bit, there is a bit to select local master loopback mode for testing.
8. An option is implemented in this FPGA design to implement FIFOs on both transmit and receive (Full Duplex only) mode.
9. M68HC11 implementation supports only byte transfer. In this design either a byte, half-word, or word transfer can be configured using the Transaction Width parameter.

10. The baud rate generator is specified by Motorola to be programmable using bits in the control register; however, in this FPGA design the baud rate generator is programmable using parameters in the VHDL implementation. Thus, run time configuration of the baud rate is not possible. Beyond the ratios of 2, 4, 16 and 32, all integer multiples of 16 up to 2048 are allowed.
11. Most of the SPI slave devices support the SPI modes 0 and 3. For some of these devices, the data valid time of 8 ns from the falling edge of SCK is applicable. While operating with these devices at a higher speed of 50 MHz (most of the instructions support this speed), the core should be configured with **Frequency Ratio** set to 2 (where the AXI is configured to operate at 100 MHz).

    Due to limited time availability in the design as well as real SPI slave behavior for data change, the data in the SPI core is registered in the middle of each SPI rising and next consecutive falling edge inside the core. This adds
    5 ns of time to the core (while operating at 100 MHz on the `ext_spi_clk` port) for registering the data. As per the M68HC11 document, the master should register data on each rising edge of SCK in SPI modes 1 and 3. Note that the data registering mechanism when **Frequency Ratio** is 2 follows a different pattern than specified in the standard although this is applicable to the data registering mechanism in the core only. The SPI core, when configured in master mode, changes data on each falling edge and this behavior is as per the M68HC11 standard.
12. When the AXI Quad SPI core is configured in slave mode (**Mode** set to **Standard**), the data in the core is registered on the SCK rising edge + one AXI clock cycle. Internally, this data is registered on the next rising edge of the AXI clock cycle. The core changes the data on the SCK falling edge + one AXI clock cycle.
13. In Standard SPI mode of operation, when the SCK RATIO = 16 is used, the core provides approximately 7 cycles of `ext_spi_clk` set up time for the downstream device to register the data on the next SPI rising edge.

Other Exceptions

1. The AXI Quad SPI core supports one memory selection at a time. This means, in multi-slave systems, the core should be configured to select and perform operation only on one slave at a time.
2. The core is based on the specific memory parts from Winbond (W25Q80) and Numonyx (N25Q256). To test the core with other memory parts, ensure that the internal command decoding logic and its bit positions are understood for correct operation of the core. Also, check the common command set between the Winbond or Numonyx memory part data sheet and other memory parts to confirm that the common commands between these documents are executed. In quad mode, the design supports the Numonyx memory parts with HOLD functionality only. The memory parts with RESET functionality are not supported in the design.

3. See Unsupported Commands for Dual/Quad SPI Mode and Winbond or Numonyx Memory in Chapter 3.
4. Dedicated memory should be used as AXI Quad SPI slaves because the core supports a limited set of commands which are common in Winbond and Numonyx memories in terms of command, address, data requirement and their behavior. If single dedicated memories are used with the core, a wider range of commands is supported for the respective memory and performance is optimized.
5. In XIP mode, there are several clock domain crossing signals present between the AXI4 and AXI4-Lite interfaces and the SPI domain. When reading the XIP SR, note that five clock cycles are taken by the core to update the status bits.
6. The XIP configuration of the core does not support byte access mode.
7. The core does not support queued commands. The core design is based on commands supported by standard SPI devices such as Winbond and Numonyx memory only.

Designing with the Core

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

General Design Guidelines

Functionality Based on AXI Interfaces

AXI4-Lite Interface

If the AXI4-Lite interface is chosen, all the registers including the SPI DTR and SPI DRR are 32-bit, single-access registers. The SPI DTR and SPI DRR registers have only 8 valid bits out of 32.

AXI4 Memory-Mapped Interface (Enhanced Mode)

If the AXI4 memory-mapped interface is chosen, it is mandatory that all the registers are single-length access only. If burst is attempted (except for the SPI DTR or SPI DRR), the core behavior is not guaranteed. The SPI DTR and SPI DRR registers can be accessed as 32-bits. The FIXED burst is allowed only for the SPI DTR and SPI DRR registers. Out of 32 bits of FIXED burst, only 8 bits are valid. Read the occupancy registers before initiating the recursive FIXED burst. The occupancy registers provide the length of the FIXED burst to be performed. Only the last eight bits should be considered as actual data.

While carrying out the write burst operation in the DTR register, it is illegal to have holes in the write strobes. This is not allowed by the core and core behavior in this instance is not guaranteed.

When starting any new transaction at the SPI, the DTR FIFO should be always filled with a command followed by address, dummy bytes, then data bytes. This sequence should be strictly adhered to by the application. The read or write data occupancy registers indicate the number of locations left in the receive or transmit FIFO which help to determine the burst length of the next transaction. This mode is most suitable for CDMA-based applications.

AXI4 Memory Mapped Read-Only Interface (XIP Mode)

In this interface, only the read transactions are allowed from the AXI4 interface. The write transactions are not allowed and are considered an error by the core. The XIP registers are accessible through the AXI4-Lite interface. This mode is most suitable for using flash devices in ROM-based applications.

Standard SPI Device Features with only AXI4-Lite Interface

The SPI device includes these standard features in Standard SPI configuration plus those listed in the Features section of IP Facts:

- Supports multi-master configuration within the FPGA with separated _I, _O, _T representation of 3-state ports.
- Works with N times 8-bit data characters in default configuration. The default mode implements manual control of the $\overline{SS}$ output using data written to the SPISSR. This appears directly on the $\overline{SS}$ output when the master is enabled. This mode can only be used with external slave devices. An optional operation can be selected where the $\overline{SS}$ output is toggled automatically with each 8-bit character transfer by the master device using a bit in the SPICR for SPI master devices.
- Multi-master environment supported (implemented with 3-state drivers and requires software arbitration for possible conflict). See AXI4-Lite Interface Functionality in Standard SPI Multi-Master Configuration.
- Multi-slave environment supported (automatic generation of additional slave select output signals for the master).
- Supports maximum SPI clock rates up to one-half the AXI clock rate in master mode and one-fourth the AXI clock rate in slave modes (C_SCK_RATIO = 2 is not supported in slave mode due to the synchronization method used between the AXI and SPI clocks). It is required that the AXI and external clock signals be aligned when configured in slave mode.
- Parameterizable baud rate generator for the SPI clock signal.
- The WCOL flag is not supported as a write collision error as described in the M68HC11 reference manual. Do not write to the transmit register when a SPI data transfer is in progress.
- Back-to-back transactions are supported — multiple, uninterrupted byte/half-word/word transfers can occur provided that the transmit FIFO never gets empty and the receive FIFO never gets full.
- All SPI transfers are full-duplex where an 8-bit data character is transferred from the master to the slave and an independent 8-bit data character is transferred from the slave to the master. This can be viewed as a circular 16-bit shift register in that an 8-bit shift register in the SPI master device is connected to another 8-bit shift register in a SPI slave device.

AXI4-Lite Interface Functionality in Standard SPI Multi-Master Configuration

The SPI bus to a given slave device (Nth device) consists of four wires:

- Serial clock (SCK)
- IO0 (Master out, slave in (MOSI))
- IO1 (Master in, slave out (MISO))
- Slave select (SS(N))

The signals SCK, IO0(MOSI), and IO1(MISO) are shared for all slaves and masters. See Figure 3-1.

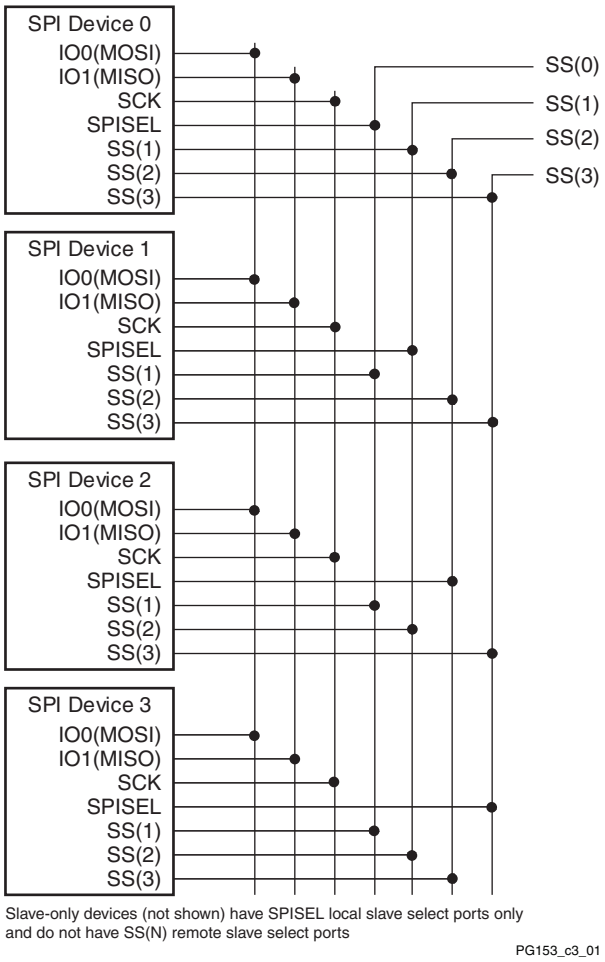

Figure 3-1: **Multi-Master Configuration Block Diagram for Standard SPI Mode**

Each master SPI device has the functionality to generate an active-Low, one-hot encoded $\overline{SS}$(N) vector where each bit is assigned an $\overline{SS}$ signal for each slave SPI device. It is possible for both SPI master/slave devices to be internal to the FPGA and SPI slave devices to be external to the FPGA. SPI pins are automatically generated through the Vivado® Design Suite when interfacing to an external SPI slave device. Multiple SPI master/slave devices are shown in Figure 3-1. The same configuration diagram is applicable for dual mode.

AXI4-Lite Interface Standard SPI Mode — Optional FIFOs in Legacy Mode

Only if the core is configured in standard SPI mode is there the flexibility of including optional FIFOs of either 16 or 256 deep in the design. Because the AXI Quad SPI is full-duplex, both transmit and receive FIFOs are instantiated as a pair and can be included in the core as shown in Figure 1-1, page 5.

When FIFOs are implemented, the slave select address is required to be the same for all data buffered in the FIFOs. This is necessary because there is no FIFO for the slave select address. Both transmit and receive FIFOs are 16 (or 256) elements deep and are accessed using single AXI transactions because burst mode is not supported.

The transmit FIFO is write-only. When data is written into the FIFO, the occupancy number is incremented and when a SPI transfer is completed, the number is decremented. As a consequence of this operation, aborted SPI transfers still have the data available for the transmission retry. The transfers can only be aborted in the master mode by setting the master transaction inhibit bit, bit 23 of the SPICR, to 1 during a transfer. Setting this bit in the slave mode has no effect on the operation of the slave. These aborted transfers are on the SPI interface. The occupancy number is contained in a read-only register.

If a write is attempted when the FIFO is full, an acknowledgement is given along with a generated error signal. Interrupts associated with the transmit FIFO include:

- Data transmit FIFO empty
- Transmit FIFO half empty
- Transmit FIFO under-run

See Interrupt Register Set Description in Chapter 2 for details.

The receive FIFO is read-only. When data is read from the FIFO, the occupancy number is decremented and when a SPI transfer is completed, the number is incremented. If a read is attempted when the FIFO is empty, acknowledgement is given along with a generated error signal. When the receive FIFO becomes full, the receive FIFO full interrupt is generated.

Data is automatically written to the FIFO from the SPI module shift register after the completion of a SPI transfer. If the receive FIFO is full and more data is received, a receive FIFO overflow interrupt is issued. When this occurs, all attempts to write data to the full receive FIFO by the SPI module are lost.

When the AXI Quad SPI core is configured with FIFOs, SPI transfers can be started in two different ways depending on when the enable bit in the SPICR is set. If the enable bit is set prior to the first data being loaded in the FIFO, the SPI transfer begins immediately after the write to the master transmit FIFO. If the FIFO is emptied using SPI transfers before additional elements are written to the transmit FIFO, an interrupt is asserted. When the AXI to SPI SCK frequency ratio is sufficiently small, this scenario is highly probable.

Alternatively, the FIFO can be loaded with up to 16 or 256 elements and then the enable bit can be set, which starts the SPI transfer. In this case, an interrupt is issued after all elements are transferred. In all cases, more data can be written to the transmit FIFOs to increase the number of elements transferred before emptying the FIFOs.

AXI4-Lite Interface Dual/Quad SPI Mode — Optional FIFO Depth

When the core is configured in dual or quad SPI mode, the FIFO is always present and there is the option to choose its depth. The depth of this FIFO can be either 16 or 256. The width of the FIFO in this mode is always 8 bits.

AXI4-Lite Interface SPI Master Loopback Operation

SPI master loopback operation, although not included in the M68HC11 reference manual, has been implemented to expedite testing. This operation is selected by setting the loopback bit in the SPICR (SPICR(0)) enabling an internal connection from the transmitter output to the receiver input. The receiver and transmitter operate normally, except that received data (from a remote slave) is ignored. This operation is relevant only when the SPI device is configured as a master and operated in standard SPI transaction mode. When the core is configured in dual or quad SPI mode, the loopback mode is not supported.

AXI4-Lite Interface Hardware Error Detection

The SPI architecture relies on software-controlled bus arbitration for multi-master configurations to avoid conflicts and errors. However, limited error detection is implemented in the SPI hardware. The first error detection mechanism to be discussed is contention error detection. This mechanism detects when a SPI device, configured as a master, is selected (that is, its SS bit is asserted) by another SPI device which is simultaneously configured as master. In this scenario, the master being selected as a slave immediately drives its outputs as necessary to avoid hardware damage due to simultaneous drive contention. The master also sets the mode-fault error (MODF) bit in the SPISR. This bit is automatically cleared by reading the SPISR. Following a MODF error, the master must be disabled and re-enabled with correct data. When configured with FIFOs, the process might require clearing the FIFOs.

A similar error detection mechanism has been implemented for SPI slave devices. The detected error occurs when a SPI device configured as a slave, but not enabled, is selected (that is, its SS bit is asserted) by another SPI device. When this condition is detected, IPISR bit 1 is set by a strobe to the IPISR register.

Underrun and overrun condition error detection is also provided. Underrun conditions can happen only when operated in slave mode. This condition occurs when a master commands a transfer, but the slave does not have data in the transmit register or FIFO to transfer. In this case, the slave underrun interrupt is asserted and the slave shift register is loaded with all zeros for transmission. An overrun condition can occur to both master and slave devices where a transfer occurs when the receive register or FIFO is full. During an overrun condition, the data received in that transfer is not registered (it is lost) and the IPISR overrun interrupt bit 5 is asserted.

Setting the Frequency Ratio Parameter

The AXI Quad SPI core is tested in hardware with SPI slave devices such as serial ATMEL, STMicro-Electronics, Winbond, and Intel flash memories in standard SPI mode (standard, dual and quad modes are tested with Winbond and Numonyx SPI flash memories). Read the data sheet of the targeted SPI slave flash memory or EEPROMs for the maximum speed of operation. Ensure that the correct values are used when deciding on the AXI clock and selecting the Frequency Ratio parameter. The AXI clock and the Frequency Ratio parameter determine the clock frequency at the SCK pin of the core. While using different external SPI slave devices, the Frequency Ratio parameter should be set carefully, and the maximum clock frequencies supported by all the external SPI slave devices should be taken into account.

AXI4-Lite Interface SPI Slave Mode — Standard SPI Configuration in Legacy Mode Only

The AXI Quad SPI core can be configured in slave mode by connecting the slave select line of the external master to SPISEL and by setting bit 2 of the SPI control register (SPICR) to 0. Slave mode is allowed only in standard SPI mode. In dual or quad SPI mode, the core only supports master mode.

All incoming signals are synchronized to the AXI clock when the Frequency Ratio parameter is greater than 4. Because of the tight timing requirements when Frequency Ratio parameter is set to 4, the incoming SCK clock signal and its synchronized signals are used directly in the internal logic. Therefore, the external clock must be synchronized with the AXI clock when Frequency Ratio parameter is 4. For other Frequency Ratio parameter values, it is preferred, but might not be necessary to have such synchronization.

RECOMMENDED: *In slave mode operation, use the FIFO by setting* ***FIFO Depth*** *to 16 or 256 in standard SPI mode.*

In slave mode, two interrupts are available in addition to the other interrupts in the IPISR:

- DRR_Not_Empty (bit 8)
- Slave_Mode_Select (bit 7)

Before the other SPI master starts communication, it is mandatory to fill the slave core transmit FIFO with the required data beats. When the master starts communication, with the core configured in slave mode, the core transfers data until the data exists in its transmit FIFO. At the end of last data beat transmitted from the slave FIFO, the core (in slave mode) generates the DTR empty signal to notify that new data beats need to be filled in its transmit FIFO before further communication can start.

Using the Enable STARTUPE2 Primitive Parameter

The Enable STARTUPE2 Primitive parameter is applicable for 7 series devices and if the core is in master SPI mode. When this parameter is included in the design, the STARTUPE2 primitive (for 7 series devices), is included in the design and becomes part of the core after configuration of the FPGA.

See the answer records or the device user guide to understand more about the functionality and use of the STARTUPE2 primitive.

Enable STARTUPE2 Primitive is Selected

Core Behavior and Ports

- The SCK_O port from the core is interfaced with the STARTUPE2 primitive. The STARTUPE2 can also be used in the pre-configuration process of the FPGA where the external SPI slave memory is configured prior to the FPGA.
- After configuration of the FPGA, the SCK_O port (output from the core) drives the USRCCLK0 port of the primitive. This signal is not available as the external port of the core.
- It is recommended to use lower frequency up to 80 MHz on `ext_spi_clk` when the STARTUPE2 block is used.

Enable STARTUPE2 Primitive is Not Selected

Core Behavior and Ports

- As the SCK_O and IO1_I ports are part of the core and no primitive in instantiated in the core, these ports are available as external ports of the core and they are placed in an IOB at a user-configured location.

Core Behavior in Legacy and Enhanced Non-XIP Mode

This mode is set whether or not **Enable Performance Mode** is selected and when it is, **Enable XIP Mode** is not selected. The AXI Quad SPI core supports Winbond and Numonyx memories. Check the commands if different memories must be tested with the core. If the commands, address, and data behavior are the same for a different memory, that device can be chosen as the base memory to test the core.

The core understands the commands and its expected behavior for the targeted memory through internal logic. The commands which are not supported by the Winbond or Numonyx memory device mentioned in the data sheet are marked with a command error. After the command error is set, the core does not execute the SPI transaction for that command and a command error interrupt is generated. After the command phase, if there is an address phase included, the next DTR contents are transferred on a SPI transaction in the modes defined by the address mode bits. If the data phase is present for the particular command, the data phase is executed based on read or write, with the modes set by the data mode bits.

The dummy bytes, which are required in some of the instructions for the selected memory, should be part of the SPI DTR along with the number of bytes intended to read from memory. For more information on the number of dummy bytes needed for a particular instruction, consult the data sheet for the targeted memory.

For read commands, after the transmission of the address bits, the core immediately reverts to input mode and starts storing data in the DRR. Therefore, be aware of how many dummy bytes are to be ignored in the DRR. For example, for the fast read dual output command in Winbond memory, the DTR should be filled with one command byte plus three address bytes plus two dummy bytes for the dummy cycles plus the number of dummy bytes to be read from memory. The command and address are transferred in standard SPI mode, after which the core reverts to the input mode and starts storing the data. The data is transferred on the IO0_I and IO_1 lines and stored in the SPI DRR, which includes the two dummy cycles plus valid data. Therefore, while reading the SPI DRR, ignore the first 6 bytes of the SPI DRR. The valid data available in the FIFO begins with the seventh byte. This also applies to other dual or quad read commands.

For each fresh transaction, the SPI DTR FIFO must be cleared. The first entry in the SPI DTR is always considered the command entry, which is cross-checked against built-in logic for the respective memory in the selected SPI mode.

Core Behavior in XIP Mode

When **Enable Performance Mode** and **Enable XIP Mode** are both selected, the core supports standard, dual and quad modes:

- Standard mode is set with **Mode** is **Standard** and **Slave Device** is **Winbond** or **Numonyx**.
- Dual mode is set with **Mode** is **Dual** and **Slave Device** is **Winbond** or **Numonyx**.
- Quad mode is set with **Mode** is **Quad** and **Slave Device** is **Winbond** or **Numonyx**.

Assumptions for this mode are:

Winbond Memory

Before executing the DIOFR (`0xBBh`) or QIOFR (`0xEBh`), the core executes the high performance mode command on each power-on reset state. This ensures that the Winbond memory is configured in high performance mode.

IMPORTANT: *When quad mode is set, the Winbond memory must be pre-configured by writing to the status register to set the QE bit to 1.*

It is your responsibility to pre-configure the memory. The core does not write anything to the status register. The core assumes that this exercise is completed previously in XIP mode.

When the core is configured in dual or quad mode prior to executing the DIOFR or QIOFR commands, the core performs the high performance command write to the memory on POR before accepting the transaction on the AXI4 interface. The HPM command requires one command and three dummy SPI cycles. This writing of the HPM command in memory is performed only at the power-on condition. The memory is now placed in high performance mode (HPM — `0xA3h`) and allows DIOFR or QIOFR to operate in their respective modes.

Numonyx Memory

In Numonyx memory, the volatile as well as nonvolatile configuration registers are left with the default configuration of dummy cycles. That is, VCR[7:4] and NVCR[15:12] are set to 1111. The core behavior is based on this assumption only and if these dummy cycle register values are changed, the core behavior is not guaranteed.

RECOMMENDED: *Do not change the default VCR and NVCR configuration.*

At the start of each new transaction, the core sends the respective command, address and required dummy cycles and then receives data.

Common Supported Commands for Dual SPI and Mixed Memory Mode

For the configuration with **Mode** set to **Dual** and **Slave Device** set to **Mixed**, the core supports the commands listed in Table 3-1, which are identical (in terms of command, address and data behavior) in both Winbond and Numonyx memory devices. See the memory data sheet for command details. The commands which are not supported in this mode include fast read, dual I/O fast read, and dual output fast read. These commands are not supported by the core in mixed mode because the dummy bytes or dummy cycles are different between the Winbond and Numonyx devices.

Also, in mixed mode, the volatile configuration register of the Numonyx device is not supported as this command is not present in the Winbond device. See Unsupported Commands for Dual/Quad SPI Mode and Winbond or Numonyx Memory for a list of unsupported commands.

Table 3-1: **Supported Commands in Dual SPI Mode and Mixed Memory Mode**

Opcode (Hex)	Command Description
01	Write status register
02	Page program
03	Read data bytes
04	Write disable
05	Read status register
06	Write enable
20	SubSector erase
4B	Read OTP (Read of OTP area)
75	Program/erase suspend
7A	Program/erase resume
9F	Read identification ID
C7	Bulk erase
D8	Sector erase

Common Supported Commands for Quad SPI and Mixed Memory Mode

For the configuration with **Mode** set to **Quad** and **Slave Device** set to **Mixed**, the core supports the commands listed in Table 3-2, which are identical (in terms of command, address and data behavior) in Winbond and Numonyx memory devices. See the memory data sheet for command details. The commands which are not supported in this mode include fast read, dual output fast read, quad output fast read, dual i/o fast read, and quad i/o fast read. These commands are not supported by the core in mixed mode because the dummy bytes or dummy cycles are different in Winbond and Numonyx devices. Also in mixed mode, the volatile configuration register of Numonyx devices is not supported as this command is not present in the Winbond device. See Unsupported Commands for Dual/Quad SPI Mode and Winbond or Numonyx Memory for a list of unsupported commands.

Table 3-2: **Supported Commands in Quad Mode and Mixed Mode Memory**

Opcode (Hex)	Command Description
01	Write status register
02	Page program
03	Read data bytes
04	Write disable

Table 3-2: **Supported Commands in Quad Mode and Mixed Mode Memory *(Cont'd)***

Opcode (Hex)	Command Description
05	Read status register
06	Write enable
20	Sector erase
32	Quad IP page program
C7	Bulk erase
75	Erase suspend
7A	Erase resume
4B	Read unique ID number
9F	Read JEDEC ID number
D8	Sector erase

XIP Mode commands

In XIP mode, the core supports three read commands:

- In standard mode: fast read - `0x0Bh`
- In dual mode: fast read dual I/O - `0xBBh`
- Quad mode: fast read quad I/O - `0xEBh`

Unsupported Commands for Dual/Quad SPI Mode and Winbond or Numonyx Memory

Winbond Memory (Ex: W25Q64VSFIG)

The core supports 24-bit addressing mode only.

Unsupported commands/features for this memory are:

- Fast read dual I/O continuous read mode.
- Fast read quad I/O continuous read mode.
- The command `ABh` is supported only for releasing the flash from power-down or high-performance mode. This command should not be used for reading the device ID.

Exceptional behavior for certain commands:

- Release power down/high-performance mode (`ABh`): This command supports release power-down/high-performance mode or reading the device ID with a different combination of dummy bytes. The core only supports release power-down/high performance-mode where only one command byte is required to be placed in the DTR.

The other mode of the command is not supported, as there is another command (90h) to read the device ID.

Numonyx Memory (Ex: N25Q256)

The core supports 24 and 32-bit addressing modes.

Unsupported commands/features for this memory are:

- XIP mode or continuous read mode in both the memories is not supported in Legacy or enhanced mode.
- All commands in dual or quad mode are supported in extended SPI mode. DIO and QIO modes are not supported.
- In quad mode, the design supports the Numonyx memory parts with HOLD functionality only. The parts with RESET functionality are not supported in the design.

Clocking

SPI Clock Phase and Polarity Control

Software can select any of four combinations of serial clock (SCK) phase and polarity with programmable bits in the SPICR. The clock polarity (CPOL) bit selects an active-High (clock idle state is Low) or active-Low clock (clock idle state is High). Determination of whether the edge of interest is the rising or falling edge depends on the idle state of the clock (that is, the CPOL setting).

The clock phase (CPHA) bit can be set to select one of two different transfer formats. If CPHA is 0, data is valid on the first SCK edge (rising or falling) after $\overline{SS}$(N) has been asserted. If CPHA is 1, data is valid on the second SCK edge (rising or falling) after $\overline{SS}$(N) has asserted. For successful transfers, the clock phase and polarity must be identical to those of the master SPI device and the selected slave device.

The first SCK cycle begins with a transition of the SCK signal from its idle state, which denotes the start of the data transfer. Because the clock transition from idle denotes the start of a transfer, the M68HC11 specification notes that the $\overline{SS}$(N) line can remain active-Low between successive transfers. The specification states that this format is useful in systems with a single master and single slave. In the context of the M68HC11 specification, transmit data is placed directly in the shift register on a write to the transmit register. Consequently, it is your responsibility to ensure that the data is properly loaded into the SPISSR register prior to the first SCK edge.

The $\overline{SS}$ signal is toggled for all CPHA configurations and there is no support for SPISEL being held Low. It is required that all $\overline{SS}$ signals be routed between SPI devices internally to the FPGA. Toggling the $\overline{SS}$ signal reduces FPGA resources.

Resets

The core supports active-Low reset input from the AXI interface in all configuration modes. The core also supports the internal reset register which generates a local reset signal to the core causing all registers to obtain default settings on each bit.

Protocol Description

For the SPI protocol description, see the *Motorola M68HC11-Rev. 4.0 Reference Manual*.

For the AXI protocol description, see the *AMBA AXI4-Stream Protocol Specification* [Ref 2].

AXI Quad SPI Core Behavior in Legacy and Enhanced Mode

When the Numonyx SPI memory is targeted as a slave, be aware of the different types of instruction sets supported. The Numonyx memory part N25Q_256_3 data sheet supports only extended SPI Instructions. Read the data sheet to verify the expected behavior of the core in different modes (set when **Mode** is **Dual** or **Quad** and **Slave Device** is set to **Numonyx**).

Mode = Dual

The core is configured to support dual and standard mode SPI commands. The configuration modes and their behavior are described in these sections:

Slave Device = Mixed

In this mode, it is assumed that there is more than one SPI slave device. As the core supports only Winbond and Numonyx memories, the slave must be one of these two devices.

This is considered to be a mixed memories mode where Winbond memories are taken as the base for defining the behavior of the core. The instructions which are common to Winbond and Numonyx memories (from the extended SPI protocol instructions set) are supported. Table 3-3 shows the core behavior.

Table 3-3: **SPI Command Core Behavior for Dual Mode and Mixed Mode Memories**

Command Type	Winbond	Numonyx	Command Error	Core Behavior
Standard SPI	Supported	Supported	No	Standard format
Standard SPI	Not supported	Supported	Yes	No SPI transaction

Table 3-3: **SPI Command Core Behavior for Dual Mode and Mixed Mode Memories *(Cont'd)***

Command Type	Winbond	Numonyx	Command Error	Core Behavior
Standard SPI	Supported	Not supported	No	Standard format
Standard SPI	Not supported	Not supported	Yes	No SPI transaction
Dual mode	Supported	Supported	No	Dual mode instruction format
Dual mode	Not supported	Supported	Yes	No SPI transaction
Dual mode	Supported	Not supported	No	Dual mode instruction format
Dual mode	Not supported	Not supported	Yes	No SPI transaction
Quad mode	Supported	Supported	Yes	No SPI transaction
Quad mode	Not supported	Supported	Yes	No SPI transaction
Quad mode	Supported	Not supported	Yes	No SPI transaction
Quad mode	Not supported	Not supported	Yes	No SPI transaction

Notes:

1. **Slave Device** = **Mixed** is mixed memory mode. For **Mode** = **Dual**, the dual as well as standard SPI commands are supported. For each command, the Winbond memory base behavior is used as the default operating mode. For the commands supported only by Numonyx, the command error flag is set and the command is not executed. In this mode, the command set in common with Winbond and Numonyx memories is supported.

Slave Device = Winbond

This is a dedicated mode and supports only Winbond memories as SPI slave devices. Most of the Winbond memory SPI commands are supported. Table 3-4 shows the core behavior.

Table 3-4: **SPI Command Core Behavior for Dual Mode and Winbond Memory**

Command Type	Winbond Memory	Command Error	Core Behavior
Standard SPI	Supported	No	Standard format
Standard SPI	Not supported	Yes	No SPI transaction
Dual mode	Supported	No	Dual mode instruction format as given in the data sheet
Dual mode	Not supported	Yes	No SPI transaction
Quad mode	Supported	Yes	No SPI transaction
Quad mode	Not supported	Yes	No SPI transaction

Notes:

1. The core is designed to support Winbond W25Q64BV memory. See the device data sheet for the command, address, dummy bytes and data bytes required for each command and for the command, address and data bits operating details.

Slave Device = Numonyx

This is a dedicated mode and supports only Numonyx memories as SPI slave devices. The core supports most of the dual and standard Numonyx memory SPI commands. Table 3-5 shows the core behavior.

Table 3-5: **SPI Command Core Behavior for Dual Mode and Numonyx Memory**

Command Type	Numonyx Memory	Command Error	Core Behavior
Standard SPI	Supported	No	Standard format
Standard SPI	Not supported	Yes	No SPI transaction
Dual mode	Supported	No	Dual mode instruction format as given in the data sheet
Dual mode	Not supported	Yes	No SPI transaction
Quad mode	Supported	Yes	No SPI transaction
Quad mode	Not supported	Yes	No SPI transaction

Notes:

1. The core is designed to support the Numonyx N25Q256-3V memory in this mode for all dual and standard mode commands. See the device data sheet for the command, address, dummy bytes and data bytes required for each command and for the command, address and data bits operating details.

Mode = Quad

The core is configured to support quad, dual and standard mode SPI commands based on the type of memory used as a SPI slave. The configuration modes and their behavior are described in these sections:

Slave Device = Mixed

In this mode, it is assumed that there is more than one SPI slave device. As the core supports only the Winbond and Numonyx memories, the slave must be one of these two devices.

This is a mixed memories mode where Winbond memories are taken as the base for defining the behavior of the core. Most of the instructions which are common to Winbond and Numonyx memories (Extended SPI commands) are supported. Table 3-6 shows the core behavior.

Table 3-6: **SPI Command Core Behavior for Quad Mode and Mixed Mode Memories**

Command Type	Winbond	Numonyx	Command Error	Core Behavior
Standard SPI	Supported	Supported	No	Standard format
Standard SPI	Not supported	Supported	Yes	No SPI transaction
Standard SPI	Supported	Not supported	No	Standard Format
Standard SPI	Not supported	Not supported	Yes	No SPI transaction
Dual mode	Supported	Supported	No	Dual mode instruction format
Dual mode	Not supported	Supported	Yes	No SPI transaction
Dual mode	Supported	Not supported	No	Dual mode instruction format
Dual mode	Not supported	Not supported	Yes	No SPI transaction
Quad mode	Supported	Supported	Yes	Quad mode instruction format

Table 3-6: **SPI Command Core Behavior for Quad Mode and Mixed Mode Memories *(Cont'd)***

Command Type	Winbond	Numonyx	Command Error	Core Behavior
Quad mode	Not supported	Supported	Yes	No SPI transaction
Quad mode	Supported	Not supported	Yes	Quad mode instruction format
Quad mode	Not supported	Not supported	Yes	No SPI transaction

Notes:

1. **Slave Device** = **Mixed** is mixed memory mode. In **Mode** = **Quad**, the quad, dual and standard SPI commands are supported. For each command, the Winbond memory base behavior is taken as the default operating mode. For commands supported only by Numonyx, the command error flag is set and the command is not executed. In this mode, the command set in common with Winbond and Numonyx memories is supported.

Slave Device = Winbond

This is a dedicated mode and supports only Winbond memories as SPI slave devices. Most of the Winbond W25Q64BV memories SPI commands are supported. Table 3-7 shows the generalized core behavior.

Table 3-7: **SPI Command Core Behavior for Dual Mode and Winbond Memory**

Command Type	Winbond	Command Error	Core Behavior
Standard SPI	Supported	No	Standard format
Standard SPI	Not supported	Yes	No SPI transaction
Dual mode	Supported	No	Dual mode instruction format as given in the data sheet
Dual mode	Not supported	Yes	No SPI transaction
Quad mode	Supported	No	Quad mode instruction format as given in the data sheet
Quad mode	Not supported	Yes	No SPI transaction

Notes:

1. The core is designed to support the Winbond W25Q64BV memory. See the device data sheet for the command, address, dummy bytes and data bytes required for each command and for the command, address and data bits operating details.

Slave Device = Numonyx

This is a dedicated mode and supports only Numonyx memories as SPI slave devices. The core supports most of the Numonyx N25Q256-3V memory quad, dual, and standard SPI commands. Table 3-8 shows the core behavior.

Table 3-8: **SPI Command Core Behavior for Dual Mode and Numonyx Memory**

Command Type	Numonyx	Command Error	Core Behavior
Standard SPI	Supported	No	Standard SPI format
Standard SPI	Not supported	Yes	No SPI transaction
Dual mode	Supported	No	Dual mode instruction format as given in the data sheet
Dual mode	Not supported	Yes	No SPI transaction
Quad mode	Supported	No	Quad mode instruction format as given in the data sheet
Quad mode	Not supported	Yes	No SPI transaction

Notes:

1. The core is designed to support the Numonyx N25Q256-3V memory device in this mode for all quad, dual and standard commands. See the device data sheet for the command, address, dummy bytes and data bytes requirements for each command and for the command, address and data bits operating details.

Core Behavior in XIP Mode

This mode is set when **Enable XIP Mode** is selected. This mode is especially useful when using the flash in ROM operations where the executable file is stored and accessed by the processor or any master.

In XIP mode, the core supports 24-bit addressing mode only. This 24-bit addressing is applicable to both Winbond and Numonyx memories with the core in Standard, Dual and Quad modes.

In this mode, the AXI4-Lite interface is first used to configure the core with the proper CPOL and CPHA modes. The valid modes are 00 and 11. Any other combination causes the core to not accept the AXI4 memory-mapped transaction. The AXI4-Lite interface is only used to set the configuration register and read the status register. The AXI4 memory-mapped interface is used to read the data from memory with the command configured using the AXI4-Lite interface and the address provided by the AXI4 memory-mapped interface. The read channel of the AXI4 memory-mapped interface should provide the starting address which is converted by the core to the SPI transactions at the SPI interface. The operating mode of the core is set using **Mode** while the targeted memory is selected based on the **Slave Device** setting. In this case, a single memory is targeted and multiple memories are not supported by the core. The target memory can be any from Winbond or Numonyx or any other memory which supports the default three read commands. The default commands are fast read (`0x0Bh`), fast read dual I/O (`0xBBh`) and fast read quad I/O (`0xEBh`). Based on the setting of **Mode**, the same command is used throughout the operation. The command cannot be changed as it is generated internally by the core.

The core has internal reference logic, which pads the dummy bytes required for the particular read command.

The XIP mode of core operation is based on the Winbond and Numonyx memory data sheet specification for read command behavior.

Parameters used for this configuration:

- Enable Performance Mode
- Enable XIP Mode
- Mode
- Slave Device
- Enable STARTUPE2 Primitive

The core uses the `ext_spi_clk` as the reference clock for the SPI logic. This clock is separate from the AXI4 interface clock and it should be double the desired SPI frequency at the SPI interface. The core uses a **Frequency Ratio** setting of 2, which is fixed for this mode, for generating the SPI clock at the SPI interface with reference to this clock. There is no soft reset register or interrupt register associated with XIP mode. The only way to reset the core is to reset the interconnect.

In this mode, the core supports WRAP as well as INCR type AXI4 transactions only. The FIXED transaction results in an error and the core does not accept the transaction. Instead, the transaction error flag is set in the XIP status register. In this case, the targeted memory is Winbond and if the core is configured in quad mode, ensure that the QE bit of the status register of the Winbond memory is set prior to the booting the core from SPI flash. This should be performed using external programming tools.

Standard SPI Mode Transactions

This section provides information on setting the SPI registers to initiate and complete bus transactions.

SPI Master Device with/without FIFOs and Slave Select Vector Asserted Manually Using SPICR Bit 24

This flow permits the transfer of N number of byte/half-word/word elements with a single toggling of the slave select vector. This is the default mode of operation. Use these steps to successfully complete the SPI transaction:

1. Start from a known state, including SPI bus arbitration.
2. Configure DGIER and IPIER registers as required.
3. Configure the target slave SPI device as required. This includes configuration of the DTR and control register of the slave SPI core as well as enabling it.

4. Write initial data to the master SPI DTR register/FIFO. This assumes that the SPI master is disabled.
   a. In Legacy mode, the AXI4-Lite transactions are written to the DTR one at a time.
   b. In Enhanced mode, the AXI4 interface must generate a FIXED burst transaction only. An INCR transaction with length 0 is acceptable but if the INCR burst is targeted at the FIFO locations (DTR or DRR), the core behavior is not guaranteed. The INCR transactions are treated as FIXED transactions. To avoid FIFO overflow or underflow errors, The transmit or receive occupancy register should be read before initiating a burst of any length.
5. Ensure the SPISSR register contains all ones.
6. Write configuration data to the master SPI device SPICR as required, including setting bit 7 for manual assertion of the $\overline{SS}$ vector and setting both enable and master transfer inhibit bits. This initializes SCK and IO0 but inhibits transfer.
7. Write to the SPISSR register to manually assert the $\overline{SS}$ vector.
8. Write the preceding configuration data to the master SPI device SPICR register, but clear the inhibit bit which starts the transfer.
9. Wait for an interrupt (typically IPISR bit 4) or poll status for completion. The wait time depends on the SPI clock ratio.
10. Set the master transaction inhibit bit to service the interrupt request.
11. Write new data to the master register/FIFOs and slave devices.
12. Clear the master transaction inhibit bit to continue the N 8-bit element transfer.

    Note: An overrun of the SPI DRR register/FIFO can occur if the SPI DRR register/FIFOs are not read properly. Also note that SCK has stretched the idle levels between element transfers (or groups of element transfers if using FIFOs) and that IO0 can transition at the end of an element transfer (or group of transfers), but will be stable for divide-by-2 clocking modes. IO0 will be valid at the falling edge of SCK, and for all other modes, it is two `ext_spi_clk` cycles after the falling edge of SCK.
13. Note that there are no idle cycles between each new SPI transaction.
14. Repeat step 10 through step 13 until all data is transferred.
15. Write all ones to the SPISSR register or exit the manual slave select assert mode to deassert the $\overline{SS}$ vector while SCK and IO0 are in the idle state.
16. Disable devices as required.

SPI Master and Slave Devices without FIFOs Performing One 8-bit/16-bit/ 32-bit Transfer (Optional Mode)

Use these steps to successfully complete a SPI transaction:

1. Start from a known state, including SPI bus arbitration.
2. Configure the master DGIER and IPIER registers. Also configure the slave DGIER and IPIER registers as required.
3. Write configuration data to the master SPI device SPICR register as required.
4. Write configuration data to the slave SPI device SPICR register as required.
5. Write the active-Low, one-hot encoded slave select address to the master SPISSR register.
6. Write data to the slave SPI DTR as required.
7. Write data to the master SPI DTR to start the transfer.
8. Wait for interrupt (typically IPISR bit 4) or poll status for completion.
9. Read the IPISR register of both master and slave SPI devices as required.
10. Perform interrupt requests as required.
11. Read the SPISR register of both master and slave SPI devices as required.
12. Perform actions as required or dictated by the SPISR register data.

SPI Master and Slave Devices Where Registers/FIFOs Are Filled Before the SPI Transfer Begins and Multiple Discrete 8-bit Transfers are Performed (Optional Mode)

The slave operation of the core supports a FIXED burst at the transmit or receive FIFO only. The length of this burst transaction should be based on the **FIFO Depth** parameter as well as the transmit or receive occupancy register. Take note of this to avoid any overrun or underrun errors of the DTR or DRR FIFO.

Use these steps to successfully complete a SPI transaction:

1. Start from proper state including SPI bus arbitration.
2. Configure the master DGIER and IPIER registers. Also configure the slave DGIER and IPIER registers as required.
3. Write configuration data to the master SPI device SPICR register as required.
4. Write configuration data to the slave SPI device SPICR register as required.
5. Write the active-Low, one-hot encoded slave select address to the master SPISSR register.

6. Write all data to the slave SPI DTR register/FIFO as required.
7. Write all data to the master SPI DTR register/FIFO.
8. Write the enable bit to the master SPICR register which starts the transfer.
9. Wait for interrupt (typically IPISR bit 4) or poll status for completion.
10. Read the IPISR register of both master and slave SPI devices as required.
11. Perform interrupt requests as required.
12. Read the SPISR register of both master and slave SPI devices as required.
13. Perform actions as required or dictated by SPISR register data.

SPI Master and Slave Devices with FIFOs Where Some Initial Data is Written to FIFOs, the SPI Transfer is Started, Data is Written to the FIFOs as Fast or Faster than the SPI Transfer and Multiple Discrete 8-bit Transfers are Performed (Optional Mode).

Use these steps to successfully complete a SPI transaction:

1. Start from the proper state including SPI bus arbitration.
2. Configure the master DGIER and IPIER registers. Also configure the slave DGIER and IPIER registers as required.
3. Write configuration data to the master SPI device SPICR register as required.
4. Write configuration data to the slave SPI device SPICR register as required.
5. Write the active-Low, one-hot encoded slave select address to the master SPISSR register.
6. Write initial data to the slave transmit FIFO as required.
7. Write initial data to the master transmit FIFO.
8. Write the enable bit to the master SPICR register which starts the transfer.
9. Continue writing data to both the master and slave FIFOs.
10. Wait for interrupt (typically IPISR bit 4) or poll status for completion.
11. Read the IPISR register of both master and slave SPI devices as required.
12. Perform interrupt requests as required.
13. Read the SPISR register of both master and slave SPI devices as required.
14. Perform actions as required or dictated by SPISR register data.

Dual/Quad SPI Mode Transactions

In this mode, the core must be configured in master mode only. Otherwise, an error interrupt is generated.

The sequence flow to operate the core in dual mode from the core point of view is shown. Check the inter-dependency of the commands before filling the SPI DTR register and enabling the SPI core to start the transaction.

1. Ensure that **Mode** is **Dual** and that **Slave Device** is set for the correct SPI slave memory.
2. Ensure that the instructions driven by the required SPI clock and set by **Frequency Ratio**, are listed.
3. Set the **FIFO Depth** parameter. This parameter can either be 16 or 256.
4. Write to the soft reset register to reset the core. This reset is active for 16 AXI cycles, during which each FIFO register is in the reset state.
5. Write to the SPICR register to put the core in master mode; set the CPOL, CPHA values, and make sure that the master transaction inhibit bit is set.
6. Write to the IPIER and IPISR registers to enable the required interrupts.
7. Write to the SPI DTR with the command, address, dummy, and data bytes to be transmitted in the same sequence provided in the data sheet of the target device.
8. Write to the SPI DTR with the number of data bytes intended to be read or written to memory along with command, address, and dummy bytes.
9. Write to the SPISSR register to assert the chip select signal from the core.
10. Write to the SPICR register to enable the master transaction inhibit bit, so that the core starts the SPI clock.
11. For a write, wait until the DTR empty interrupt is generated.
12. When the DTR empty interrupt is generated, data can still be written into the SPI DTR for further transactions if the slave select register remains unchanged (such as if the slave is selected).
13. If further data is written, the SPISSR register continues to be asserted; only the SPI clock is stopped. When the DTR FIFO is not empty, the SPI clock is enabled again and data is transmitted.
14. When reading, step 11, step 12 and step 13 also apply. Fill the DTR FIFO with any random data beats while reading the SPI slave memory.
15. After disabling the selected slave by setting all SPISSR register bits to 1 (or carrying out the master transactions inhibit bit set to 1 in the SPICR register), the SPI clock is stopped. If the SPI DTR FIFO is filled again, the core considers this a fresh transaction and compares the first entry in the DTR FIFO with the supported commands.

16. Between new transactions, make sure that the SPI DTR and DRR FIFOs are reset by writing into the SPICR register bits while the slave is deselected. This allows these two FIFOs to be reset and the first entry of the DTR FIFO to be compared with the supported commands.

Dual/Quad Mode SPI Configuration

In dual or quad mode SPI configuration, based on the type of memory (either Winbond or Numonyx), the SPI DTR FIFO must be filled prior to the transmission of SPI data beats. Reset both the SPI DTR and DRR FIFOs before filling the new transaction. The SPI DTR FIFO should be filled in command, address, dummy bytes and data order format only. This means that the first entry in the SPI DTR FIFO should always be the command followed by the address and then the data. The first entry in the SPI DTR FIFO is always compared with the internal command map table and, based on the supported command, the core behavior is determined. If the first entry in the SPI DTR FIFO does not match any commands in the supported command list, the core treats this as an error and SPI transactions do not proceed. An interrupt is also generated for this error.

Always check the supported and unsupported command list for the Winbond and Numonyx memories (unsupported commands for dual/quad SPI Mode and Winbond or Numonyx Memory). If unsupported commands are executed, the core behavior is not guaranteed. An interrupt is set to indicate a command error, which means that the command does not match any of the supported commands in the core.

Transfer Formats

In Standard SPI mode CPHA - CPOL, any one of the 00, 01, 10, or 11 modes are allowed.

In dual or quad SPI mode CPHA - CPOL, only 00 or 11 modes are allowed. If any other modes are set in the SPICR, an interrupt is set indicating the error and the core does not perform as expected.

CPHA Equals Zero Transfer Format

Figure 3-2 shows the timing diagram for a standard SPI mode data write-read cycle when CPHA = 0. The waveforms are shown for CPOL = 0, LSB First = 0, and the value of generic C_SCK_RATIO = 4. All AXI and SPI signals have the same relation regarding S_AXI_ACIk and SCK respectively.

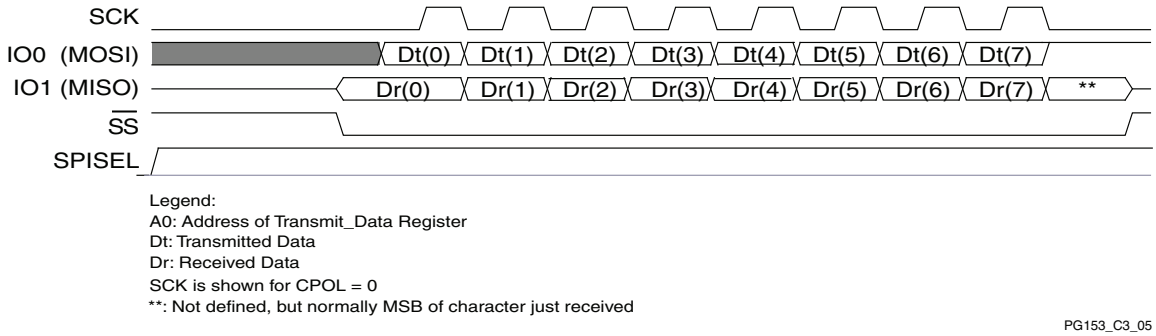

Figure 3-2: **Data Write-Read Cycle on SPI Bus with CPHA = 0 and SPICR(24) = 0 for 8-bit Data**

Signal SCK remains in the idle state until one-half period following the assertion of the slave select line which denotes the start of a transaction. Because assertion of the $\overline{SS}$(N) line denotes the start of a transfer, it must be deasserted and re-asserted for sequential element transfers to the same slave device.

One bit of data is transferred per SCK clock period. Data is shifted on one edge of SCK and is sampled on the opposite edge when the data is stable. Consistent with the M68HC11 SPI specification, selection of clock polarity and a choice of two different clocking protocols on an 8-bit/16-bit/32-bit oriented data transfer is possible using bits in the SPICR.

The IO0 pin is equivalent to IO0(MOSI) in standard SPI mode. The IO1 pin is equivalent to IO1(MISO) in standard SPI mode.

The IO0 and IO1 ports behave differently depending on whether the SPI device is configured as a master or a slave. When configured as a master, the IO0 port is a serial data output port, while the IO1 is a serial data input port. The opposite is true when the device is configured as a slave; the IO1 port is a slave serial data output port and the IO0 is a serial data input port. There can be only one master and one slave transmitting data at any given time. The bus architecture provides limited contention error detection (that is, multiple devices driving the shared IO1 and IO0 signals) and requires the software to provide arbitration to prevent possible contention errors.

All SCK, IO0, and IO1 pins of all devices are hard wired together. For all transactions, a single SPI device is configured as a master and all other SPI devices on the SPI bus are configured as slaves.

The single master drives the SCK and IO0 pins to the SCK and IO0 pins of the slaves. The uniquely-selected slave device drives data from its IO1 pin to the IO1 master pin, thus realizing full-duplex communication.

The Nth bit of the $\overline{SS}$(N) signal selects the Nth SPI slave with an active-Low signal. All other slave devices ignore both SCK and IO0 signals. In addition, the non-selected slaves (that is, $\overline{SS}$ pin High) drive their IO1 pin to 3-state so as not to interfere with SPI bus activities.

When external slave SPI devices are implemented, the SCK, IO0 and IO1, as well as the needed $\overline{SS}$(N) signals, are brought out to pins. All signals are true 3-state bus signals and erroneous external bus activity can corrupt internal transfers when both internal and external devices are present.

Ensure that the external pull-up or pull-down of external SPI 3-state signals are consistent with the sink/source capability of the FPGA I/O drivers. The I/O drivers can be configured for different drive strengths, as well as internal pull-ups. The 3-state signals for multiple external slaves can be implemented per the system design requirements, but the external bus must follow the SPI M68HC11 specifications.

CPHA Equals One Transfer Format

With CPHA = 1, the first SCK cycle begins with an edge on the SCK line from its inactive level to active level (rising or falling depending on CPOL) as shown in Figure 3-3. The waveforms are shown for CPOL = 0, LSB First = 0, and the value of generic C_SCK_RATIO = 4. All AXI and SPI signals have the same relation regarding S_AXI_Clk and SCK respectively.

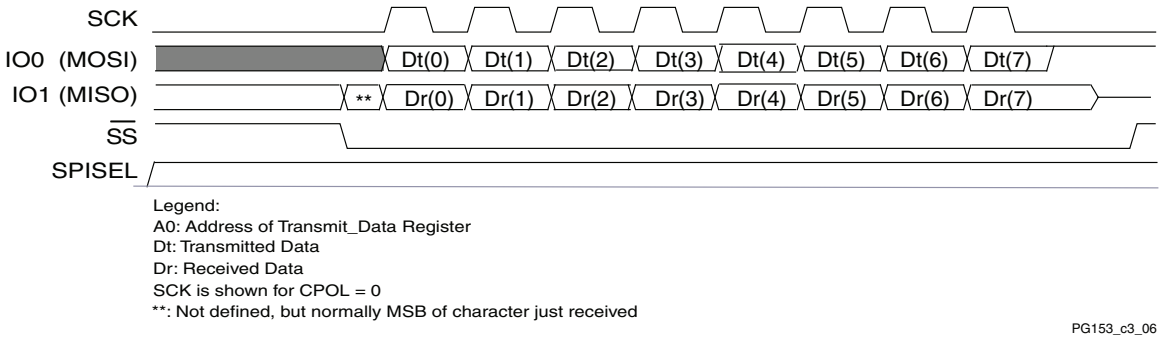

Figure 3-3: **Data Write-Read Cycle on SPI Bus with CPHA = 1 and SPICR(24) = 0 for 8-bit Data**

SPI Protocol Slave Select Assertion Modes

The standard mode SPI protocol is designed to have automatic slave select assertion and manual slave select assertion which are described in these sections. All the SPI transfer formats described in the SPI Clock Phase and Polarity Control section are valid for both automatic and manual slave select assertion mode.

SPI Protocol with Automatic Slave Select Assertion

This section describes the SPI protocol where slave select ($\overline{SS}$(N)) is asserted automatically by the SPI master device (SPICR bit 7 = 0).

This configuration mode is provided to permit transfer of data with automatic toggling of the slave select ($\overline{SS}$) signal until all elements are transferred. In this mode, the data in the SPISSR register appears on the $\overline{SS}$(N) output when the new transfer starts. After every beat of the SPI transaction (configured through **Transaction Width** in the Vivado Integrated Design Environment (IDE)), the $\overline{SS}$(N) output goes to 1. The data in the SPISSR register again appears on the $\overline{SS}$(N) output at the beginning of a new transfer. The slave select signal does not need to be manually controlled. This mode is not supported in dual or quad SPI modes in cores using the AXI4-Lite and AXI4 memory-mapped interfaces.

SPI Protocol with Manual Slave Select Assertion

This section briefly describes the SPI protocol where the slave select, $\overline{SS}$(N), is manually asserted by the user (that is, SPICR bit 7 = 1). This configuration mode is provided to permit transfers of an arbitrary number of elements without toggling slave select until all the elements are transferred. In this mode, the data in the SPISSR register appears directly on the $\overline{SS}$(N) output.

As described earlier, SCK must be stable before the assertion of slave select. Therefore, when manual slave select mode is used, the SPI master must be enabled first (SPICR bit 7 = 1) to put SCK in the idle state prior to asserting slave select.

The master transfer inhibit (SPICR bit 8) can be used to inhibit master transactions until the slave select is asserted manually and all FIFO data registers are initialized as required. This can be used before the first transaction and after any transaction that is allowed to complete.

When the preceding rules are followed, the timing is the same as presented for the automatic slave select assertion mode, with the exception that you control the assertion of the slave select signal and the number of elements transferred. While performing complete memory read or page read operations, the manual slave select mode should be used.

Beginning and Ending SPI Transfers

The details of the beginning and ending periods depend on the CPHA format selected and whether the SPI is configured as a master or a slave. These sections describe the beginning and ending period for SPI transfers.

Transfer Begin Period

The definition of the transfer beginning period for the AXI Quad SPI core is consistent with the M68HC11 reference manual. This manual can be referenced for more details. All SPI transfers are started and controlled by a master SPI device.

As a slave, the processor considers a transfer to begin with the first SCK edge or the falling edge of $\overline{SS}$, depending on the CPHA format selected. When CPHA equals zero, the falling edge of $\overline{SS}$ indicates the beginning of a transfer. When CPHA equals one, the first edge on the SCK indicates the start of the transfer. In either CPHA format, a transfer can be aborted by de-asserting the $\overline{SS}$(N) signal, which causes the SPI slave logic and bit counters to be reset. In this implementation, the software driver can deselect all slaves (that is, $\overline{SS}$(N) is driven High) to abort a transaction. Although the hardware is capable of changing slaves during the middle of a single or burst transfer, the software should be designed to prevent this.

In slave configuration, the data is transmitted from the SPI DTR register on the first AXI rising clock edge following $\overline{SS}$ signal being asserted. The data should be available in the register or FIFO. If data is not available, then the underrun interrupt is asserted.

Transfer End Period

The definition of the transfer end period for the AXI Quad SPI core is consistent with the M68HC11 reference manual. The SPI transfer is signaled complete when the SPIF flag is set. However, depending on the configuration of the SPI system, there might be additional tasks to be performed before the system can consider the transfer complete.

When configured without FIFOs, the Rx_Full bit (1) in the SPISR is set to denote the end of a transfer. When data is available in the SPI DRR register, bit 4 of the IPISR is asserted as well. The data in the SPI DRR is sampled on the same clock edge as the assertion of the SPI DRR register full interrupt.

When the SPI device is configured as a master without FIFOs, these steps occur:

- Rx_Empty bit (0), Tx_Full bit, and bit 3 in the SPISR are cleared.
- Tx_Empty bit (2), Rx_Full bit, and bit 1 in SPISR are set.
- DRR Full bit (4), Slave MODF bit, and bit 1 in the IPISR are set on the first rising AXI clock edge after the end of the last SCK cycle.

Note that the end of the last SCK cycle is a transition on SCK for CPHA = 0, but is not denoted by a transition on SCK for CPHA = 1 (see Figure 3-2 and Figure 3-3). However, the internal master clock provides this SCK edge which prompts the setting and clearing of the bits noted.

In this design, a counter is implemented that permits the simultaneous setting of SPISR and IPISR bits for both master and slave SPI devices. Note that external SPI slave devices can use an internal AXI clock that is asynchronous to the SCK clock. This can cause status bits in the SPISR and IPISR to be inconsistent with each other. Therefore, the AXI Quad SPI core cannot be used in a system with external SPI slave devices that do not use the AXI clock.

When the AXI Quad SPI core is configured with FIFOs and a series of consecutive SPI 8-bit/16-bit/32-bit element transfers are performed (based on parameter settings), the SPISR bits and IPISR do indicate completion of the first and the last SPI transfers with no indication of intermediate transfers. The only way to monitor when intermediate transfers are completed is to monitor the receive FIFO occupancy number. There is also an interrupt when the transmit FIFO is half empty, bit 6 of IPISR.

When the SPI device is configured as a slave, the setting/clearing of the bits discussed previously for a master coincides with the setting or clearing of the master bits for both cases of CPHA = 0 and CPHA = 1. Keep in mind that for CPHA = 1 (that is, no SCK edge denoting the end of the last clock period), the slave has no way of knowing when the end of the last SCK period occurs, unless an AXI clock period counter was included in the SPI slave device.

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the Vivado IP Integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3] for detailed information. IP Integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the Tcl Console.

Vivado Integrated Design Environment

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP, or select the **Customize IP** command from the toolbar or popup menu.

For details, see the sections, "Working with IP" and "Customizing IP for the Design" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1] and the "Working with the Vivado IDE" section in the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4].

Figure 4-1 shows the AXI Quad SPI Vivado Integrated Design Environment (IDE) screen.

Note: Figures in this chapter are illustrations of the Vivado IDE. This layout might vary from the current version.

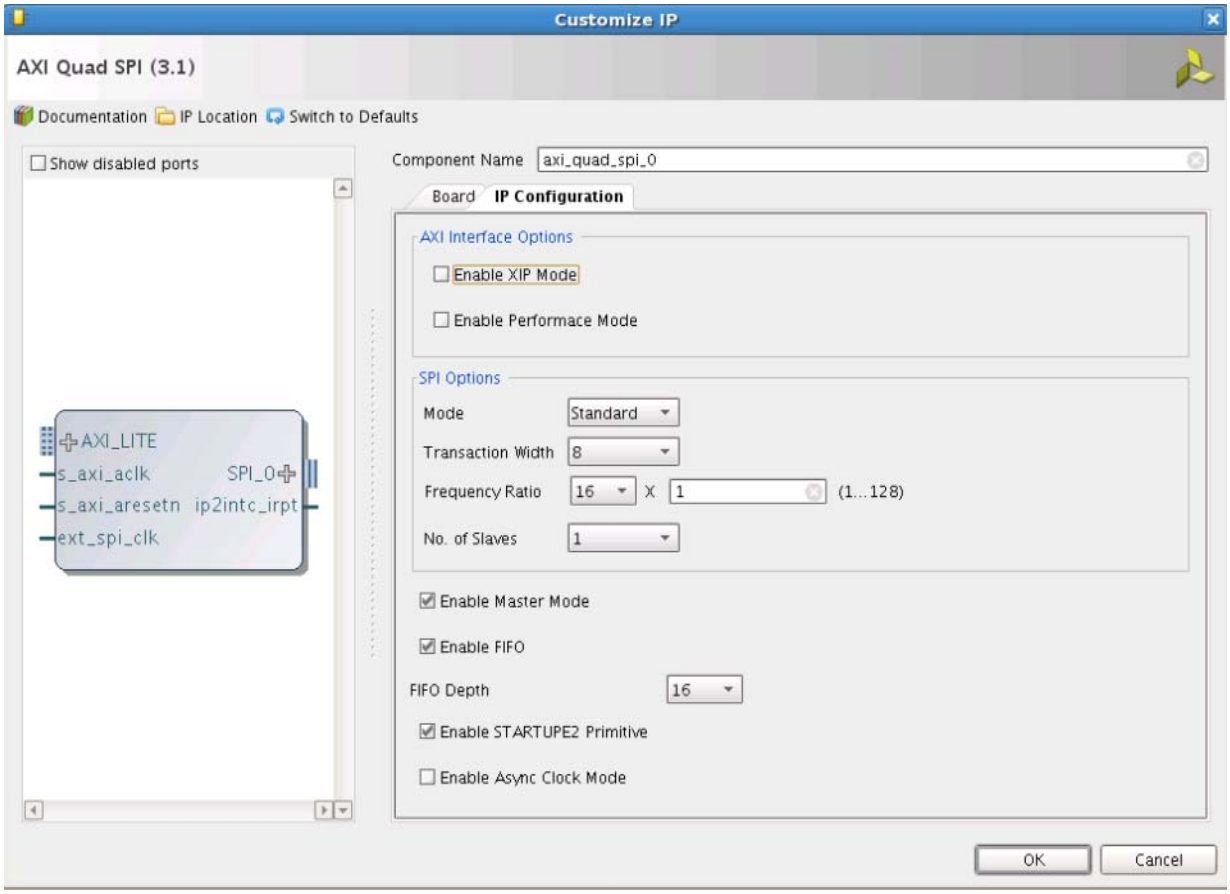

Figure 4-1: **AXI Quad SPI Vivado IDE Screen**

AXI Interface Options

- **Enable XIP Mode**: Enables the eXecute In Place (XIP) mode. This option also enables the AXI4 and AXI4-Lite interfaces. The choice of 24-bit or 32-bit addressing mode should be selected based on the downstream SPI device.
- **Enable Performance Mode**: Enables the AXI4 interface. Using the AXI4 memory-mapped interface also enables the burst capability at the transmit and receive FIFO addresses of the core. When this option is not selected, the AXI4-Lite interface is used.

SPI Options

- **Mode**: Selects standard, dual or quad mode. The correct mode is selected based on the targeted SPI slave device and application.
- **Transaction Width**: Selects 8, 16 or 32-bit transactions. Each SPI transaction incorporates the selected number of bits. In dual and quad SPI modes, the transaction width is restricted to 8 bits.

- **Frequency Ratio**: Selects a power of two divisor value from 2 to 2048. The resulting SPI clock frequency is the quotient of the frequency associated with the `ext_spi_clk` signal divided by the selected value. In dual or quad SPI mode, the divisor is restricted to 2.
- **No. of Slaves**: Selects the number on non-XIP mode SPI slave devices from 1 to 32. In XIP mode, the number of SPI slave devices is restricted to 1.
- **Slave Device**: Selects the dual or quad SPI mode slave device category from:
  - **Mixed**: Selects a command subset in common with both Winbond and Numonyx memory specifications.
  - **Winbond**: Selects a Winbond-specific memory command set.
  - **Numonyx**: Selects a Numonyx-specific memory command set.

  The slave device parameter is ignored in standard SPI mode.

Remaining Options

- **Enable Master Mode**: Enables master SPI mode when checked, slave SPI mode when not checked. The enable master mode parameter is applicable only in standard SPI mode. In dual or quad SPI mode, only master SPI mode is supported.
- **Enable FIFO**: Includes the transmit or receive FIFO in the design when checked, omits the FIFO when not checked. The enable FIFO parameter is applicable only in standard SPI mode. In dual or quad SPI mode, the FIFO is always included in the design.
- **FIFO Depth**: Selects the depth of the included FIFO from 0, 16 or 256 beats. In standard SPI mode, this parameter is only available when the FIFO is included. In dual or quad SPI mode, the FIFO depth is limited to either 16 or 256 beats. The FIFO width is fixed at eight bits.
- **Enable STARTUPE2 Primitive**: Includes the STARTUPE2 primitive in the design when checked, omits the primitive when not checked. The STARTUPE2 primitive is featured in 7 series devices and is useful in sharing the SPI clock with an external SPI slave device.
- **Enable Async Clock Mode**: Enables async clock mode when checked, disables when not checked. Enable this option only when the core is in standalone mode and the AXI interface and external SPI clocks differ in terms of phase/polarity and frequency. When enabled, the async clock mode results in the addition of XDC constraints to the design.

Output Generation

For details, see "Generating IP Output Products" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Constraining the Core

The necessary Xilinx design constraints (XDCs) are delivered when the core is generated.

Simulation

This chapter contains information about simulating IP in the Vivado® Design Suite.

- For comprehensive information about Vivado Design Suite simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5].
- For information about simulating the example design, see Simulating the Example Design.

Synthesis and Implementation

This chapter contains information about synthesizing and implementing IP in the Vivado® Design Suite.

- For details about synthesis and implementation, see "Synthesizing IP" and "Implementing IP" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].
- For information about synthesizing and implementing the example design, see Implementing the Example Design.

Example Design

This chapter contains information about the provided example design in the Vivado® Design Suite environment.

Overview

The top module instantiates all components of the core and example design that are needed to implement the design in hardware, as shown in Figure 1. This includes clock generator (MMCME2), Register configuration module, Read data checker and Slave.

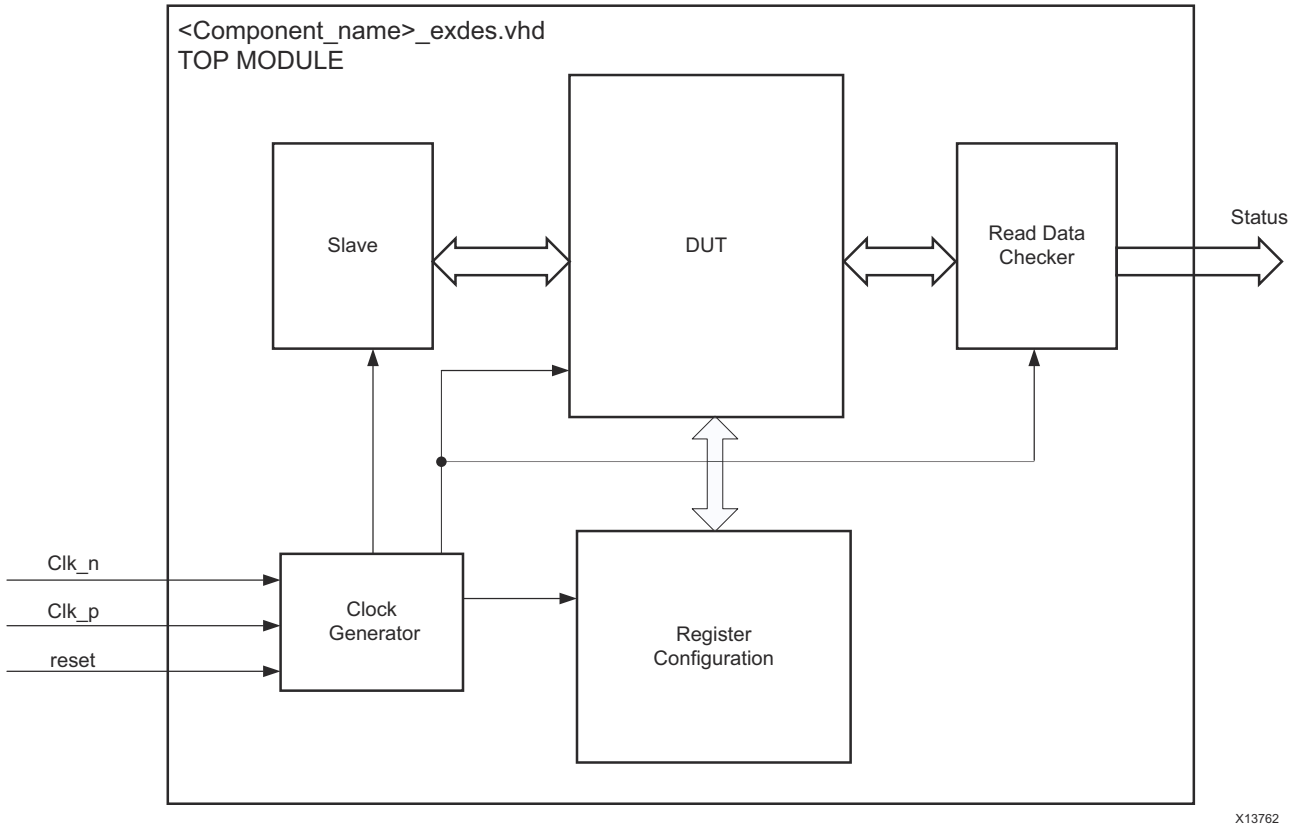

Figure 8-1: **Example Design Block Diagram**

The example design demonstrates transactions of AXI Quad SPI in its different configurations. It shows read and write transactions from AXI Quad SPI to the slave memory in different configured modes.

- **Clock generator:** The MMCME2 is used to generate the clock for the example design. It generates a 200 MHz clock for the DUT and other modules used in the example design. The DUT of the example design is kept under reset until MMCME2 is locked.
- **Register configuration module**: The AXI Traffic Generator is used for configuring the internal registers of the DUT and other modules (see Standard SPI Mode Transactions, page 78). After initial configuration of the registers sequence of data is sent to the AXI Quad SPI as AXI transactions, the AXI Quad SPI generates SPI transactions for the same data and writes into the Slave memory connected to it.
- **Read data checker**: The AXI Traffic Generator asks AXI Quad SPI to read the data from the connected slave memory by generating the commands, and verifies the correct data which is written previously is read.
- **Slave**: Memory models delivered mimic the behavior of Winbond and Numonyx flashes, and are used as slave memory to AXI Quad SPI when it is configured in master mode.

In slave mode the AXI Quad SPI core is instantiated itself as a master internally to show the slave behavior of the core.

IMPORTANT: *The example design is not supported when the start-up block is enabled.*

Implementing the Example Design

After following the steps described in Chapter 4, Customizing and Generating the Core, implement the example design as follows:

1. Right-click the core in the Hierarchy window, and select **Open IP Example Design**.
2. A new window pops up where you can specify a new directory name for the example design, or keep the default directory.

   A new project is created in the selected directory and is opened in a new Vivado window.

3. In the Flow Navigator (left side pane), click **Run Implementation** and follow the directions.

Example Design Directory Structure

In the current project directory, a new project called `<component_name>_example` is created and the files are delivered in the `<component_name>_example/<component_name>_example.srcs/` directory. This directory and its subdirectories contain all the source files that are required to create the AXI Quad SPI example design.

Table 8-1 shows the design files generated. They are delivered in the `<component_name>_example/<component_name>_example.srcs/sources_1/imports/<component_name>/<component_name>/example_design/` directory.

Table 8-1: **Example Design Directory**

Name	Description
<component_name>_exdes.vhd	Top-level HDL file for the example design.
memory.vhd	Memory model used in XIP mode.
memory_model.vhd	Memory model used in other modes.

Table 8-2 shows the COE files generated for data transmission. They are delivered in the `<component_name>_example/<component_name>_example.srcs/sources_1/imports/<component_name>/` directory.

Table 8-2: **COE Design Directory**

Name	Description
qspi_addr_*.coe	Delivers address information to the AXI Traffic Generator.
qspi_data_*.coe	Delivers data information to the AXI Traffic Generator.
qspi_ctrl_*.coe	Delivers control information to the AXI Traffic Generator.
qspi_mask_*.coe	Delivers maskl information to the AXI Traffic Generator.
init_data_coe	Initialization data to BMG.

Notes:

1. The range from 1 to 3 as each file corresponds to a particular AXI Traffic Generator.

Table 8-3 shows the files delivered in the `<component_name>_example/<component_name>_example.srcs/sources_1/ sim_1/imports/component_name>/component_name>/example_design/` directory.

Table 8-3: **Simulation Directory**

Name	Description
<component_name>_exdes_tb.vhd	Test bench for Exdes.

Table 8-4 shows the files delivered in the `<component_name>_example/<component_name>_example.srcs/sources_1/ constrs_1/imports/example_design/` directory.

Table 8-4: **Constraints Directory**

Name	Description
exdes.xdc	Top-level constraints file for the example design.

The example design has been verified on KC705 boards. Board constraints are also specified in the `exdes.xdc` file but are commented out by default.

IMPORTANT: *Uncomment the pin constraints while testing on the board.*

Testing the Example Design on a KC705 Board

Follow these steps to test the example design on a KC705 board:

1. Configure the core using the Vivado Integrated Design Environment (IDE) in standalone mode.
2. Implement the example design.
3. Generate the bitstream by selecting the implementation and generate bitstream option.

The status of the transaction, such as Done, is displayed on F16, while the E18 and G19 pins indicate the AXI Traffic Generator core status. For more reference about these pins, refer to the `exdes_xdc.ttcl` file. It might be necessary to use the `exdes_xdc.ttcl` file after you un-comment the LOC constraints. For more information, see Checking Results, page 101.

With the external memories targeted as slave SPI, the example design is not supported on the board, but you can test the bitstream with the memory model on a given FPGA.

You must set the parameter combinations for checking the core behavior through Vivado IDE options.

The built-in memory model is used as target slave. The memory model and core are configured with the predefined command, address, and data. When the example design simulations are enabled, both the model and the core exchange data.

Legacy Mode and Performance Mode Example Design Behavior

For Standard SPI mode, Dual and Quad SPI mode, the core and the memory model have predefined commands. The standard page program command (0x03h) is used for writing the data in the memory model when the FIFO is enabled in the design. In case of read from memory model, the commands are chosen based on the SPI mode of the core.

XIP Mode Example Design Behavior

In the XIP mode, the core has built-in commands as per the mode chosen. The memory model is pre-configured with the data and the same data is read by the core when a transaction is initiated.

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite.

Overview

Figure 9-1 shows test bench for AXI Quad SPI example design. The top-level test bench generates 200 MHz clock and drives initial reset to the example design.

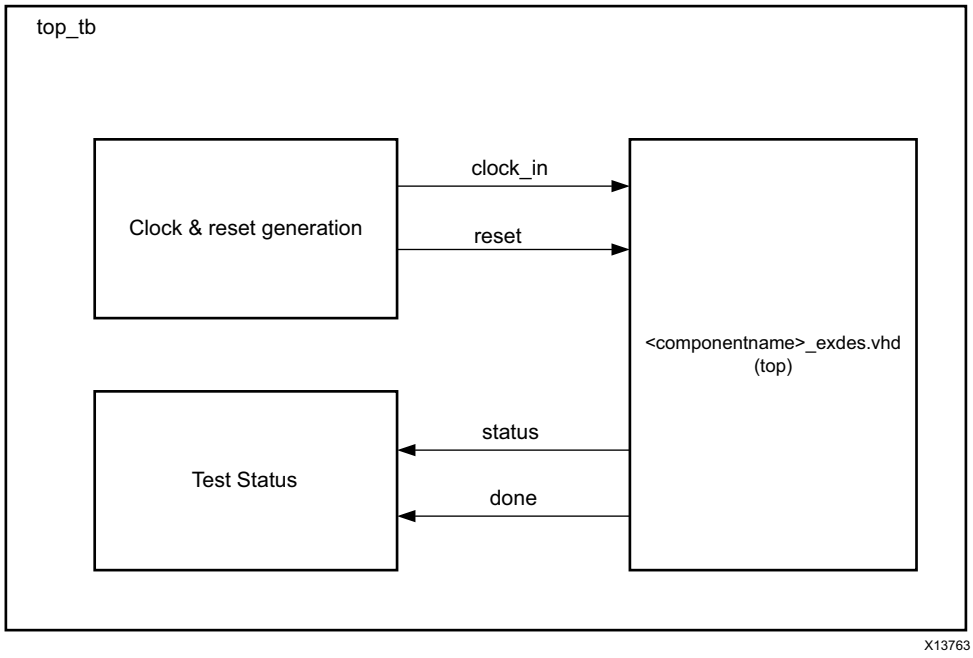

Figure 9-1: **AXI Quad SPI Example Design Test Bench**

Checking Results

The signal status is used to convey information about the test case results.

- Status (1 downto 0) indicates whether the test passes, failed or hanged.

  01 - Test completed successfully.

  10 - Test failed.

  11 - Test hanged.

- Status (9 downto 2) gives the index of MIF where the test stopped.

Note: The `Done` signal goes High after the test is complete.

Simulating the Example Design

Using the example design delivered as part of the AXI Quad SPI, you can quickly simulate and observe the behavior of the AXI Quad SPI.

Setting up the Simulation

The Xilinx simulation libraries must be mapped to the simulator. To set up the Xilinx simulation models, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5]. To switch simulators, click **Simulation Settings** in the Flow Navigator (left pane). In the Simulation options list, change **Target Simulator**.

The example design supports functional (behavioral) and post-synthesis simulations. For information how to run simulation, see the *Vivado Design Suite User Guide: Logic Simulation (UG900)* [Ref 5].

Simulation Results

The simulation script compiles the AXI Quad SPI example design, and supporting simulation files. It then runs the simulation and checks that it completed successfully.

If the test passes, following message is displayed:

```
Test Completed Successfully
```

If the test hangs, the following message is displayed.

```
Test Failed !! Test Timed Out.
```

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® Design Suite to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information on migrating to the Vivado Design Suite, see *ISE to Vivado Design Suite Migration Methodology Guide* (UG911) [Ref 6].

Upgrading in the Vivado Design Suite

There are no port or parameter changes. However, the SPISEL port is hidden when master SPI mode is selected. This port is internally driven to VCC in master mode. In case of slave SPI operation mode, this port is available and should be connected to the Slave Select port of other SPI masters.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI Quad SPI, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the AXI Quad SPI. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core are listed here, and can also be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool messages
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the AXI Quad SPI

AR: 54408

Contacting Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Additional Resources

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which files to include with the WebCase.

Note: Access to WebCase is not available in all cases. Login to the WebCase tool to see your specific support options.

Debug Tools

There are many tools available to address AXI Quad SPI design issues. It is important to know which tools are useful for debugging various situations.

Vivado Lab Tools

Vivado® lab tools insert logic analyzer (ILA) and virtual I/O (VIO) cores directly into your design. Vivado lab tools allow setting trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature represents the functionality in the Vivado IDE that is used for logic debugging and validation of a design running in Xilinx devices in hardware.

The Vivado logic analyzer is used to interact with the logic debug of LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See *Vivado Design Suite User Guide: Programming and Debugging*(UG908) [Ref 7].

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado lab tools are a valuable resource to use in hardware debug.

1. Use the Vivado logic analyzer for hardware debug at SPI interface.
2. Attach the logic analyzer ports on all SPI interface ports like `SS`, `SCK`, `IO0`, and `IO1` (also, `IO2`, and `IO3` if used in Quad mode).
3. Generate the transactions at the AXI interface, and observe whether or not the Slave Select line is asserted, and whether or not the SPI clock is toggled.
4. Observe whether or not the data from the core provides sufficient set up and hold time.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output `s_axi_arready` asserts when the read address is valid, and output `s_axi_rvalid` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `s_axi_aclk` and `aclk` inputs are connected and toggling.
- The interface is not being held in reset, and `s_axi_areset` is an active-Low reset.
- The interface is enabled, and `s_axi_aclken` is active-High (if used).
- The main core clocks are toggling, and the enables are also asserted.
- If the simulation has been run, verify in simulation and/or a Vivado lab tools capture that the waveform is correct for accessing the AXI4-Lite interface.
- At a given time, the core will accept either AXI read or AXI write transaction.

- Each read or write AXI transaction can be observed on IPIF interface with `bus2ip_*` signals. For read transaction, please observe the particular signal lane selected by `bus2ip_rdce` vectored port.
- For each write transaction, observe the particular signal lane selected by the `bus2ip_wrce` vectored port, and the AXI data available on the `bus2ip_data` port.
- Each read or write transaction is completed by the core when it generates the `ip2bus_rdack` or `ip2bus_wrack`.
- In case of any error transaction, `ip2bus_errorack` is generated along with a read or write `ack`.
- These IPIF transactions are further completed by the core at the AXI interface after you enable the write or read `ready` signals.

AXI4 Memory Mapped Interfaces

The AXI Quad SPI core supports AXI4 Memory Mapped interface in Enhanced mode and in XIP mode.

- In Enhanced mode, only FIXED AXI4 transactions are allowed for Data Transmit and Data Receive FIFO locations. Any other AXI4 transaction is not allowed and core behavior is not guaranteed.
- In XIP mode, all AXI4 transactions are allowed. It is recommended that you use 32-bit size AXI4 transactions. The following points cover further debug information.

Enhanced Mode Debug

1. Make sure the clock is connected to the `s_axi4_aclk` port.
2. Make sure the active-Low `reset` signal is connected to the `s_axi4_aresetn` port. The reset should be de-activated to make sure the core is in working condition.
3. All the registers should be accessed with a single AXI transaction and at word boundary.
4. The DTR and DRR FIFO can be accessed using a single transaction or a burst FIXED transaction.
5. The AXI4 transactions are converted in the `bus2ip_*` signal transactions.
6. For AXI4 read transactions, check that the corresponding signal (for registers) for Bus2IP_RdCE is enabled. The core is responding with the correct register data on IP2Bus_Data and IP2bus_RdAck signal. This completes the AXI4 read transaction
7. For AXI4 write transactions, check if the corresponding signal (for registers) for Bus2IP_WrCE is enabled. The Bus2IP_Data carries the AXI4 write data. The core completes the transaction by updating the register bits with IP2Bus_WrAck.

8. For each SPI transaction, it is necessary to have data beats are available in DTR FIFO. The SPI transactions are enabled only when the SPICR register is configured, and the slave register should be updated to select a particular slave. The SPI transactions can be observed on SPI interface like Slave Select, SPI clock, IO0 and IO1.
9. The Slave Select line must be asserted for any SPI transaction.

XIP Mode Debug

1. Make sure all the clocks and resets are connected to proper AXI interface.
2. This mode of the core supports all AXI4 transactions like FIXED, WRAP and INCR. Use a 32-bit AXI transaction for the best results. In this mode, the core operates in read-only mode.
3. Based on the number of address bits supported by the downstream device, select 24-bit or 32-bit addressing mode in the Vivado Integrated Design Environment (IDE). Based on this choice, the core considers either 24 or 32 bit from AXI4 address transactions.
4. Based on the choice of SPI mode, the core chooses the suitable SPI to read commands and add dummy cycles before accepting the correct data bits from memory.
5. The SPI transactions are observed on the SPI interface where the Slave Select line is asserted, and the SPI clock is generated by the core.
6. After the data is available in the core, the core supplies the data to an AXI4 read channel through an internal FIFO.

Additional Resources

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm

References

These documents provide supplemental material useful with this product guide:

1. *Vivado Design Suite User Guide: Designing with IP* (UG896)
2. *AMBA AXI4-Stream Protocol Specification* (ARM IHI 0051A)
3. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
4. *Vivado Design Suite User Guide: Getting Started* (UG910)
5. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
6. *ISE to Vivado Design Suite Migration Methodology Guide* (UG911)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
8. *Motorola M68HC11-Rev. 4.0 Reference Manual*
9. *Motorola MPC8260 PowerQUICC II™ Users Manual 4/1999 Rev. 0*
10. *LogiCORE IP AXI4-Lite IPIF* (PG155)
11. *LogiCORE IP AXI Interconnect* (PG059)
12. *Winbond memory data sheet* (W25Q64BV)
13. *Numonyx memory data sheet* (N25Q256-3v)

14. *7 Series FPGAs Overview* (DS180)
15. *7 Series FPGAs Configuration User Guide* (UG470)
16. *Execute-in-Place (XIP) with AXI Quad SPI Using Vivado IP Integrator Application Note* (XAPP1176)
17. *Throughput Performance Measurement Application Note* (XAPP797)
18. *Vivado Design Suite User Guide: Implementation* (UG904)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/20/2013	1.0	Initial release in product guide format. This document replaces DS843. • Moved Core Mode and Parameter Value table to Chapter 1. • Revised I/O signal widths where affected. • Revised Resets section in Chapter 3. • Added 24-bit addressing restriction note to XIP mode section of Chapter 3. • Added IDE screen documentation and removed obsolete Design Parameter section in Chapter 4.
10/02/2013	3.1	• Updated for the core v3.1. Version in this table aligns to core version. • Added 32-bit address support for Numonyx Memories in XIP mode. • Added the core design example, and test bench information. • Added information related to simulation, synthesis and implementation, and Vivado IP integrator support. • Added hardware debug and interface debug information.
12/18/2013	3.1	Added UltraScale™ architecture support.

Notice of Disclaimer