XILINX®

LogiCORE IP AXI Quad Serial Peripheral Interface (AXI Quad SPI) v2.00a

DS843 December 18, 2012

Product Specification

Introduction

The AXI Quad SPI connects the AXI4 interface to those SPI slave devices that support Standard, Dual or Quad SPI protocol instruction set. This 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) 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 AXI4 for XIP mode of operation
- Enhanced non-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
- Four signal interfaces:
  - IO0 (MOSI) - Standard SPI mode
  - IO1 (MISO) - Standard SPI mode
  - SCK - Standard and Dual SPI mode
  - $\overline{\text{SS}}$ - Standard and Dual SPI mode
- In Quad SPI mode supports six signal interfaces: IO0, IO1, IO2, IO3, SCK and $\overline{\text{SS}}$
- Configurable Slave Select ($\overline{\text{SS}}$) lines on the SPI bus
- 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	
Core Specifics	
Supported Device Family[1]	Zynq-7000[2], Virtex-7[3], Kintex-7[3], Artix-7 [3], Virtex-6[4], Spartan-6[5]
Supported User Interfaces	AXI4-Lite, AXI4 Full
Resources	See Tables 29 to 34.
Provided with Core	
Design Files	ISE: VHDL Vivado: RTL
Example Design	N/A
Test Bench	N/A
Constraints File	N/A
Simulation Model	N/A
Supported S/W Driver[6]	Standalone and Linux
Tested Design Tools[7]	
Design Entry Tools	Xilinx Platform Studio (XPS) 14.4 Vivado Design Suite 2012.4[8]
Simulation	Mentor Graphics ModelSim
Synthesis Tools	Xilinx Synthesis Technology (XST) Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete list of supported derivative devices, see the Embedded Edition Derivative Device Support.
2. Supported in ISE Design Suite implementations only.
3. For more information on 7 series devices, see [Ref 8].
4. For more information on Virtex-6 devices, see [Ref 9].
5. For more information on Spartan-6 devices, see [Ref 10].
6. Standalone driver details can be found in the EDK or SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from wiki.xilinx.com.
7. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.
8. Supports only 7 series devices.

Supported Features Detail

- 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

Unsupported Features

The following features relate to the AXI4 memory mapped interface. These features are not supported by the core.

- INCR of length more than 1 and WRAP bursts are not supported in Enhanced mode
- FIXED burst is not supported in XIP mode
- Narrow bursts in Enhanced mode are not supported; only the last 8 bits from the 32 burst bits are valid in Enhanced mode.
- Write Channel and transactions are not supported 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 are not supported in Enhanced mode
- Un-aligned address when the core is configured in Read only XIP mode
- Byte access in XIP mode

Functional Description

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

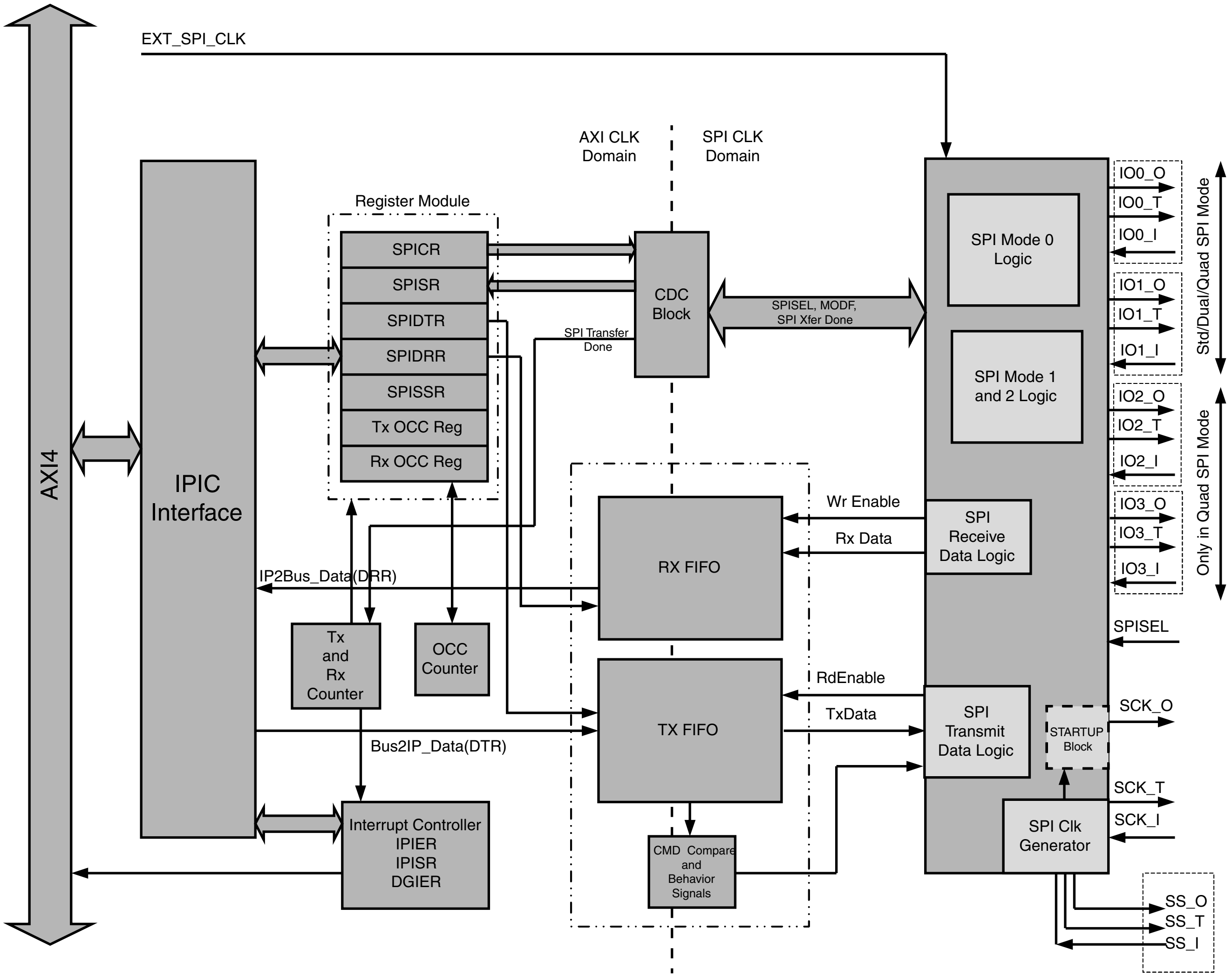


Figure 1: **AXI Quad SPI Core Top-Level Block Diagram**

The choice of either AXI4-Lite or AXI4 memory mapped interface is based on the parameter C_TYPE_OF_AXI4_INTERFACE. The default value of this parameter is 0 which selects the AXI4-Lite interface. For the selected AXI4 interface, the core always works as a slave in the AXI4 interface.

Legacy Mode

Legacy mode is selected by setting the parameter C_TYPE_OF_AXI4_INTERFACE to 0. It uses the AXI4-Lite interface. This mode of operation is fully compatible with the AXI Quad SPI v1.00a 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 you to manually control the slave select line by 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, you must toggle the slave select line.

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.

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 C_SPI_MODE is set to 0. The relevant parameters in this mode are:

- C_SPI_MODE
- C_SPI_MEMORY
- C_USE_STARTUP
- C_NUM_TRANSFER_BITS
- C_NUM_SS_BITS
- C_SCK_RATIO
- C_FIFO_DEPTH

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

1. The choice of inclusion of the FIFO is based on the C_FIFO_DEPTH parameter, which if included in the design, the transmit and receive FIFO depth is limited to 16 or 256. It is recommended that a FIFO depth of 256 is used, because this is the most suitable depth to co-relate with the flash memory page size.
2. The valid values for the C_FIFO_DEPTH parameter in this mode are 0 or 16 or 256.

When C_FIFO_DEPTH = 0, no FIFO is included in the core. Data transmission occurs through the single transmit and receive register. When C_FIFO_DEPTH = 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 through the C_NUM_TRANSFER_BITS parameter.

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

When the core is configured as a slave, if the slave select line (SPISEL) goes High (inactive state) inadvertently in between 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 C_SPI_MODE is set to 1. The relevant parameters in this mode are:

- C_SPI_MODE
- C_SPI_MEMORY
- C_USE_STARTUP
- C_NUM_TRANSFER_BITS
- C_NUM_SS_BITS
- C_FIFO_DEPTH

The properties associated with the FIFO are:

- The depth of the FIFO is based on the C_FIFO_DEPTH parameter., 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 8 bit only

The behavior of the ports in Dual mode is:

- For Standard SPI mode instruction, the IO0 and IO1 pins are unidirectional (the same as the MOSI and MISO pins).
- For Dual mode SPI instruction, the IO0 and IO1 pins are bidirectional - depending on the type of memory chosen and type of command.

The Quad SPI mode is selected when C_SPI_MODE is set to 2. The behavior of the ports in the Quad mode is:

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

The parameter C_SPI_MODE = 1 or 2 forces the core to operate in Dual or Quad SPI mode, 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 C_SPI_MODE parameter settings also decide 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 communicates 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 word access to all SPI and interrupt register modules.

AXI4 Memory Mapped Interface

The AXI4 memory mapped interface is selected when C_TYPE_OF_AXI4_INTERFACE is 1. In this mode, the core can be operated in Enhanced mode (C_XIP_MODE = 0) or XIP mode (C_XIP_MODE = 1). In this 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 Full interface. This mode also supports Standard, Dual and Quad modes using the C_SPI_MODE parameter. The target slave memory can be chosen by setting the C_SPI_MEMORY to 0 or 1 or 2. All the registers are mapped at the same offset which are mapped for 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 accessed as single access only. This should be noted while designing the application for the core.

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

XIP Mode

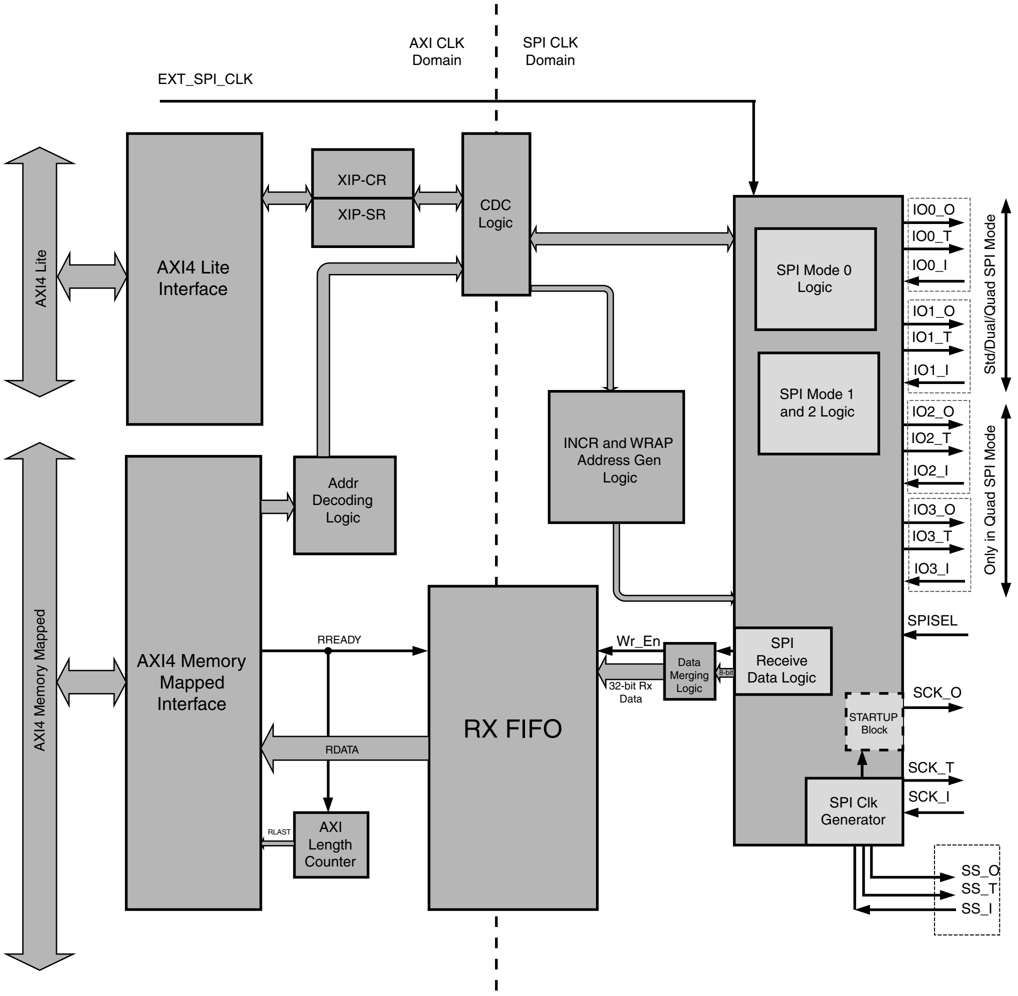


Figure 2: **Block Diagram of AXI Quad SPI in XIP Mode**

In XIP mode, the core has an AXI4-Lite as well as an AXI4 memory mapped interface (Figure 2). The AXI4-Lite interface is chosen for accessing the configuration register and the status register. The AXI4 full interface is used only for reading. The AXI4 full 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) - Clock Phase (CPHA) configuration. 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 the AXI4-Lite as well as AXI4 memory mapped interface. The three main read commands which are in built into the core are Fast Read (0x0Bh), DIOFR (0xBBh) and QIOFR (0xEBh).

Core Internal Submodules

The AXI Quad SPI IP core submodules are described in the following sections.

C_TYPE_OF_AXI4_INTERFACE is 0

In this mode, the core is backward compatible with v1.00a. The core includes the following 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 memory mapped registers (Figure 1). It interfaces to the AXI4-Lite interface. It 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 interrupt related registers, namely, the device global interrupt enable register (DGIER), IP interrupt enable register (IPIER), and 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 C_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 generic C_NUM_TRANSFER_BITS. 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

The STARTUP 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 setting the C_USE_STARTUP parameter to 1. 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.

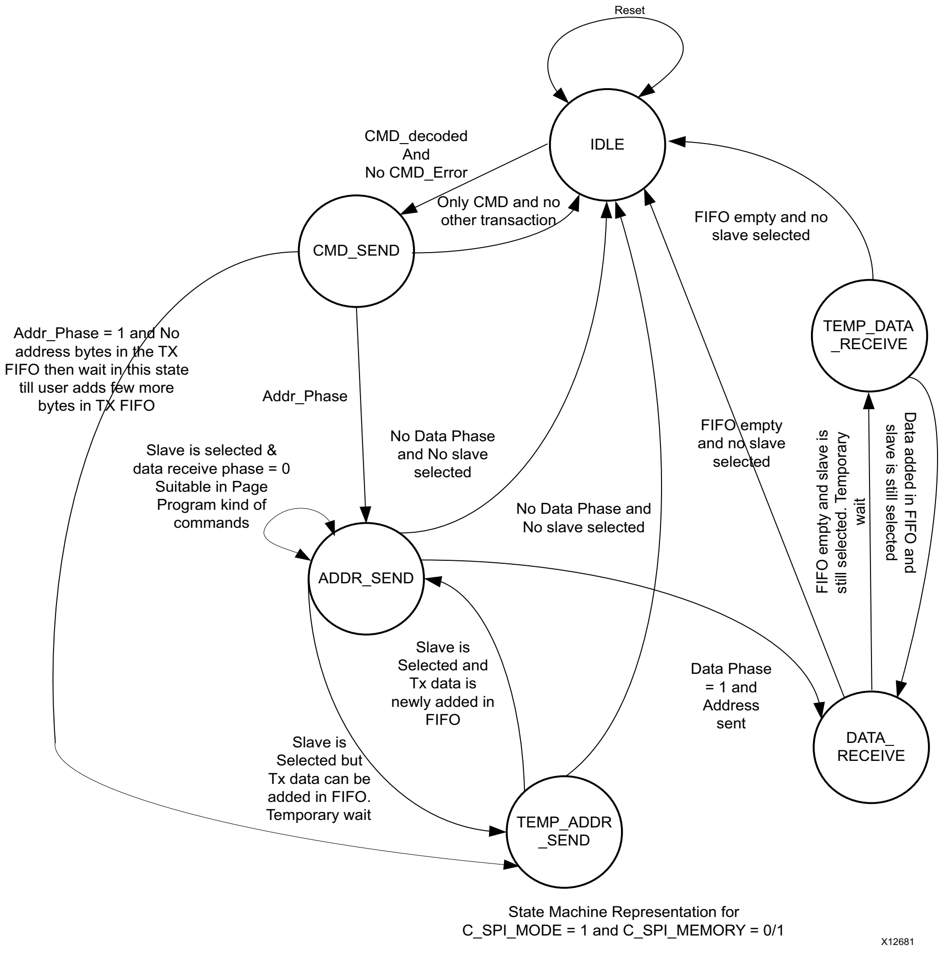


Figure 3: **Logical State Representation for Dual SPI Mode and C_SPI_MEMORY = 0 or 1**

Design Parameters

To design an AXI Quad SPI IP core that is uniquely tailored for your system, certain features can be parameterized. Parameterization affords a measure of control over the function, resource usage and performance of the implemented AXI Quad SPI core. The features that can be parameterized are shown in Table 1.

In addition to the parameters listed in this table, there are also *inferred* parameters for each AXI interface in the EDK tools. Through the design, these EDK-inferred parameters control the behavior of the AXI interconnect. For a complete list of the interconnect settings related to the AXI interface, see the *AXI Interconnect IP Data Sheet (DS768)* [Ref 5].

Table 1: **Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameters					
G1	Target FPGA family	C_FAMILY	virtex6, spartan6, virtex7, kintex7, artix7, zynq	virtex6	string
Common AXI Parameters					
G2	AXI4-Lite base address	C_BASEADDR	Valid Address[1]	None[3]	std_logic_vector
G3	AXI4-Lite high address	C_HIGHADDR	Valid Address[1]	None[3]	std_logic_vector
G4	AXI4 memory mapped Base Address	C_S_AXI4_BASEADDR	Valid Address[2]	None[3]	std_logic_vector
G5	AXI4 memory mapped High Address	C_S_AXI4_HIGHADDR	Valid Address[2]	None[3]	std_logic_vector
G6	AXI4-Lite address bus Width	C_S_AXI_ADDR_WIDTH	32	32	integer
G7	AXI4-Lite AXI data bus width	C_S_AXI_DATA_WIDTH	32	32	integer
G8	AXI4 memory mapped ID Width	C_S_AXI4_ID_WIDTH	1- 16[4]	4	integer
G9	AXI4 memory mapped address bus width	C_S_AXI4_ADDR_WIDTH	32[4]	32	integer
G10	AXI4 memory mapped AXI data bus width	C_S_AXI4_DATA_WIDTH	32[4]	32	integer
AXI Quad SPI IP Core Parameters					
G11	Choice of AXI interface	C_TYPE_OF_AXI4_INTERFACE	0, 1 0 = AXI4-Lite interface 1 = AXI4 memory mapped interface	0	integer
G12	Choice of Non-XIP and XIP mode	C_XIP_MODE	0, 1 0 = Enhanced mode (non-XIP mode) 1 = XIP mode	0[5]	integer

Table 1: **Design Parameters *(Cont'd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G13	Receive and transmit FIFO depth	C_FIFO_DEPTH	0, 16, 256 0 = FIFOs are not included in the design 16 = FIFOs are included in the design (Standard, Dual/Quad SPI mode, Legacy/Enhanced non-XIP mode only) 256 = FIFOs are included in the design (Dual/Quad SPI mode, Legacy/Enhanced non-XIP mode only)	16[6][7]	integer
G14	SPI clock frequency ratio	C_SCK_RATIO	2[8], 4, 8, Nx16 for N = 1, 2, 3, ...,128	16[7][9]	integer
G15	Total number of slave select bits	C_NUM_SS_BITS	1 - 32	1[7]	integer
G16	Select number of transfer bits as 8	C_NUM_TRANSFER_BITS	8, 16, 32	8[7][10]	integer
G17	SPI modes	C_SPI_MODE	0, 1, 2 0 = Standard SPI mode 1 = Dual SPI mode 2 = Quad SPI mode	0[11]	integer
G18	Use STARTUP primitive	C_USE_STARTUP	0, 1 0 = Exclude the STARTUP primitive from core 1 = Include the STARTUP primitive in the core	0[12]	integer
G19	The SPI memory device used as SPI slave	C_SPI_MEMORY	0, 1, 2 0 = Mixed mode memories 1 = Winbond memories are used as SPI slaves 2 = Numonyx memories are used as SPI slaves	1[13] [14]	integer

Notes:

1. The range C_BASEADDR to C_HIGHADDR is the address range for the AXI Quad SPI IP. This range is subject to restrictions to accommodate the simple address decoding scheme that is employed. The size of C_HIGHADDR - C_BASEADDR + 1 must be a power of two and must be at least 0x80 to accommodate all AXI Quad SPI IP core registers. However, a larger power of two can be chosen to reduce decoding logic. C_BASEADDR must be aligned to a multiple of the range size. These parameters are useful only in Legacy mode as well as XIP mode.
2. The C_S_AXI4_BASEADDR and C_S_AXI4_HIGHADDR are used only when the core is configured in Enhanced mode as well as in XIP mode.
3. No default value is specified to ensure that an actual value appropriate to the system is set by the user.
4. The parameters C_BASEADDR, C_HIGHADDR, C_S_AXI4_ID_WIDTH, C_S_AXI4_BASEADDR and C_S_AXI4_HIGHADDR are applicable only when the core is in XIP mode.
5. The C_XIP_MODE parameter is applicable only when the C_TYPE_OF_AXI4_INTERFACE is set to 1. When C_TYPE_OF_AXI4_INTERFACE = 1, then C_XIP_MODE can be either 0 or 1. When C_XIP_MODE = 0, the AXI4-Lite interface is replaced by AXI4 memory mapped interface. When C_XIP_MODE = 1, then AXI4-Lite and AXI4 memory mapped interface are used in the core. The C_TYPE_OF_AXI4_INTERFACE = 1 and C_XIP_MODE = 0 combination is mainly used for CDMA based applications. The C_TYPE_OF_AXI4_INTERFACE = 1 and C_XIP_MODE = 1 can be mainly used for booting based applications.
6. In Standard SPI mode (for Legacy as well as Enhanced non-XIP configuration), the option to include or exclude the FIFO in the design is allowed. In Standard SPI mode, if the FIFO is included in the design, the FIFO depth is restricted to 16 or 256 only. In Dual or Quad SPI mode, the recommended FIFO depth is 256.

7. The parameters C_FIFO_DEPTH, C_SCK_RATIO, C_NUM_SS_BIT, C_NUM_TRANSFER_BITS are not user-configurable when the core is configured in XIP mode. These parameters are fixed to default values like C_SCK_RATIO = 2, C_NUM_SS_BITS = 1, C_NUM_TRANSFER_BITS = 8 while C_FIFO_DEPTH is not used in the design.
8. The C_SCK_RATIO = 2 parameter is not supported when the AXI Quad SPI IP core is configured as a slave in Standard SPI mode. See Assigning the C_SCK_RATIO Parameter when using this parameter. The SPI clock is generated from the core AXI clock.
9. The C_SCK_RATIO parameter values from 2,4,8,16,32... are applicable only when C_TYPE_OF_AXI4_INTERFACE = 0, Standard SPI mode, and C_TYPE_OF_AXI4_INTERFACE = 1, Standard SPI mode. For Dual/Quad modes (in Legacy mode and Enhanced non-XIP mode) and XIP mode, the C_SCK_RATIO is always 2. Set the EXT_SPI_CLK to double the intended SPI clock at the SPI interface. The C_SCK_RATIO = 2 is used in this case to divide the EXT_SPI_CLK by 2 to generate the SPI clock. For example, if the SPI clock is chosen to operate at 50 MHz, then the EXT_SPI_CLK should be at 100 MHz. The C_SCK_RATIO is tied to 2 for all the modes which are described above (except Standard mode)
10. C_NUM_TRANSFER_BITS is allowed to vary between 8 or 16 or 32 only when the IP is configured in Standard SPI mode. When the core operates in Dual or Quad SPI mode, the transfer size is allowed to be 8 bits because the present SPI slave devices that support Dual or Quad mode are 8 bits.
11. C_SPI_MODE indicates the operating mode of this device. In Standard mode, the present AXI Quad SPI IP core functionality is inferred. The number of ports, as well as the operating mode is determined by this parameter.
12. Use of STARTUP primitive is applicable only when the core is intended for use in the Master mode. Read the user guide of respective FPGA family to understand the STARTUP primitive functionality. The STARTUP primitive is allowed to be included in the design when the core is targeted to use with Virtex-6 or 7 series FPGA families.
13. The core supports most of the command set for the Winbond and Numonyx memories. When the S_SPI_MEMORY parameter is set to 0, most common commands from Winbond and Numonyx are supported. When this parameter is set to 1, most of the Winbond memory commands are supported. When this parameter is set to 2, most of the Numonyx memory commands are supported. Depending SPI mode and the target memory, the command you set varies. Although this core supports most of the commands, read the exceptions for the commands which are not supported. This parameter is effective only when the core is configured in Dual or Quad SPI mode. Because the core gives better command database support while operating in single memory mode, it is recommended that dedicated memories are used in place of mixed mode memories when using the core in Dual or Quad SPI mode.
14. This parameter is applicable only in Dual or Quad mode.
15. It is recommended that the AXI4 interface clock and EXT_SPI_CLK clock originate from the same clock source.

Core Operation Mode and Design Parameters

Table 2 defines the parameters used to configure the core.

Table 2: **Core Operation Mode and Design Parameter Values**

Core Operation	Parameters							
	C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_FIFO_DEPTH	C_SPI_MODE	C_SCK_RATIO	C_NUM_SS_BITS	C_SPI_MEMORY	C_USE_STARTUP
Legacy mode[3]	0	0	0, 16, 256	0	2, 4, 8, Nx16 for N = 1, 2, 3,...,128[1]	1 - 32	0, 1, 2	0, 1
	0	0	16, 256	1	2[2]	1 - 32	0, 1, 2	0, 1
	0	0	16, 256	2	2[2]	1 - 32	0, 1, 2	0, 1
Enhanced mode[3]	1	0	0, 16, 256	0	2, 4, 8, Nx16 for N = 1, 2, 3,...,128[1]	1 - 32	0, 1, 2	0, 1
	1	0	16, 256	1	2[2]	1 - 32	0, 1, 2	0, 1
	1	0	16, 256	2	2[2]	1 - 32	0, 1, 2	0, 1
XIP mode[3]	1	1	64	0	2[2]	1[4]	0, 1, 2	0, 1
	1	1	64	1	2[2]	1[4]	0, 1, 2	0, 1
	1	1	64	2	2[2]	1[4]	0, 1, 2	0, 1

Notes:

1. In this mode the EXT_SPI_CLK can be the same as the AXI_ACLK or AXI4_ACLK, but it is recommended that 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, you need to set EXT_SPI_CLK to double the intended SPI clock. The C_SCK_RATIO divides this clock by 2 and the SPI clock is generated.
3. Most of the SPI flash memory commands operate at higher SPI clock rate. There are some exceptional command which operate at lesser frequency. It is your responsibility to configure the core with correct EXT_SPI_CLK and C_SCK_RATIO and use the appropriate commands. If slower operating commands are executed on higher SPI clock ratio, then the core does not generate any 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, C_NUM_SS_BITS is always equal to 1 and this parameter cannot be updated.

I/O Signals

Legacy Mode

The I/O signals when C_TYPE_OF_AXI4_INTERFACE is 0 are described in Table 3.

Table 3: **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 [(C_S_AXI_ADDR_WIDTH - 1):0]	AXI	I	-	AXI Write address. The write address bus gives the address of the write transaction.
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 [(C_S_AXI_DATA_WIDTH - 1):0]	AXI	I	-	Write data
P8	S_AXI_WSTB [((C_S_AXI_DATA_WIDTH/8) - 1):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 [(C_S_AXI_ADDR_WIDTH - 1):0]	AXI	I	-	Read address. The read address bus gives the address of a read transaction.

Table 3: **I/O Signal for Legacy Mode (AXI4-Lite Interface)** ***(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
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 [(C_S_AXI_DATA_WIDTH - 1):0]	AXI	O	0	Read data
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[(C_NUM_SS_BITS - 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[(C_NUM_SS_BITS - 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
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.

Table 3: **I/O Signal for Legacy Mode (AXI4-Lite Interface)** ***(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P30	IO0_T	SPI	O	1	3-state enable master output slave input. Active-Low
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. 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.

Parameters - I/O Signal Dependencies for Legacy Mode

The dependencies between the AXI Quad SPI IP core design parameters and I/O signals are described in Table 4.

Table 4: **Parameters - Signal Dependencies for Legacy Mode**

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G4	C_S_AXI_ADDR_WIDTH	P4, P14	-	Affects the number of bits in address bus
G5	C_S_AXI_DATA_WIDTH	P8, P7, P17		Affects the number of bits in data bus
G15	C_NUM_SS_BITS	P24, P25	-	Defines the total number of slave select bits
I/O Signals				
P4	S_AXI_AWADDR [(C_S_AXI_ADDR_WIDTH - 1):0]	-	G4	Width of the AXI bus write address varies with C_S_AXI_ADDR_WIDTH.
P7	S_AXI_WDATA [(C_S_AXI_DATA_WIDTH - 1):0]	-	G5	Width of the S_AXI_WDATA varies according to C_S_AXI_ DATA_WIDTH.
P8	S_AXI_WSTB [((C_S_AXI_DATA_WIDTH/8) - 1):0]	-	G5	Width of the S_AXI_WSTB varies according to C_S_AXI_DATA_WIDTH.
P14	S_AXI_ARADDR [(C_S_AXI_ADDR_WIDTH - 1):0]	-	G4	Width of the AXI bus read address varies with C_S_AXI_ADDR_WIDTH.
P17	S_AXI_RDATA [(C_S_AXI_DATA_WIDTH - 1):0]	-	G5	Width of the S_AXI_RDATA varies according to C_S_AXI_ DATA_WIDTH.
P24	SS_I[(C_NUM_SS_BITS - 1):0]	-	G11	The number of SS_I pins are generated based on C_NUM_SS_BITS.
P25	SS_O[(C_NUM_SS_BITS - 1):0]	-	G11	The number of SS_O pins are generated based on C_NUM_SS_BITS.

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 5.

Table 5: **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 [(C_S_AXI4_ID_WIDTH-1):0]	AXI	I	-	Write address ID: This signal is the identification tag for the write address group of signals.
P5	S_AXI4_AWADDR [(C_S_AXI_ADDR_WIDTH-1):0]	AXI	I	-	AXI4 Write address: The write address bus gives the address of the first transfer in a write burst transaction.

Table 5: **I/O Signal for Enhanced Mode (AXI4 Memory Mapped) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
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.
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 [(C_S_AXI _DATA_WIDTH-1):0]	AXI	I	-	Write data bus.
P15	S_AXI4_WSTB [((C_S_AXI_DATA_WIDTH/8)-1):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.

Table 5: **I/O Signal for Enhanced Mode (AXI4 Memory Mapped)** ***(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
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 [(C_S_AXI4_ID_WIDTH-1):0]	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.
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[(C_S_AXI4_ ID_WIDTH-1):0]	AXI	I	-	Read address ID: This signal is the identification tag for the read address group of signals.
P24	S_AXI4_ARADDR[(C_S_ AXI_ADDR_WIDTH -1):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 5: **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 [(C_S_AXI4_ID_ WIDTH - 1):0]	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 [(C_S_AXI_ DATA_WIDTH -1):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.
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

Table 5: **I/O Signal for Enhanced Mode (AXI4 Memory Mapped) *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P42	SCK_T	SPI	O	1	3-state enable for SPI bus clock. Active-Low
P43	SS_I[(C_NUM_SS_BITS - 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[(C_NUM_SS_BITS - 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.
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.

Parameters - I/O Signal Dependencies

The dependencies between the AXI Quad SPI IP core design parameters and I/O signals are described in Table 6.

Table 6: **Parameters - Signal Dependencies for Enhanced Mode**

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G10	C_S_AXI4_ADDR_WIDTH	P5, P13,P23	-	Affects the number of bits in address bus
G9	C_S_AXI4_DATA_WIDTH	P15, P14, P34		Affects the number of bits in data bus
G8	C_S_AXI4_ID_WIDTH	P4, P19, P23,P33		Affects the ID width of read and write channel
G11	C_NUM_SS_BITS	P43, P44	-	Defines the total number of slave select bits
I/O Signals				
P4	S_AXI4_AWID [(C_S_AXI4_ID_WIDTH - 1):0]	-	G8	Width of AWID signal
P5	S_AXI4_AWADDR [(C_S_AXI_ADDR_WIDTH - 1):0]	-	G10	Width of the AXI bus write address varies with C_S_AXI_ADDR_WIDTH.
P14	S_AXI4_WDATA [(C_S_AXI_DATA_WIDTH - 1):0]	-	G9	Width of the S_AXI_WDATA varies according to C_S_AXI_DATA_WIDTH.
P15	S_AXI4_WSTB [((C_S_AXI_DATA_WIDTH/8) - 1):0]	-	G9	Width of the S_AXI_WSTB varies according to C_S_AXI_ DATA_WIDTH.
P19	S_AXI4_BID [(C_S_AXI4_ID_WIDTH - 1):0]	-	G8	Width of BID signal
P23	S_AXI4_ARID [(C_S_AXI4_ID_WIDTH - 1):0]	-	G8	Width of ARID signal
P24	S_AXI4_ARADDR [(C_S_AXI4_ADDR_WIDTH - 1):0]	-	G10	Width of the AXI bus read address varies with C_S_AXI_ADDR_WIDTH.
P33	S_AXI4_RID [(C_S_AXI4_ID_WIDTH - 1):0]	-	G8	Width of RID signal
P34	S_AXI4_RDATA [(C_S_AXI4_DATA_WIDTH - 1):0]	-	G9	Width of the S_AXI_RDATA varies according to C_S_AXI_ DATA_WIDTH.
P43	SS_I[(C_NUM_SS_BITS - 1):0]	-	G11	The number of SS_I pins are generated based on C_NUM_SS_BITS.
P44	SS_O[(C_NUM_SS_BITS - 1):0]	-	G11	The number of SS_O pins are generated based on C_NUM_SS_BITS.

Enhanced XIP Mode

When C_TYPE_OF_AXI_INTERFACE is 1 and C_XIP_MODE is 1, the design has the I/O ports described in Table 7.

Table 7: **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	-	AXI 4Clock
P4	S_AXI4_ARESETN	AXI4	I	-	AXI 4 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 [(C_S_AXI4_ADDR_WIDTH - 1):0]	AXI	I	-	AXI Write address. The write address bus gives the address of the write transaction.
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 [(C_S_AXI_DATA_WIDTH - 1):0]	AXI	I	-	Write data
P10	S_AXI_WSTB [((C_S_AXI_DATA_WIDTH/8) - 1):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 [(C_S_AXI_ADDR_WIDTH - 1):0]	AXI	I	-	Read address. The read address bus gives the address of a read transaction.

Table 7: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
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_ARREADY, is High.
P18	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					
P19	S_AXI_RDATA [(C_S_AXI_DATA_WIDTH - 1):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 [(C_S_AXI4_ID_WIDTH-1):0]	AXI	I	-	Write address ID: This signal is the identification tag for the write address group of signals.
P24	S_AXI4_AWADDR [(C_S_AXI_ADDR_WIDTH-1):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 7: **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 [(C_S_AXI _DATA_WIDTH-1):0]	AXI	I	-	Write data bus.
P34	S_AXI4_WSTB [((C_S_AXI_DATA_WIDTH/8)-1):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 [(C_S_AXI4_ID_WIDTH-1):0]	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.
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.

Table 7: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
AXI4 Memory Mapped Read Address Channel Signals					
P42	S_AXI4_ARID [(C_S_AXI4_ID_WIDTH-1):0]	AXI	I	-	Read address ID: This signal is the identification tag for the read address group of signals.
P43	S_AXI4_ARADDR [(C_S_AXI4_ADDR_WIDTH -1):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.
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.

Table 7: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
AXI4 Memory Mapped Read Data Channel Signals					
P52	S_AXI4_RID [(C_S_AXI4_ID_ WIDTH - 1):0]	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 [(C_S_AXI_ DATA_WIDTH -1):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.
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[(C_NUM_SS_BITS - 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[(C_NUM_SS_BITS - 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.

Table 7: **I/O Signal Description in XIP Mode *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
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 C_NUM_SS_BITS is always 1 and this parameter cannot be changed.

Parameters - I/O Signal Dependencies for Enhanced XIP Mode

The dependencies between the AXI Quad SPI IP core design parameters and I/O signals are described in Table 8.

Table 8: **Parameters - Signal Dependencies for XIP Mode**

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G6	C_S_AXI_ADDR_WIDTH	P6, P10	-	Affects the number of bits in address bus
G7	C_S_AXI_DATA_WIDTH	P19, P9, P16		Affects the number of bits in data bus
G8	C_S_AXI4_ID_WIDTH	P5, P20, P24,P52, P54		Affects the no. of bits in ID width
G9	C_S_AXI4_ADDR_WIDTH	P43, P24		Affects the number of bits in address bus
G10	C_S_AXI4_DATA_WIDTH	P33, P34		Affects the number of bits in data bus
I/O Signals				
P6	S_AXI_AWADDR [(C_S_AXI_ADDR_WIDTH - 1):0]	-	G6	Width of the AXI bus write address varies with C_S_AXI_ADDR_WIDTH.
P16	S_AXI_WDATA [(C_S_AXI_DATA_WIDTH - 1):0]	-	G7	Width of the S_AXI_WDATA varies according to C_S_AXI_DATA_WIDTH.
P9	S_AXI_WSTB [((C_S_AXI_DATA_WIDTH/8) - 1):0]	-	G7	Width of the S_AXI_WSTB varies according to C_S_AXI_DATA_WIDTH.
P10	S_AXI_ARADDR [(C_S_AXI_ADDR_WIDTH - 1):0]	-	G6	Width of the AXI bus read address varies with C_S_AXI_ADDR_WIDTH.
P19	S_AXI_RDATA [(C_S_AXI_DATA_WIDTH - 1):0]	-	G7	Width of the S_AXI_RDATA varies according to C_S_AXI_DATA_WIDTH.
P23	S_AXI4_AWID [(C_S_AXI4_ID_WIDTH-1):0]	-	G8	Width of AWID signal varies with this parameter
P24	S_AXI4_AWADDR [(C_S_AXI4_ADDR_WIDTH-1):0]	-	G9	Width of the AXI4 memory mapped bus write address varies with C_S_AXI4_ADDR_WIDTH

Table 8: **Parameters - Signal Dependencies for XIP Mode *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
P33	S_AXI4_WDATA [(C_S_AXI_DATA_WIDTH - 1):0]	-	G10	AXI4 Write data: The write data bus gives the data of the first transfer in a write burst transaction.
P34	S_AXI4_WSTB [((C_S_AXI4_DATA_WIDTH/8) - 1):0]	-	G10	Width of the S_AXI4_WSTB varies according to C_S_AXI4_DATA_WIDTH.
P38	S_AXI4_BID [((C_S_AXI4_ID_WIDTH/8) - 1):0]	-	G8	The number of BID generated based on C_S_AXI4_ID_WIDTH
P42	S_AXI4_ARID [(C_S_AXI4_ID_WIDTH-1):0]	-	G8	Width of ARID signal varies with this parameter
P43	S_AXI4_ARADDR [(C_S_AXI4_ADDR_WIDTH-1):0]	-	G9	Width of the AXI4 memory mapped bus read address varies with C_S_AXI4_ADDR_WIDTH
P52	S_AXI4_RID [(C_S_AXI4_ID_WIDTH-1):0]	-	G8	Width of RID signal varies with this parameter
P53	S_AXI4_RDATA [(C_S_AXI_DATA_WIDTH - 1):0]	-	G10	AXI4 read data: The write data bus gives the data of the first transfer in a read burst transaction.

Register Overview Table

Legacy and Enhanced Non-XIP Mode

Table 9 shows the set of registers which is applicable when C_TYPE_OF_AXI4_INTERFACE is 0 or 1 and C_XIP_MODE is 0. The AXI Quad SPI core has some registers which should be accessed individually. These registers are configurable and accessible through either the AXI4-Lite interface or the AXI4 memory mapped interface (Enhanced mode). If the AXI4 interface is chosen then all registers are accessed as 32-bit access.

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

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
Core Grouping				
C_BASEADDR + 40	SRR	Write	N/A	Software Reset Register
C_BASEADDR + 60	SPICR	R/W	0x180	SPI Control Register
C_BASEADDR + 64	SPISR	Read	0x0a5	SPI Status Register
C_BASEADDR + 68	SPI DTR	Write	0x0	SPI Data Transmit Register A single register or a FIFO
C_BASEADDR + 6C	SPI DRR	Read	N/A[1]	SPI Data Receive Register A single register or a FIFO
C_BASEADDR + 70	SPISSR	R/W	No slave is selected 0xffff	SPI Slave Select Register
C_BASEADDR + 74	SPI Transmit FIFO Occupancy Register[2]	Read	0x0	Transmit FIFO Occupancy Register
C_BASEADDR + 78	SPI Receive FIFO Occupancy Register[2]	Read	0x0	Receive FIFO Occupancy Register
Interrupt Controller Grouping				

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

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_BASEADDR + 1C	DGIER	R/W	0x0	Device Global Interrupt Enable Register
C_BASEADDR + 20	IPISR	R/TOW[3]	0x0	IP Interrupt Status Register
C_BASEADDR + 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. It is recommended that this register not be considered for power on reset conditions.
2. Exists only when C_FIFO_DEPTH = 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 C_TYPE_OF_AXI4_INTERFACE is either 0 (Legacy mode) or 1 (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 upon reset and return random value if read.

Register Details

Software Reset Register (SRR)

The Software Reset Register permits the programmer to reset the core independently of other cores in the system. To activate the software generated reset, the value of 0x0000_000a must be written to this register. This action resets the core register for 4 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 3 and described in Table 10. Any attempt to read this register returns undefined data.

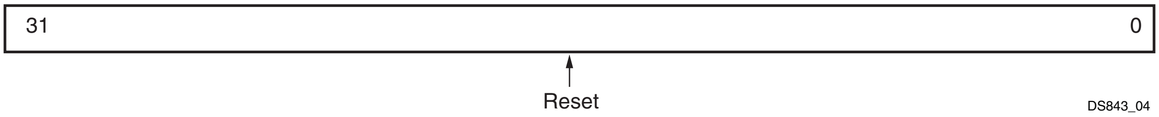


Figure 4: **Software Reset Register (C_BASEADDR + 0x40)**

Table 10: **Software Reset Register (SRR) Description (C_BASEADDR + 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 IP core.

SPI Control Register (SPICR)

The SPI Control Register (SPICR) allows programmer control over various aspects of the AXI Quad SPI IP core. The bit assignment in the SPICR is shown in Figure 5 and described in Table 11.

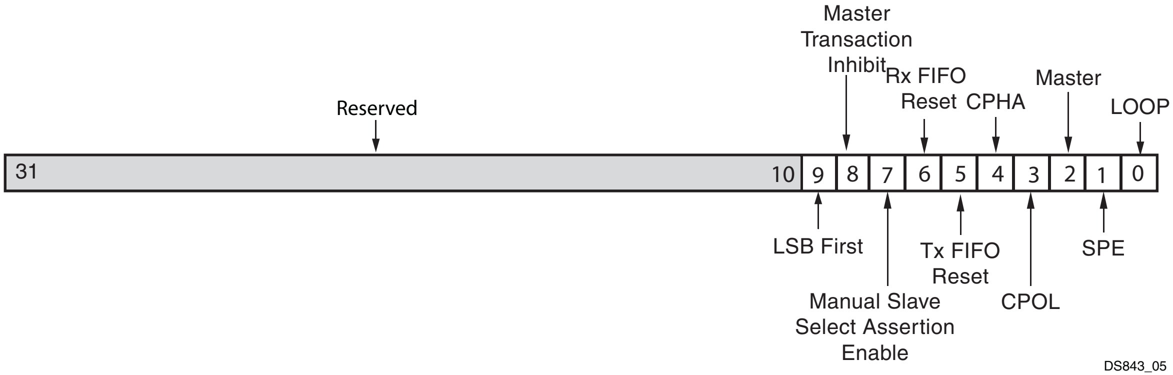


Figure 5: **SPI Control Register (C_BASEADDR + 0x60)**

Table 11: **SPI Control Register (SPICR) Description (C_BASEADDR + 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
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 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

Table 11: **SPI Control Register (SPICR) Description (C_BASEADDR + 0x60) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
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.
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.
1	SPE	R/W	0	SPI System Enable: Setting this bit to 1 enables the SPI devices as noted below. When set to - 0= SPI system disabled. Both master and slave outputs are in '3-state' and slave inputs 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 C_SPI_MODE = 0.

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 (SPISR)

The SPI Status Register (SPISR) is a read-only register that provides the status of some aspects of the AXI Quad SPI IP core to the programmer. The bit assignment in the SPISR is shown in Figure 6 and described in Table 12. Writing to the SPISR does not modify the register contents.

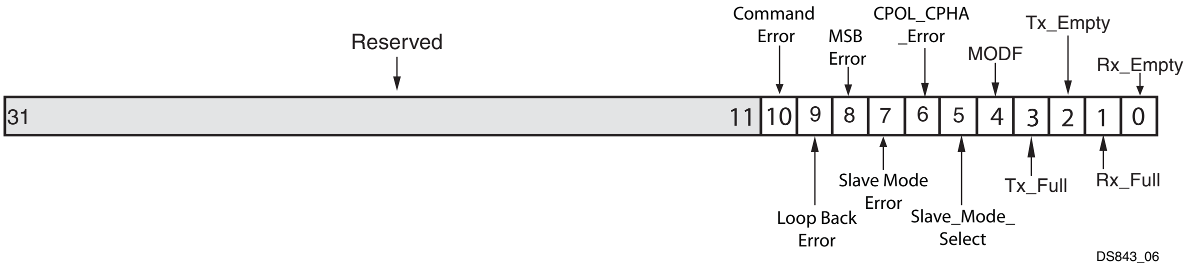


Figure 6: **SPI Status Register (C_BASEADDR + 0x64)**

Table 12: **SPI Status Register (SPISR) Description (C_BASEADDR + 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	Loop Back 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 12: **SPI Status Register (SPISR) Description (C_BASEADDR + 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 0in the control register (SPICR). 0= Master mode is set in the control register (SPICR). ***Note:*** In Dual/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 $\overline{\text{SS}}$ 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.
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 upon an AXI write to the FIFO or transmit register. For Dual/Quad SPI mode, the FIFO is always present in the core.

Table 12: **SPI Status Register (SPISR) Description (C_BASEADDR + 0x64)** ***(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
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 (SPI DTR)

This register 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.

Before filling the SPI DTR, it should be in the reset state, so that the DTR FIFO write pointer points to the 0th location.

Dual/Quad Mode

The first write must always be an 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 command and address (optional). In one of these modes, the dummy bytes are required to fill in DTR, which is used for internal count of data reception from memory. In commands such as Dual mode read, these bytes are not transferred on data lines, but are used by the internal logic to keep the 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 a subsequent write overwrites the data. The SPI DTR is shown in Figure 7 and Table 13 shows the data format.

When a transmit FIFO exists, data is written directly in 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 C_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.

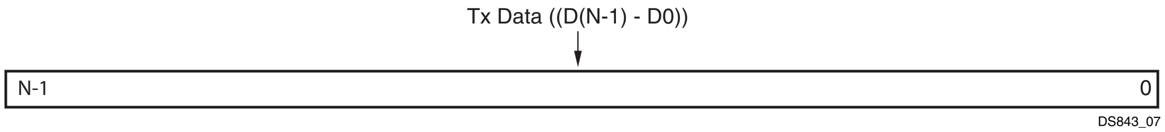


Figure 7: **SPI Data Transmit Register (C_BASEADDR + 0x68)**

Table 13: **SPI Data Transmit Register (SPI DTR) Description (C_BASEADDR + 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 C_NUM_TRANSFER_BITS = 8
				N = 16 when C_NUM_TRANSFER_BITS = 16
				N = 32 when C_NUM_TRANSFER_BITS = 32

Notes:

1. The D_{N-1} bit always represents the MSB bit irrespective of 'LSB first' or 'MSB first' transfer selection. When C_NUM_TRANSFER_BITS = 8 or 16, the unused upper bits ((C_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 (SPI DRR)

This register 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 (C_FIFO_DEPTH = 16 or 256) or exclusion (C_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 C_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 to read an empty receive register or FIFO is made, 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 in the receive FIFO during core transactions, the data in this register should be considered for reading. The SPI DRR is shown in Figure 8, while the specifics of the data format is described Table 14.

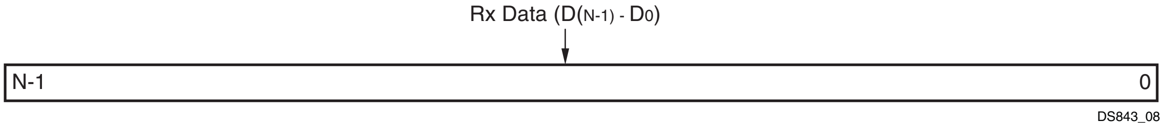


Figure 8: **SPI Data Receive Register (C_BASEADDR + 0x6C)**

Table 14: **SPI Data Receive Register (SPI DRR) Description (C_BASEADDR + 0x6C)**

Bit(s)	Name	Core Access	Reset Value	Description

[N-1] - 0	Rx Data[1] (D_{N-1} - D_0)	Read only	NA	N-bit SPI receive data. N can be 8, 16 or 32.[2]
				N = 8 when C_NUM_TRANSFER_BITS = 8
				N = 16 when C_NUM_TRANSFER_BITS = 16
				N = 32 when C_NUM_TRANSFER_BITS = 32

Notes:

1. The D_{N-1} bit always represents the MSB bit irrespective of 'LSB first' or 'MSB first' transfer selection. When C_NUM_TRANSFER_BITS = 8 or 16, the unused upper bits ((C_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 (SPISSR)

This register contains an active-Low, one-hot encoded slave select vector $\overline{SS}$ of length N, where N is the number of slaves set by parameter C_NUM_SS_BITS. The $\overline{SS}$ occupies the right-most bits of the register. At most, one bit can be asserted Low. This bit denotes the slave with whom the local master communicates. The bit assignment in the SPISSR is shown in Figure 9 and described in Table 15.

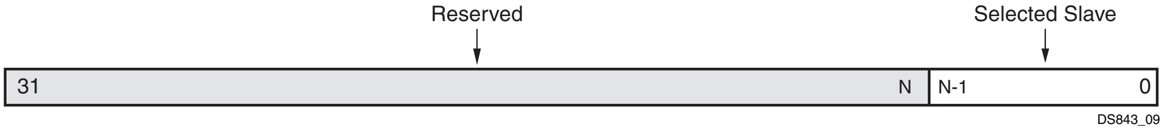


Figure 9: **SPI Slave Select Register (C_BASEADDR + 0x70)**

Table 15: **SPI Slave Select Register (SPISSR) Description (C_BASEADDR + 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 (Tx_FIFO_OCY)

The SPI Transmit FIFO Occupancy Register is present only if the AXI Quad SPI IP core is configured with FIFOs (C_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. A write to it (or a read when the FIFO is empty) does not affect the register contents. 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 10, while the specifics of the data format are described in Table 16.

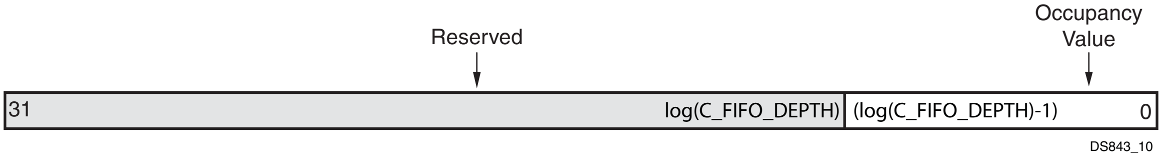


Figure 10: **SPI Transmit FIFO Occupancy Register (C_BASEADDR + 0x74)**

Table 16: **SPI Transmit FIFO Occupancy Register Description (C_BASEADDR + 0x74)**

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

Notes:

1. In Standard SPI mode - Only 3 - 0 bits of this register are valid as the FIFO depth of 16 only is allowed.
2. In Dual or Quad SPI mode - based on the FIFO depth the bit position is changed. Maximum of 256 deep byte wide FIFO is allowed in the design.

SPI Receive FIFO Occupancy Register (Rx_FIFO_OCY)

The SPI Receive FIFO Occupancy Register is present only if the AXI Quad SPI IP core is configured with FIFOs (C_FIFO_DEPTH = 16 or 256). If it 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 11, while the specifics of the data format are described in Table 17.

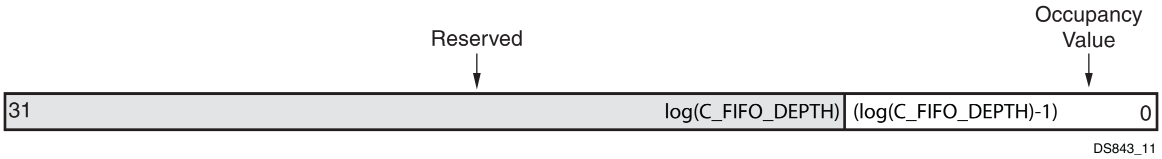


Figure 11: **SPI Receive FIFO Occupancy Register (C_BASEADDR + 0x78)**

Table 17: **SPI Receive FIFO Occupancy Register Description (C_BASEADDR + 0x78)**

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

Notes:

1. In Standard SPI mode - Only 3 - 0 bits of this register are valid as the FIFO depth of 16 is only allowed.
2. In Dual or Quad SPI mode - based on the FIFO depth the bit position is changed. Maximum of 256 deep byte wide FIFO is allowed in the design.

Interrupt Register Set Description

The AXI Quad SPI IP core has many distinct interrupts that are sent to the interrupt controller. The AXI Quad SPI IP interrupt controller allows each interrupt to be enabled independently (using the IP interrupt enable register (IPIER)). The interrupt registers are in 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, C_FIFO_DEPTH, is set to 16 or 256 then almost all of the interrupts shown in Table 19 are available.

In Standard SPI mode, when C_FIFO_DEPTH is 0, all of the interrupts except bit 6, transmit FIFO Half Empty and bit 8 DRR Not Empty, which is not present in this mode, are available.

Device Global Interrupt Enable Register (DGIER)

The Device Global Interrupt Enable Register is used to globally enable the final interrupt output from the interrupt controller as shown in Figure 12 and described in Table 18. This bit is a read/write bit and is cleared upon reset.

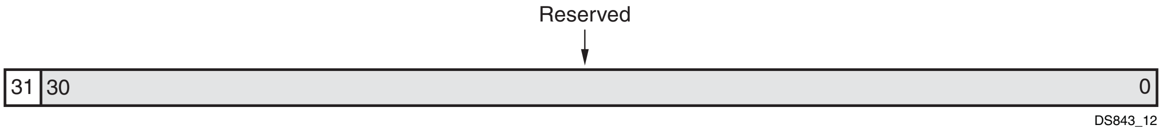


Figure 12: **Device Global Interrupt Enable Register (DGIER) (C_BASEADDR + 0x1C)**

Table 18: **Device Global Interrupt Enable Register(DGIER) Description (C_BASEADDR + 0x1C)**

Bit(s)	Name	Access	Reset Value	Description
31	GIE	R/W	0	Global Interrupt Enable Enables all individually enabled interrupts to be passed 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 interrupt controller has the 32-bit Interrupt Status Register that can enable each interrupt independently. This register collects all of the interrupt events. Bit assignments are shown in Figure 13 and described in Table 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 upon reset.

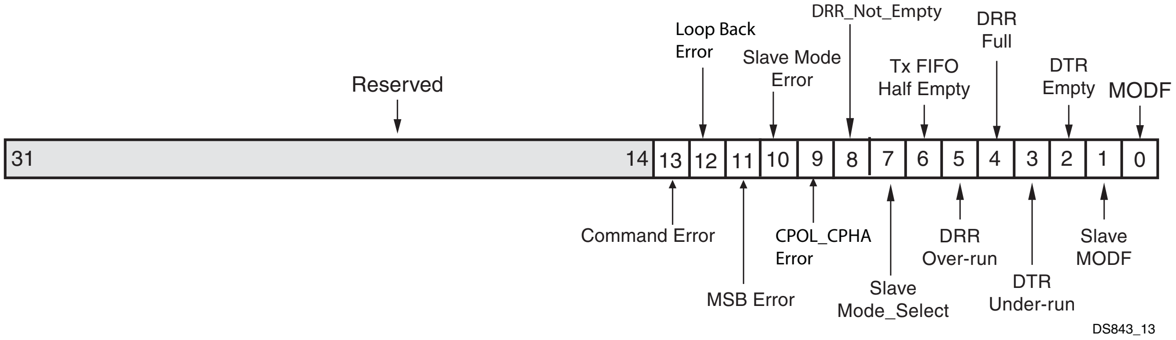


Figure 13: **IP Interrupt Status Register (IPISR) (C_BASEADDR + 0x20)**

Table 19: **IP Interrupt Status Register (IPISR) Description (C_BASEADDR + 0x20)**

Bit(s)	Name	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	Loop Back 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 0state.
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 0state.
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 0state.
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 C_FIFO_DEPTH = 16/256 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 one-clock period strobe to the interrupt register when the core receives first data beat. ***Note:*** The assertion of this bit is applicable only when the C_FIFO_DEPTH = 16/256 and the core is configured in slave mode in Standard SPI mode. When C_FIFO_DEPTH = 0 this bit always returns 0. This bit has no significance in Dual/Quad mode.

Table 19: **IP Interrupt Status Register (IPISR) Description (C_BASEADDR + 0x20)** ***(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
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 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. As an 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. 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 IP 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 an 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).
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 an SPI element (An element can be a byte, half-word, or word depending on the value of the C_NUM_TRANSFER_BITS generic) 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 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 an 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 an 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 an 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 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.

Table 19: **IP Interrupt Status Register (IPISR) Description (C_BASEADDR + 0x20) *(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
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 $\overline{SS}$ signal goes active while the SPI device is configured as a slave, but is not enabled. This bit is set immediately upon $\overline{SS}$ going active and continually set if $\overline{SS}$ 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 $\overline{SS}$ signal goes active while the SPI device is configured as a master. This bit is set immediately upon $\overline{SS}$ 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 controller IPIER register allows the system interrupt output to be active. This interrupt is generated if the enabled bit in IPIER detects any activity on the corresponding IPISR bit. The IPIER has an enable bit for each defined bit of the IPISR as shown in Figure 14 and described in Table 20. All bits are cleared upon reset.

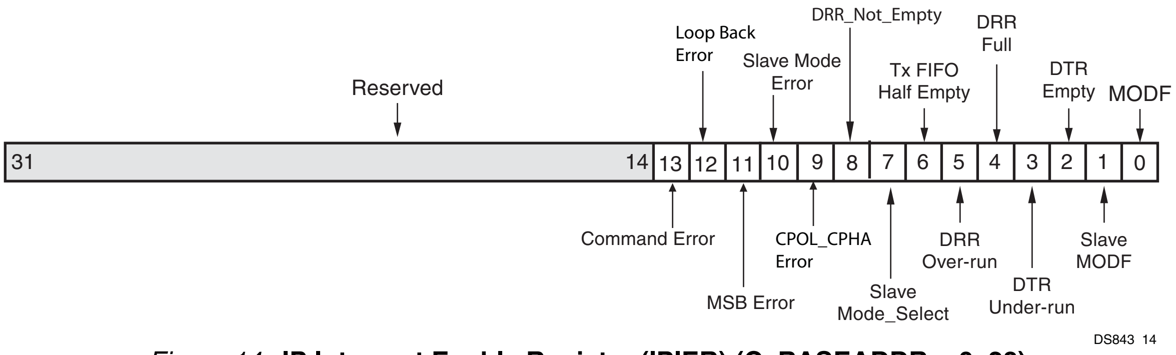


Figure 14: **IP Interrupt Enable Register (IPIER) (C_BASEADDR + 0x28)**

Table 20: **IP Interrupt Enable Register (IPIER) Description (C_BASEADDR + 0x28)**

Bit(s)	Name	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	Loop Back 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.

Table 20: **IP Interrupt Enable Register (IPIER) Description (C_BASEADDR + 0x28)** ***(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
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.
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 C_FIFO_DEPTH = 1 and the core is configured in slave mode of Standard SPI mode. If C_FIFO_DEPTH = 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 IPIER. Therefore, it is recommended that this bit is used only when C_FIFO_DEPTH is not equal to 0 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
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 core is configured in XIP mode, only a limited number of registers are available through the AXI4-Lite interface. These registers are the XIP Configuration Register (XIP-CR) and the XIP Status Register (XIP-SR). The AXI Quad SPI core has some registers which should be accessed individually. These registers are configurable and accessible through the AXI4-Lite interface and are accessed as 32-bit access. Table 21 provides a summary of the AXI Quad SPI IP core registers in XIP mode.

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

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

XIP Mode Commands

The registers in this mode are accessed through the AXI4-Lite interface. Based on the core configuration in this mode, the core supports three read commands. These are: Fast Read (0x0Bh), Fast Read Dual I/O (0xBBh), Fast Read Quad I/O (0xEBh). Based on the C_SPI_MODE parameter one of these commands is supported by the core. These commands are in built commands in the core and you do not need to configure any other command. These commands are fixed commands and when the mode is set using C_SPI_MODE, the same command operates over the complete transactions.

XIP Configuration Register (XIP-CR)

This register is used while configuring the XIP mode (read only mode) of operation. This is a read-write register, used to configure the CPOL and CPHA modes. In XIP mode, CPOL-CPHA = 00 or CPOL-CPHA = 11 are supported. If any other combination is selected then the error flag is set in the status register and the transaction on the AXI4 full interface is not accepted by the core. Before the start of any new transaction at the AXI4 interface, the core checks the CPOL and CPHA settings.

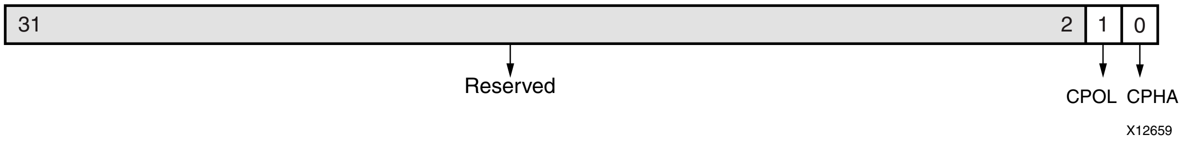


Figure 15: **XIP Configuration Register (XIP-CR) (C_BASEADDR+0x60)**

XIP Status Register (XIP-SR)

This register is used to check the status of command and other processes operated by the core. In this case, if the core receives write commands or write transactions, an error is generated and the status register indicates the status. This is a read-only register; any attempt to write to this register is completed with 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, you should check the status registers for any errors. If there are any errors, the core does not accept the AXI4 transactions.

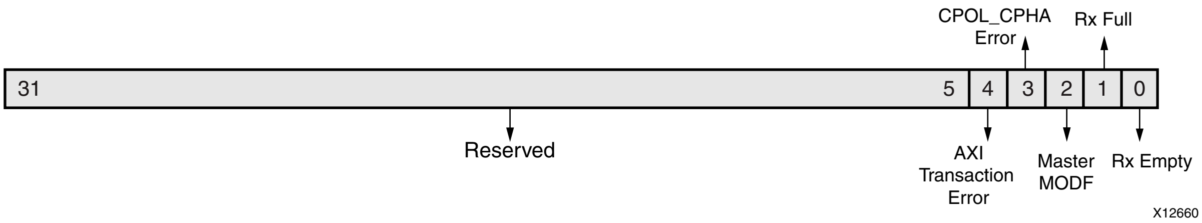


Figure 16: **XIP Status Register (XIP-SR) (C_BASEADDR+0x64)**

Table 22: **XIP Status Register Description (XIP-SR) (C_BASEADDR+0x64)**

Bit(s)	Name	Access	Reset Value	Description
31 downto 5	Reserved	NA	0	
4	AXI Transaction Error	R	0	AXI Transaction Error
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

Design Description

Functionality based on AXI Interfaces

AXI4-Lite Interface

If the AXI4-Lite interface is chosen then all the registers including the SPI DTR and SPI DRR are accessed as 32-bits. These registers are accessed as a single access at a time. The SPI DTR and SPI DRR registers have only 8 bits valid out of the 32.

AXI4 Memory Mapped Interface (Enhanced Mode)

If the AXI4 memory mapped interface is chosen then it is mandatory that all the registers are accessed as single length access only. If burst is attempted (except for the SPI DTR or SPI DRR) then the core behavior is not guaranteed. The SPI DTR and SPI DRR registers can be accessed as 32-bits (only 8 bits are valid). 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. Before initiating the recursive FIXED burst, you should read the occupancy registers. These occupancy registers provide information about what length of FIXED burst needs to be performed. The remaining 24 bits in the 32-bit access of the SPI DTR and SPI DRR are invalid. Therefore the user application should ignore this data and only the last 8 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 is not guaranteed.

When starting any new transaction at the SPI, the DTR FIFO should be always filled with command, followed by address, dummy bytes and 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 as 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 type of applications.

Standard SPI Device Features with only AXI4-Lite Interface

In addition to the features listed in the Features section, the SPI device includes the following standard features in Standard SPI configuration:

- 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 be used only with external slave devices. An optional operation where the $\overline{SS}$ output is toggled automatically with each 8-bit character transfer by the master device can be selected 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 of the AXI clock rate in master mode and one-fourth of the AXI clock rate in slave modes. C_SCK_RATIO = 2 is not supported in Slave mode (because of the synchronization method used between the AXI and SPI clocks). It is required to take care of the AXI and external clock signals alignment when configured in slave mode.
- Parameterizable baud rate generator for 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 an SPI data transfer is in progress.
- Back-to-back transactions are supported — there can be multiple byte/half-word/word transfers taking place without interruption, 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 — an 8-bit shift register in the SPI master device and another 8-bit shift register in a SPI slave device that are connected.

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

The SPI bus to a given slave device (N-th device) consists of four wires, Serial Clock (SCK), IO0(Master Out Slave In (MOSI)), IO1 (Master In Slave Out (MISO)), and Slave Select ($\overline{SS}$(N)). The signals SCK, IO0(MOSI), and IO1(MISO) are shared for all slaves and masters See Figure 17 for more reference.

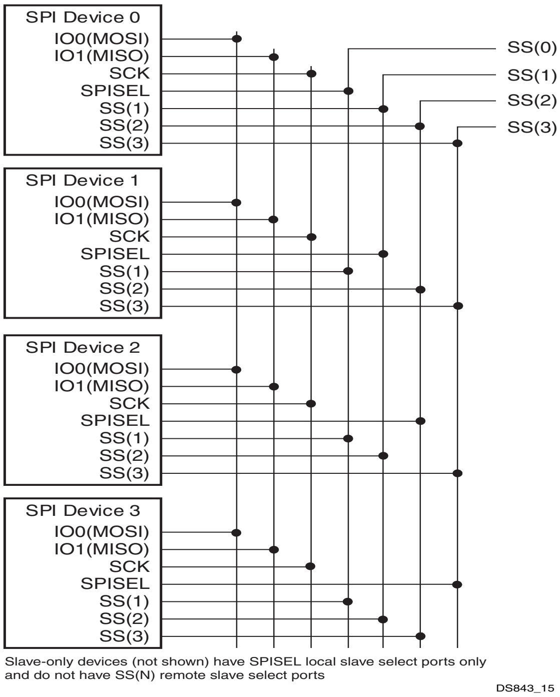


Figure 17: **Multi-Master Configuration Block Diagram when the core is configured in Standard SPI Mode**

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 SPI master/slave devices to be both internal to the FPGA and SPI slave devices to be external to the FPGA. SPI pins are automatically generated through the Xilinx Platform Generator when interfacing to an external SPI slave device. Multiple SPI master/slave devices are shown in Figure 17. The same configuration diagram is applicable for Dual mode.

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

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 you can include FIFOs in the core as shown in Figure 1.

When FIFOs are implemented, the slave select address is required to be the same for all data buffered in the FIFOs. This is required because a FIFO for the slave select address is not implemented. 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 in the FIFO, the occupancy number is incremented and when an 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 a read-only register.

If a write is attempted when the FIFO is full, an acknowledgement is given along with an error signal generation. Interrupts associated with the transmit FIFO include data transmit FIFO empty, transmit FIFO half empty, and transmit FIFO under-run. See Interrupt Register Set Description for details.

The receive FIFO is read-only. When data is read from the FIFO, the occupancy number is decremented and when an SPI transfer is completed, the number is incremented. If a read is attempted when the FIFO is empty, acknowledgement is given along with an error signal generation. 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 an 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 IP 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 the depth of FIFO. The depth of this FIFO can be any one of the 16 or 256 values. The width of the FIFO in this mode is always 8 bits wide.

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)) because the transmitter output is internally connected 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 an SPI device, configured as a master, is selected (that is, its $\overline{\text{SS}}$ 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 is when an SPI device configured as a slave, but not enabled, is selected (that is, its $\overline{\text{SS}}$ 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 condition and overrun condition error detection is also provided. Underrun conditions can happen only in slave mode operation. 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.

Assigning the C_SCK_RATIO Parameter

The AXI Quad SPI IP core is tested in hardware with the 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 with a 50 MHz SPI clock rate). Read the data sheet of the targeted SPI slave flash memory or EEPROMs for the maximum speed of operation. Ensure that you use the correct values when deciding on the AXI clock and selecting the C_SCK_RATIO parameter. The AXI clock and the C_SCK_RATIO decide the clock at SCK pin of AXI Quad SPI IP core. While using different external SPI slave devices, the C_SCK_RATIO 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 Legacy Mode Only

The AXI Quad SPI core can be configured in the 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 the incoming signals are synchronized to the AXI clock when C_SCK_RATIO > 4. Because of the tight timing requirements when C_SCK_RATIO = 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 C_SCK_RATIO = 4. For other C_SCK_RATIO values, it is preferred, but might not be necessary to have such synchronization.

In slave mode operation it is strongly recommended to use the FIFO by setting C_FIFO_DEPTH = 16 or 256 in Standard SPI mode. In slave mode, two new interrupts are available in IPISR DRR_Not_Empty (bit 8) and Slave_Mode_Select (bit 7) along with the available interrupts. 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.

SPI Transfer Formats

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 in 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.

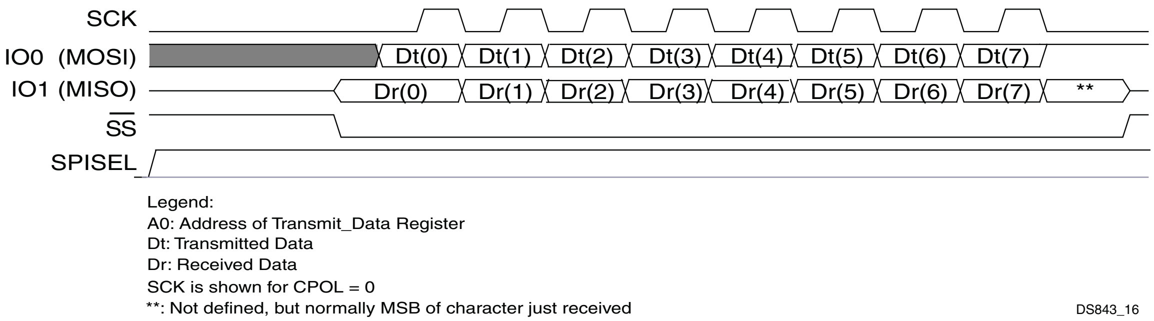
The different transfer formats are described in the following sections.

In Standard SPI mode, the CPHA - CPOL modes, any one of the 00, 01, 10, 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 18 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 with respect to S_AXI_AClk and SCK respectively.



Legend:
A0: Address of Transmit_Data Register
Dt: Transmitted Data
Dr: Received Data
SCK is shown for CPOL = 0
**: Not defined, but normally MSB of character just received

Figure 18: **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 respectively hardwired 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 out from its IO1 pin to the IO1 master pin, thus realizing full-duplex communication.

The Nth bit of the $\overline{\text{SS}}$(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, $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$(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 19. 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 with respect to S_AXI_Clk and SCK respectively.

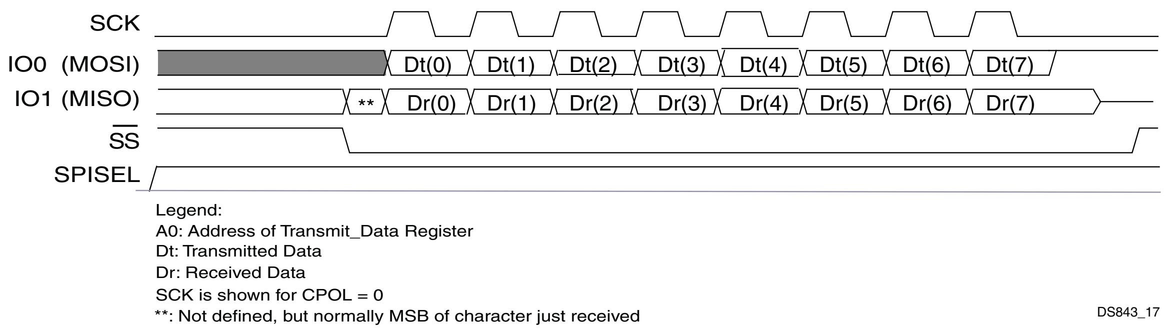


Figure 19: **Data Write-Read Cycle on SPI Bus with CPHA = 1 and SPICR(24) = 0 for 8-bit Data**

SPI Protocol Slave Select Assertion Modes

The Standard Mode SPI protocol is designed to have automatic slaves select assertion and manual slave select assertion which are described in the following sections. All the SPI transfer formats described in 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 provided to permit transfer of data with automatic toggling of the slave select ($\overline{SS}$) signal until all the 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 byte (or element) transfer, the $\overline{SS}$(N) output goes to 1. The data in 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 the AXI4-Lite and AXI4 memory mapped interface based core.

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 data registers of FIFOs 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 assertion of slave select signal and the number of elements transferred is controlled by the user. While performing complete memory read or page read operations, it is recommended that the Manual Slave Select mode is 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. The subsequent 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 IP 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, it is recommended that the software 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 $\overline{\text{SS}}$ 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, 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, the following happens:

- Rx_Empty bit, bit(0), Tx_Full bit, and bit(3) in the SPISR are cleared.
- Tx_Empty bit, bit(2), Rx_Full bit, and bit(1) in SPISR are set.
- DRR Full bit, bit(4), and 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 18 and Figure 19. However, the internal master clock provides this SCK edge which prompts the setting and clearing of the bits noted.

In this design, a counter was 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 IP core cannot be used in a system with external SPI slave devices that do not use the AXI clock.

When the AXI Quad SPI IP 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.

Using the C_USE_STARTUP Parameter

The C_USE_STARTUP parameter is applicable only for Virtex®-6, and 7 series devices and if the core is in Master SPI mode. When this parameter is included in the design, the STARTUP_VIRTEX6 (for Virtex-6 devices) or STARTUPE2 (for 7 series devices), is included in the design based on the targeted device for the core. When this parameter is included in the design, the respective primitive becomes part of the core after configuration of the FPGA. This parameter is not applicable to the Spartan®-6 devices.

See the Answer Records or the Device User Guide to understand more about the functionality and use of STARTUP primitives for the respective FPGA devices.

C_USE_STARTUP is 1

Core Behavior and Ports when Target Family is a 7 series FPGA

- The SCK_O port from the core is interfaced with the STARTUP2 primitive. The STARTUP2 can also be used in the pre-configuration process of the FPGA, where the external SPI slave memory is first configured before booting up the FPGA from it.
- 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.

Core Behavior and Ports when Target Family is a Virtex-6 FPGA

- The SCK and IO1_I port from the core is interfaced with the STARTUP_VIRTEX6 primitive.The STARTUP_VIRTEX6 can be used in the pre-configuration process of FPGA, where the external SPI slave memory is first configured, before booting up the FPGA from it.
- After configuration of the FPGA, the SCK_O port (output from the core) drives the USRCCLK0 port of the primitive. The external memory port IO1_I drives the DINSPI port of the primitive. These two signals are not available as the external ports of the core.

C_USE_STARTUP is 0

Core Behavior and Ports when Target Family is Virtex-6 or 7 Series FPGA

- 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 IOB at a user-configured location.

Core Behavior in Legacy and Enhanced Non-XIP Mode

This mode is when C_TYPE_OF_AXI4_INTERFACE is 0 or 1 and when 1, C_XIP_MODE is 0. The AXI Quad SPI core supports Winbond and Numonyx memories. Check the commands if different memories need to be tested with the core. If the commands, address, and data behavior is the same for a different memory, then it can be chosen as the base memory to test the core.

Internally, the core understands the commands and its expected behavior for the targeted memory through some intelligence in the core. The commands which are not supported by the Winbond or Numonyx memory part 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 an 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 upon read or write, with the modes set by Data Mode bits.

The dummy bytes, which are needed for some of the instructions for selected memory, should be part of 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, you must 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 1 command byte plus 3 address bytes plus 2 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 in the SPI DRR. The data is stored 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, you should ignore the first 6 bytes of the SPI DRR and the seventh byte onwards is the real data is available in the FIFO. This also applies to other Dual or Quad read commands.

For each fresh transaction, you must clear the SPI DTR FIFO. The first entry in the SPI DTR is always considered as 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 C_TYPE_OF_AXI4_INTERFACE is 1 and XIP mode is selected the core supports Standard, Dual and Quad modes.

- Standard mode is set with C_SPI_MODE = 0 and C_SPI_MEMORY = 1 or 2.
- Dual mode is set with C_SPI_MODE = 1 and C_SPI_MEMORY = 1 or 2.
- Quad mode is set with C_SPI_MODE = 2 and C_SPI_MEMORY = 1 or 2.

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.

When Quad mode is set, then you must pre-configure the Winbond memory 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 you have done this exercise before in XIP mode.

When the core is configured in Dual or Quad mode, before 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 needs one command and three dummy SPI cycles. This writing of the HPM command in the memory is done only at the power on condition. The memory is now placed in High Performance Mode (HPM - 0xA3 h) and allows DIOFR or QIOFR to operate in 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 you change these register values for the dummy cycles, then the core behavior is not guaranteed. It is recommended that the VCR and NVCR are left untouched with respect to the default configuration. At the start of each new transaction, the core sends the respective command, address and required dummy cycles and then receives data.

SPI Command, Address and Data Flow Description

For correct operation of the core, the SPI slave device instruction set is divided into SPI command, SPI address, and SPI data sections. This is shown in Figure 20.

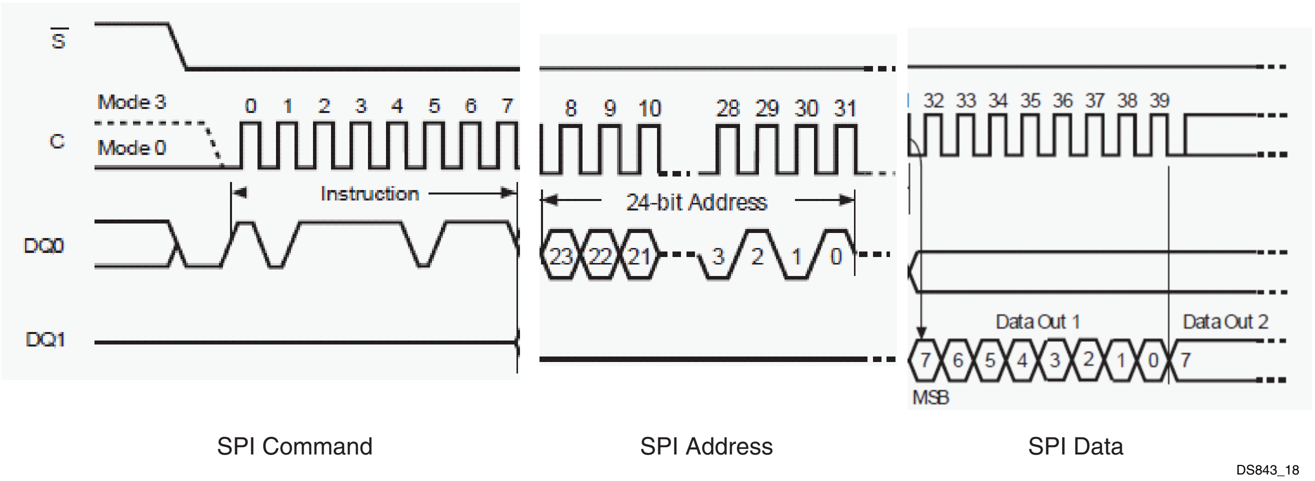


Figure 20: **SPI Transactions - CMD, ADDR and Data Phase**

SPI Command Section

If the first entry in SPI DTR matches with the Standard SPI transaction command (internal logic for Winbond or/and Numonyx memory) in the core, the core transfers the command on the IO0 (MOSI) line along with the SPI clock.

When C_SPI_MEMORY is 2 (Numonyx memory), then based on the supported command and C_SPI_MODE, the IO0, IO1, IO2, IO3 lines are used. When the first entry in SPI DTR matches the commands supported by memory, the core transfers the SPI commands on IO0, IO1 (and IO2, IO3) lines for the respective command.

When the C_SPI_MEMORY is 1 (Winbond memory), then based on the supported command, the IO0 line is used.

When the C_SPI_MEMORY is 0 (mixed mode memories), then the common command set between the two memories is supported, based on C_SPI_MODE setting. The behavior for the Winbond memory for the command/address/data flow is taken as reference. This is applicable for both the memories. In this mode, Extended SPI command instructions from Numonyx memories are supported. If Dual mode is selected, Dual as well as Standard SPI commands are supported. If Quad mode is selected, Quad, Dual, and Standard SPI commands are supported.

SPI Address Section

If the first entry in SPI DTR matches the SPI command which needs address phase, the core transfers the address on the IO0 (MOSI) line along with the SPI clock, based on the command and memory type.

When C_SPI_MEMORY is 2 (Numonyx memory) and C_SPI_MODE is 1 or 2, then based on the supported command, the IO0, IO1 (IO2, IO3 in Quad mode) lines are used.

When C_SPI_MEMORY is 1 (Winbond memory) and C_SPI_MODE is 1 or 2, based on the supported command, the IO0, IO1 (IO2, IO3 in Quad mode) lines are used to transfer the address bits.

When the C_SPI_MEMORY = 0 (mixed mode memories), then the common command set between these two memories are supported. In this case, based on C_SPI_MODE setting (1 or 2) the behavior for Winbond memory for the address flow is taken as reference. This is applicable for both memories. In this mode, the Extended SPI Command Instructions from Numonyx memories are supported. If C_SPI_MODE is 1, the Dual and Standard SPI commands are supported. If C_SPI_MODE is 2, the Quad, Dual, and Standard SPI commands are supported.

For read mode commands, after transferring the command and address, the core reverts to the input mode, where it stores all the data bytes (applicable on single or Dual or Quad lines as per the command) in the SPI DRR. Therefore, for standard read commands, the IO1_I line is used to store data in the SPI DRR. For Dual mode read command, the IO0_I, IO1_I lines are used to store the data in the SPI DRR and the same applies for Quad mode commands where IO0_I, IO1_I, IO2_I and IO3_I lines are used to store data in the SPI DRR. This behavior should be noted while filling the dummy bytes in the SPI DTR to read the number of bytes from the SPI slave. For example, for commands such as the Fast Read Dual Output (3Bh) command in Winbond, the data is received on 2 lines, for example, on the IO0_I and IO1_I lines. After transmitting the address bits, the core reverts to the input mode, where it stores the data coming on IO0 and IO1 lines in the SPI DRR. This data includes the dummy bytes also. It is your responsibility to add the same number of dummy bytes while filling the SPI DTR and ignore the same number of bytes while reading the SPI DRR contents.

SPI Data Section

If the first entry in SPI DTR matches with the Standard SPI transaction logic in the core, the core transfers the data on IO0 (MOSI) line along with the SPI clock.

When C_SPI_MEMORY is 2 (Numonyx memories) and the C_SPI_MODE (1 or 2) setting, then based on the supported command, the IO0, IO1 (IO2, IO3 in Quad mode) lines are used.

When C_SPI_MEMORY is 1 (Winbond memories) and C_SPI_MODE (1 or 2) setting, based on the supported command, the IO0, IO1 (IO2, IO3 in Quad mode) lines are used to transfer the data bits.

When C_SPI_MEMORY is 0 (mixed mode memories (Winbond and Numonyx)), then the common command set between these two memories is supported. In this case, based on C_SPI_MODE setting (Dual or Quad), the behavior for Winbond memory for the data flow is taken as reference. This is applicable for both memories.

Configure the core carefully and read the section where non-supported commands are listed for each of the memories, while executing any instruction for the Winbond or Numonyx memory. In this mode, the Extended SPI Command Instructions from Numonyx memories are supported. If C_SPI_MODE = 1 is selected, the Dual and Standard SPI commands are supported. If C_SPI_MODE is 2, the Quad, Dual, and Standard SPI commands are supported.

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. At present, the Numonyx memory part N25Q_256_3 data sheet supports only Extended SPI Instructions. Read the data sheet to match the expected behavior of the core in different modes (set by C_SPI_MODE (1 or 2)), when C_SPI_MEMORY is set to 2.

C_SPI_MODE = 1

The core is configured to support Dual and Standard mode SPI commands. The configuration modes and their behavior are described in the following sections:

C_SPI_MEMORY = 0

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 at present, the slave is one of these two memories.

This mode is considered to be a mixed mode 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 23 shows the core behavior.

Table 23: **Core Behavior For SPI Commands 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
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. The C_SPI_MEMORY = 0 mode is mixed mode memory mode. For C_SPI_MODE = 1, the Dual as well as Standard SPI commands are supported. For each command, the base behavior of Winbond memory is taken as default operation mode. For the commands supported only by Numonyx, the Command Error flag is set and the command is not executed. The command set supported in this mode is the common command set between Winbond and Numonyx memories.

C_SPI_MEMORY = 1

This is a dedicated mode and supports only Winbond memories as SPI slave devices. Most of the SPI commands from the Winbond memories are supported. Table 24 shows the core behavior.

Table 24: **Core Behavior For SPI Commands 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

Note:

1. The present core is designed to support the Winbond memory W25Q64BV. See the data sheet for the command, address, dummy bytes and data bytes required for each command and to see how the command, address and data bits operate.

C_SPI_MEMORY = 2

This mode is dedicated mode and supports only Numonyx memories as SPI slave devices. The core supports most of the Dual and Standard SPI commands for Numonyx memories. Table 25 shows the core behavior.

Table 25: **Core Behavior For SPI Commands 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 present core is designed to support the Numonyx memory device N25Q256-3V in this mode for all Dual and Standard mode commands. See the device data sheet for the command, address, dummy bytes and data bytes requirements for each command and for more information on how the command, address and data bits operate.

C_SPI_MODE = 2

The core is configured to support the Quad, Dual and Standard mode SPI commands based on the type of memory used as an SPI slave. The configuration modes and their behavior are described in the following sections.

C_SPI_MEMORY = 0

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 at present, the slave is one of these two memories.

This mode is considered to be mixed mode 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 26 shows the core behavior.

Table 26: **Core Behavior For SPI Commands 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
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. The C_SPI_MEMORY = 0 mode is mixed mode memory mode. In C_SPI_MODE = 2, the Quad, Dual as well as Standard SPI commands are supported. For each command, the base behavior of Winbond memory is taken as default operation mode. For the commands supported only by Numonyx, the Command Error flag is set and the command is not executed. The command set supported in this mode is the common command set between Winbond and Numonyx memories.

C_SPI_MEMORY = 1

This mode is dedicated mode and supports only Winbond memories as SPI slave devices. Most of the SPI commands from the Winbond memories (W25Q64BV) are supported. Table 27 shows the generalized core behavior.

Table 27: **Core Behavior For SPI Commands 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 present core is designed to support the Winbond memory W25Q64BV. See the data sheet for the command, address, dummy bytes and data bytes required for each command and information on how the command, address and data bits operate.

C_SPI_MEMORY = 2

This mode is dedicated mode and supports only Numonyx memories as SPI slave devices. The core supports most of the Quad, Dual, and Standard SPI commands for Numonyx memories (N25Q256-3V). Table 28 shows the core behavior.

Table 28: **Core Behavior for SPI Commands in 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 present core is designed to support the Numonyx memory device N25Q256-3V 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 more information on how the command, address and data bits operate.

Core Behavior in XIP Mode

The mode is when C_TYPE_OF_AXI4_INTERFACE is 1 and C_XIP_MODE is 1. 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 this mode, the AXI4-Lite interface is first used to configure the core with proper CPOL, CPHA modes. The valid modes are 00 and 11. Any other combination causes the core to not accept the AXI4 memory mapped transaction. AXI4-Lite interface is only used for configuration register setting as well as reading the status register. The AXI4 Memory mapped interface is used to read the data from memory with the command configured using AXI4-Lite interface and the address provided by the AXI4 Memory mapped interface. The read channel of AXI4 Memory mapped interface should provide the starting address which is converted into the SPI transactions at the SPI interface by the core. The operation mode of the core is set using the C_SPI_MODE bit while the targeted memory is selected based on the C_SPI_MEMORY. In this case, the targeted memory is single memory and multiple memories are not supported by the core. The target memory can be any memory from Winbond or Numonyx or any other memory which supports the default three read commands. These commands are Fast Read (0x0Bh), Fast Read Dual I/O (0xBBh), Fast Read Quad I/O (0xEBh). When the mode is set by C_SPI_MODE, then the same command is used throughout the operation. The command cannot be changed as this 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 operation of the core is based on the Winbond and Numonyx memory data sheet support for the read command behavior.

Parameters used for this configuration: C_TYPE_OF_AXI4_INTERFACE, C_XIP_MODE, C_SPI_MODE, C_SPI_MEMORY, C_USE_STARTUP.

The core uses the EXT_SPI_CLK as the reference clock for the SPI logic to be operated on. 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 C_SCK_RATIO = 2, which is fixed for this mode, for generating the SPI clock at 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 of AXI4 transactions only. The FIXED transaction results into the error and the core does not accept the transaction and it sets up the transaction error flag in the XIP status register. In this case the targeted memory is Winbond and if the core is configured in Quad mode, then ensure that the QE bit of status register of the Winbond memory is set prior to the booting the core from SPI flash. This should be done through external programming tools for configuring the QE bit of Status Register of Winbond flash.

Standard SPI Mode Transactions

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

SPI master device with or without FIFOs where the slave select vector is asserted manually using SPICR bit(24) assertion

This flow permits the transfer of N number of byte/half-word/word by toggling of the slave select vector once. This is the default mode of operation. Use the following steps to successfully complete an 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 the Control Register of slave SPI core and 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 one write to the DTR at a given time.
   b. In Enhanced mode, the AXI4 interface must generate the FIXED burst transaction only. INCR transaction with length 0 is acceptable, but if the INCR burst is targeted at the FIFO locations (DTR or DRR), then the core behavior is not guaranteed. The INCR transactions are treated as FIXED transactions. To avoid the FIFO overflow or underflow errors, it is recommended that the transmit or receive occupancy register is 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 $\overline{\text{SS}}$ vector and setting both enable bit and master transfer inhibit bit. This initializes SCK and IO0, but inhibits transfer.
7. Write to SPISSR to manually assert $\overline{\text{SS}}$ vector.
8. Write the preceding configuration data to the master SPI device SPICR, 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 interrupt request. Write new data to the master register/FIFOs and slave devices, then clear master transaction inhibit bit to continue the N 8-bit element transfer. 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 a element transfer (or group of transfers), but is stable at least one-half SCK period prior to sampling edge of SCK.
11. Note that there are no idle cycles between each new SPI transaction.
12. Repeat the previous two steps until all data is transferred.
13. Write all ones to the SPISSR or exit the manual slave select assert mode to deassert $\overline{\text{SS}}$ vector while SCK and IO0 are in the idle state.
14. Disable devices as required.

SPI master and slave devices without FIFOs performing one 8-bit/16-bit/32-bit transfer (optional mode)

Use the following steps to successfully complete an SPI transaction.

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

SPI master and slave devices where registers/FIFOs are filled before the SPI transfer is started and multiple discrete 8-bit transfers are performed (optional mode)

Use the following steps to successfully complete an SPI transaction. The slave operation of the core supports the FIXED burst at Transmit or Receive FIFO only. The length of this burst transaction should be based on the C_FIFO_DEPTH 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.

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

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

Dual/Quad SPI Mode Transactions

In this mode, the core needs be configured in Master mode only, or else the 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 and enabling the SPI core to start the transaction.

1. Ensure that C_SPI_MODE is 1 and that the C_SPI_MEMORY parameter is configured for the correct SPI slave memory.
2. Ensure that the instructions which are operated on the required SPI clock, set by C_SCK_RATIO parameter, are listed.
3. Set the C_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 to put the core in master mode; set CPOL, CPHA values, and make sure that the Master Transaction Inhibit bit is set.
6. Write to the IPIER, IPISR register 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 to assert the chip select signal from the core.
10. Write to the SPICR 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 signal is generated, you can still write the data into SPI DTR for further transactions, if the slave select register is un-touched (for example, if the slave is selected).
13. In this case, the SPISSR continues to be asserted; only the SPI clock is stopped. When the DTR is non-empty, then the SPI clock is enabled again and data is transmitted.
14. Step 11, step 12, and step 13 also apply when reading. Fill the DTR with any random data beats while reading the SPI slave memory.
15. After you disable the selected slave with the SPISSR bits set to all 1 (or carrying out Master Transactions Inhibit bit set to 1 in the SPICR), the SPI clock is stopped. If the SPI DTR is filled again, the core considers this as a fresh transaction and checks the first entry in DTR with the supported command.
16. In between new transactions, make sure that the SPI DTR and SPI DRR are reset by writing into the SPICR register bits while the slave is de-selected. This allows these two FIFOs to be reset and the first entry of DTR is compared against the internal decoding logic for command decode.

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

Ensure that you 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. The interrupt is set to indicate the command error, which means that the command does not match any of the supported commands in the core.

Timing Diagrams

Reference diagrams for the core behavior are shown in Figure 21 and Figure 22.

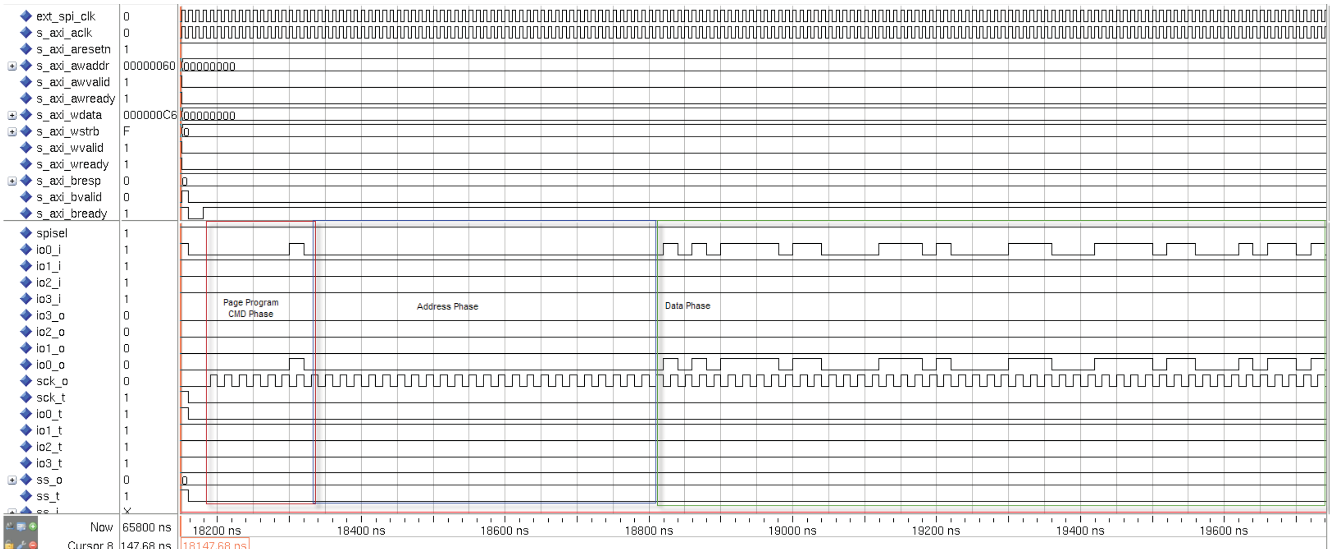


Figure 21: **Waveforms for Page Program (0x02h) Command**

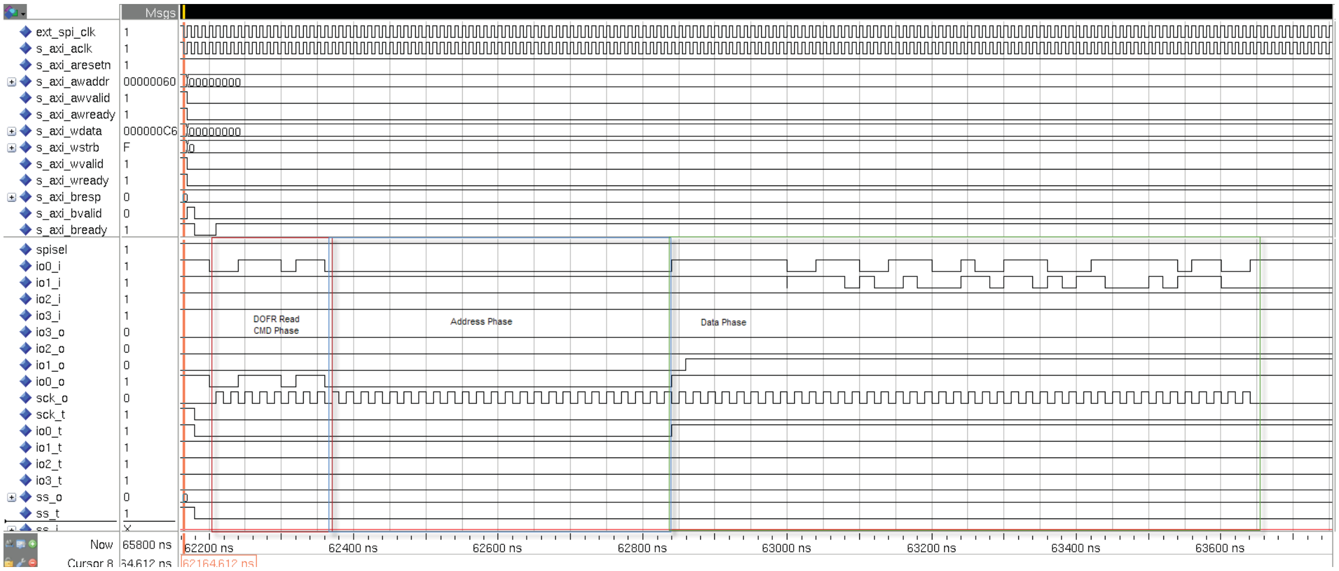


Figure 22: **Waveforms for Dual Output Fast Read (DOFR - 0x3Bh) Command**

Design Implementation

Target Technology

The target FPGA technologies for the core are Zynq™-7000, Artix™-7, Kintex™-7, Virtex-7, Virtex-6, and Spartan-6 FPGA devices.

Device Utilization and Performance Benchmarks

Because the AXI Quad SPI IP 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 IP core resource utilization for various parameter combinations measured with a Spartan-6 FPGA as the target device are detailed in Table 29. The utilization figures should be considered as reference only.

Legacy Mode

Table 29: **Performance and Resource Utilization Benchmarks on a Spartan-6 FPGA (xc6slx45tfgg484-3)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_SPI_MODE	C_SPI_MEMORY	C_FIFO_DEPTH	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	0	0	0	16	16	2	8	128	294	300	184
0	0	0	0	16	16	2	8	202	444	419	190
0	0	0	0	256	16	2	8	254	572	575	162
0	0	1	1	16	16	2	8	207	431	401	167
0	0	1	1	256	16	2	8	181	429	430	181
0	0	1	2	16	16	2	8	251	562	576	141
0	0	1	2	256	16	2	8	248	563	599	160
0	0	2	1	16	16	2	8	202	432	437	180
0	0	2	1	256	16	2	8	201	439	459	140
0	0	2	2	16	16	2	8	273	562	604	146
0	0	2	2	256	16	2	8	243	571	635	155

The core resource utilization for various parameter combinations measured with a Virtex-6 FPGA as the target device are detailed in Table 30. The utilization figures should be considered as reference only.

Table 30: **Performance and Resource Utilization Benchmarks on a Virtex-6 FPGA (xc6vlx130tff1156-1)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_SPI_MODE	C_SPI_MEMORY	C_FIFO_DEPTH	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	0	0	0	0	16	2	8	127	276	239	222
0	0	0	0	16	16	2	8	245	439	382	231
0	0	0	0	256	16	2	8	278	564	568	213
0	0	1	1	16	16	2	8	197	429	398	232
0	0	1	1	256	16	2	8	222	428	379	216
0	0	1	2	16	16	2	8	289	551	561	225
0	0	1	2	256	16	2	8	293	551	549	205
0	0	2	1	16	16	2	8	239	431	405	236
0	0	2	1	256	16	2	8	247	433	417	222
0	0	2	2	16	16	2	8	317	554	567	205
0	0	2	2	256	16	2	8	308	556	579	221

The core resource utilization for various parameter combinations measured with an Artix-7 and Zynq (based on Artix-7 FPGA logic) as the target devices are detailed in Table 31. The utilization figures should be considered as reference only.

Table 31: **Performance and Resource Utilization Benchmarks on an Artix-7 FPGA (XC7A100T-3)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_SPI_MODE	C_SPI_MEMORY	C_FIFO_DEPTH	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	0	0	0	0	16	2	8	148	273	226	184
0	0	0	0	16	16	2	8	227	437	367	170
0	0	0	0	256	16	2	8	290	561	548	150
0	0	1	1	16	16	2	8	229	427	362	201
0	0	1	1	256	16	2	8	295	549	530	130
0	0	1	2	16	16	2	8	208	428	385	169
0	0	1	2	256	16	2	8	295	551	549	136
0	0	2	1	16	16	2	8	226	431	391	154
0	0	2	1	256	16	2	8	275	554	582	171
0	0	2	2	16	16	2	8	209	433	417	163
0	0	2	2	256	16	2	8	302	555	598	165

The core resource utilization for various parameter combinations measured with a Virtex-7 FPGA as the target device are detailed in Table 32. The utilization figures should be considered as reference only.

Table 32: **Performance and Resource Utilization Benchmarks on a Virtex-7 FPGA (xc7v285tffg784-3)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_SPI_MODE	C_SPI_MEMORY	C_FIFO_DEPTH	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	0	0	0	16	16	2	8	175	287	287	225
0	0	0	0	16	16	2	8	242	451	429	235
0	0	0	0	256	16	2	8	355	579	590	215
0	0	1	1	16	16	2	8	264	444	435	235
0	0	1	1	256	16	2	8	378	569	589	218
0	0	1	2	16	16	2	8	239	445	437	225
0	0	1	2	256	16	2	8	379	570	600	215
0	0	2	1	16	16	2	8	268	450	456	236
0	0	2	1	256	16	2	8	390	575	632	222
0	0	2	2	16	16	2	8	303	452	468	215
0	0	2	2	256	16	2	8	396	576	642	221

The core resource utilization for various parameter combinations measured with a Kintex-7 and Zynq (based on Kintex-7 FPGA logic) as the target devices are detailed in Table 33. These utilization figures should be considered as reference only.

Table 33: **Performance and Resource Utilization Benchmarks on a Kintex-7 FPGA (xc7k325tffg900-3)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	**C_XIP_MODE**	**C_SPI_MODE**	**C_SPI_MEMORY**	**C_FIFO_DEPTH**	**C_SCK_RATIO**	**C_NUM_SS_BITS**	**C_NUM_TRANSFER_BITS**	**Slices**	**Slice Flip-Flops**	**LUTs**	**Fmax (MHz)**
0	0	0	0	16	16	2	8	142	287	289	314
0	0	0	0	16	16	2	8	244	451	431	250
0	0	0	0	256	16	2	8	318	579	599	231
0	0	1	1	16	16	2	8	234	444	431	258
0	0	1	1	256	16	2	8	313	569	613	237
0	0	1	2	16	16	2	8	234	445	436	257
0	0	1	2	256	16	2	8	327	570	613	227
0	0	2	1	16	16	2	8	236	450	463	253
0	0	2	1	256	16	2	8	343	575	636	216
0	0	2	2	16	16	2	8	247	452	484	237
0	0	2	2	256	16	2	8	351	576	643	201

XIP Mode

The AXI Quad SPI core resource utilization for various parameter combinations in XIP mode measured with a Spartan-6 FPGA as the target device are detailed in Table 34. The utilization figures should be considered as reference only.

Table 34: **Performance and Resource Utilization Benchmarks on a Spartan-6 FPGA (xc6slx45tfgg484-3)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_SPI_MODE	C_SPI_MEMORY	C_FIFO_DEPTH	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
1	1	0	1	64	2	1	8	240	509	599	140
1	1	0	2	64	2	1	8	236	512	606	140
1	1	1	1	64	2	1	8	252	521	640	143
1	1	1	2	64	2	1	8	256	517	607	145
1	1	2	1	64	2	1	8	258	516	628	147
1	1	2	2	64	2	1	8	243	515	608	140

The AXI Quad SPI core resource utilization for various parameter combinations in XIP mode measured with a Virtex-6 FPGA as the target device are shown in Table 35. The utilization figures should be considered as reference only.

Table 35: **Performance and Resource Utilization Benchmarks on a Virtex-6 FPGA (xc6vlx130tff1156-1)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_SPI_MODE	C_SPI_MEMORY	C_FIFO_DEPTH	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
1	1	0	1	64	2	1	8	239	507	491	205
1	1	0	2	64	2	1	8	254	510	507	195
1	1	1	1	64	2	1	8	282	520	549	141
1	1	1	2	64	2	1	8	259	516	518	125
1	1	2	1	64	2	1	8	262	512	515	173
1	1	2	2	64	2	1	8	244	513	513	194

The AXI Quad SPI IP core resource utilization for various parameter combinations in XIP mode measured with a Kintex-7 and Zynq (based on Kintex-7 FPGA logic) as the target devices are detailed in Table 36. The utilization figures should be considered as reference only.

Table 36: **Performance and Resource Utilization Benchmarks on a Kintex-7 FPGA (xc7k325tffg900-3)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_SPI_MODE	C_SPI_MEMORY	C_FIFO_DEPTH	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
1	1	0	1	64	2	1	8	295	545	624	225
1	1	0	2	64	2	1	8	297	548	618	278
1	1	1	1	64	2	1	8	330	557	679	232
1	1	1	2	64	2	1	8	298	553	651	187
1	1	2	1	64	2	1	8	303	549	634	217
1	1	2	2	64	2	1	8	300	550	630	233

The AXI Quad SPI IP core resource utilization for various parameter combinations in XIP mode measured with a Virtex-7 FPGA as the target device are detailed in Table 37. The utilization figures should be considered as reference only.

Table 37: **Performance and Resource Utilization Benchmarks on a Virtex-7 FPGA (xc7v285tffg784-3)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_SPI_MODE	C_SPI_MEMORY	C_FIFO_DEPTH	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
1	1	0	1	64	2	1	8	281	545	611	225
1	1	0	2	64	2	1	8	295	548	612	278
1	1	1	1	64	2	1	8	311	557	683	232
1	1	1	2	64	2	1	8	297	553	644	187
1	1	2	1	64	2	1	8	298	549	641	217
1	1	2	2	64	2	1	8	297	550	625	233

The AXI Quad SPI IP core resource utilization for various parameter combinations in XIP mode measured with an Artix-7 and Zynq (based on Artix-7 FPGA logic) as the target devices are detailed in Table 38. The utilization figures should be considered as reference only.

Table 38: **Performance and Resource Utilization Benchmarks on an Artix-7 FPGA (XC7A100T-3)**

Parameter Values (other parameters at default values)								Device Resources			Performance
C_TYPE_OF_AXI4_INTERFACE	C_XIP_MODE	C_SPI_MODE	C_SPI_MEMORY	C_FIFO_DEPTH	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
1	1	0	1	64	2	1	8	249	506	486	127
1	1	0	2	64	2	1	8	246	509	502	109
1	1	1	1	64	2	1	8	264	519	568	123
1	1	1	2	64	2	1	8	260	515	512	147
1	1	2	1	64	2	1	8	255	511	516	114
1	1	2	2	64	2	1	8	256	512	506	144

Performance Comparison

The commands in Table 39 are used to provide a comparison between the AXI Quad SPI v1.00 core and AXI Quad SPI v2.00a core. These commands are randomly chosen and using simulation waveforms for writing 256 bytes the time taken for completion is calculated. The time mentioned here is from the assertion of slave select - then filling the DTR of Master, enabling the transaction, waiting till the transaction is over - to the deassertion of the slave select line. In reality this time might be further reduced based on the way the DTR is filled.

Table 39: **Performance Comparison**

Command	From	To	AXI Quad v1.00	AXI Quad v2.00
DIOFR	Slave Select assertion	Slave Select de-assertion	58080 ns	21810 ns
QOFR			22400 ns	6580 ns

System Performance

To measure the system performance (F_{MAX}) of this core, this core was added to a Virtex-6 FPGA system and a Spartan-6 FPGA system as the device under test (DUT) (Figure 23). 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 this core is combined with other designs in the system, the FPGA resources of the design and timing usage vary from the results reported here. The target FPGA was filled with logic to drive the LUT and block RAM utilization to approximately 60% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target FMAX numbers are shown in Table 40.

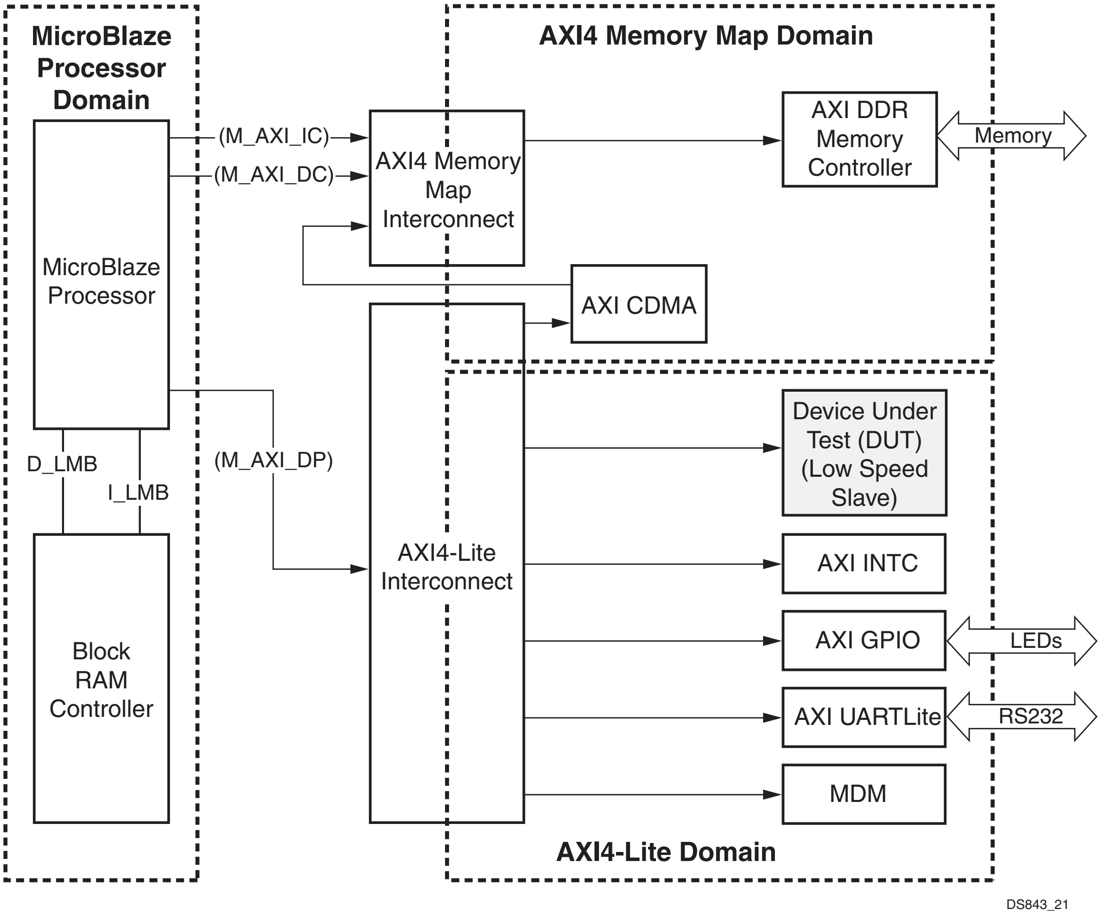


Figure 23: **Virtex-6 and Spartan-6 Devices F_{MAX} Margin System**

Table 40: **System Performance**

Target FPGA	Target F_{MAX} (MHz)		
	AXI4	AXI4-Lite	MicroBlaze™
xc6slx45t[1]	90 MHz	120 MHz	80
xc6vlx240t[2]	135 MHz	180 MHz	135

Notes:

1. Spartan-6 FPGA LUT utilization: 60%; Block RAM utilization: 70%; I/O utilization: 80%; MicroBlaze Controller not AXI4 interconnect; AXI4 interconnect configured with a single clock of 120MHz.
2. Virtex-6 FPGA LUT utilization: 70%; Block RAM utilization: 70%; I/O utilization: 80%.

The target F_{MAX} is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

Specification Exceptions

Exceptions from the Motorola M68HC11-Rev. 4.0 Reference Manual

1. A slave mode-fault error interrupt 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 $\overline{\text{SS}}$ signals are required to be routed between SPI devices internally to the FPGA. This is because toggling of the $\overline{\text{SS}}$ signal is utilized in slaves to minimize FPGA resources.
4. Manual control of the $\overline{\text{SS}}$ signals is provided by setting bit(7) in the SPICR register. When the device is configured as a master and is enabled and 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 $\overline{\text{SS}}$ vector.
5. A control bit is provided to inhibit master transfers. This bit is effective in any master mode, but its main utility is for manual control of the $\overline{\text{SS}}$ 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 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 the position 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 a generic C_NUM_TRANSFER_BITS.
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, in this implementation run time configuration of the baud rate is not possible. Furthermore, in addition to 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 mode 0 and mode 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 higher speed of 50 MHz (most of the instructions supports this speed), the core should be configured in C_SCK_RATIO = 2 mode (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 falling edge and the next consecutive rising edge. As per the M68HC11 document, the master should register data on each rising edge of SCK in SPI mode 1 and 3. Note that the data registering mechanism when C_SCK_RATIO =2, follows a different pattern than specified in the standard. Although this is applicable to the data registering mechanism in IP 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 AXI Quad SPI IP core is configured in slave mode (C_SPI_MODE = 0), the data in the core is registered on the SCK rising edge + 1 AXI clock signal. Internally, this data is registered on the next rising edge of AXI. The core changes the data on the SCK falling edge + AXI clock cycle.

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 at present is based on the specific memory parts from Winbond (W25Q80) and Numonyx(N25Q256). If you want to test the core with some other memory parts, ensure that you understand the internal command decoding logic and its bit positions for correct operation of the core. Also, you can 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.
4. It is recommended that you use dedicated memory 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, the core supports a wide range of command set for the respective memory and gives better performance.
5. In XIP mode, there are several clock domain crossing signals present between the AXI Lite, AXI4 Full 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.

Common Supported Commands for Dual SPI and Mixed Mode Memory Mode

For the setup when C_SPI_MODE is 1 and C_SPI_MEMORY is 0, the core supports the commands listed in Table 41, which are identical (in terms of command, address and data behavior) in both Winbond and Numonyx memories. See the memory data sheet for the commands. The commands which are not supported in this mode include Fast Read, Dual I/O Fast Read, Dual Output Fast Read. These commands are not supported by the core in the mixed mode because the dummy bytes or dummy cycles are different in Winbond and Numonyx memories. Also in the mixed mode the volatile configuration register of Numonyx is not supported as this command is not present in the Winbond memory data sheet. See Unsupported Commands for Dual/Quad SPI Mode and Winbond or Numonyx Memory for a list of unsupported commands.

Table 41: **Supported Commands in Dual SPI Mode and Mixed Memory Mode**

Opcode (h)	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

Table 41: **Supported Commands in Dual SPI Mode and Mixed Memory Mode** ***(Cont'd)***

Opcode (h)	Command Description
C7	Bulk Erase
D8	Sector Erase

Common Supported Commands for Quad SPI and Mixed Memory Mode

For the setup when C_SPI_MODE is 2 and C_SPI_MEMORY is 0, the core supports the commands listed in Table 42, which are identical (in terms of command, address and data behavior) in Winbond and Numonyx memories. See the memory data sheet for the commands. 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, Quad I/O Fast Read. These commands are not supported by the core in the mixed mode because the dummy bytes or dummy cycles are different in Winbond and Numonyx memories. Also in the mixed mode, the volatile configuration register of Numonyx is not supported as this command is not present in the Winbond memory data sheet. See Unsupported Commands for Dual/Quad SPI Mode and Winbond or Numonyx Memory for a list of unsupported commands.

Table 42: **Supported Commands in Quad Mode and Mixed Mode Memory**

Opcode (h)	Command Description
01	Write Status Register
02	Page Program
03	Read Data Bytes
04	Write Disable
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 No.
9F	Read JEDEC ID No.
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 32-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 (AB h) - This command supports Release Power Down/High Performance Mode OR reading the Device ID with different combination of dummy bytes. The core supports only 'Release Power Down/High Performance Mode' where only one command byte is required to put in DTR. The other mode of 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 the 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.

Support

Xilinx provides technical 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*.

Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado Design Suite and ISE Design Suite Embedded Edition tools 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.

Reference Documents

The following documents contain reference information important to understanding the AXI Quad SPI core design:

1. Motorola M68HC11-Rev. 4.0 Reference Manual
2. *Motorola MPC8260 PowerQUICC II™ Users Manual* 4/1999 Rev. 0
3. AMBA AXI4-Stream Protocol Specification
4. LogiCORE IP AXI Lite IPIF (axi_lite_ipif) Data Sheet (DS765)
5. *AXI Interconnect IP Data Sheet* (DS768)
6. Winbond memory data sheet (W25Q64BV)
7. Numonyx memory data sheet (N25Q256 - 3v)
8. 7 Series FPGAs Overview (DS180)
9. Virtex-6 Family Overview (DS150)
10. Spartan-6 Family Overview (DS160)
11. 7 Series FPGAs Configuration User Guide (UG470)
12. Virtex-6 FPGA Configuration User Guide (UG360)
13. Spartan-6 FPGA Configuration User Guide (UG380)

Revision History

Date	Version	Revision
6/22/11	1.0	Initial Xilinx Release.
10/19/11	1.1	Updated for Dual and Quad SPI modes. ISE Software Release 13.3.
04/24/12	2.0	Updated with new version change. Added Legacy, Enhanced and XIP modes of operation. Added Vivado support.
07/25/12	2.1	Updated for 14.2/2012.2. Byte access in XIP mode not supported; FIFO depth updated.
12/18/12	2.2	• Updated core to v2.00a, and tool versions to 14.4 and 2012.4. • Added addressing-mode support details for Winbond and Numonyx Memory. • Updated Figure 3.

Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: http://www.xilinx.com/warranty.htm#critapps.