LogiCORE IP AXI Quad Serial Peripheral Interface (AXI Quad SPI) (v1.00a)

DS843 October 19, 2011 **Product Specification**

Introduction

The AXI Quad Serial Peripheral Interface connects the AXI4 interface to SPI slave devices that support the Standard, Dual or Quad SPI protocols. This core provides a serial interface to SPI slave devices such as the SPI serial flash from Winbond/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. The Dual and Quad SPI behavior is based on the Winbond [Ref 6] and Numonyx [Ref 7] SPI memory data sheets.

Features

- Supports the AXI4-Lite interface which is based on the AXI4-Lite specification [Ref 3]
- Connects as a 32-bit AXI4-Lite slave
- Supports configurable SPI modes:
  - Standard SPI mode
  - Dual SPI mode
  - Quad SPI mode
- In Standard SPI mode, supports four signal interfaces:
  - IO0 (MOSI - in Standard SPI mode)
  - IO1 (MISO - in Standard SPI mode)
  - SCK
  - $\overline{\text{SS}}$
- In Dual SPI mode, supports four signal interfaces: IO0, IO1, SCK and $\overline{\text{SS}}$
- In Quad SPI mode, supports six signal interfaces: IO0, IO1, IO2, IO3, SCK and $\overline{\text{SS}}$
- Supports slave select ($\overline{\text{SS}}$) bit for each slave on the SPI bus
- Supports programmable clock phase and polarity
- Supports continuous transfer mode for automatic scanning of a peripheral
- Supports configurable FIFO depth (16 or 256 elements deep in Dual/Quad/Standard SPI mode)

LogiCORE™ IP Facts	
Core Specifics	
Supported Device Family [1]	Zynq™-7000, Virtex-7[2], Kintex-7[2], Artix-7 [2], Virtex-6[3], Spartan-6[4]
Supported User Interfaces	AXI4-Lite
	Resources
Configuration	See Tables 22 to 26.
Provided with Core	
Documentation	Product Specification
Design Files	VHDL
Example Design	N/A
Test Bench	N/A
Constraints File	N/A
Simulation Model	N/A
Tested Design Tools [5]	
Design Entry Tools	XPS software
Simulation	Mentor Graphics ModelSim
Synthesis Tools	XST
Support	
Provided by Xilinx, Inc.	

Notes:

1. For a complete list of supported derivative devices, see the IDS Embedded Edition Derivative Device Support.
2. For more information on 7 series devices, see [Ref 8].
3. For more information on Virtex-6 devices, see [Ref 9].
4. For more information on Spartan-6 devices, see [Ref 10].
5. For a listing of the supported tool versions, see the ISE Design Suite 13: Release Note Guide.

Supported Features

In Standard SPI mode:

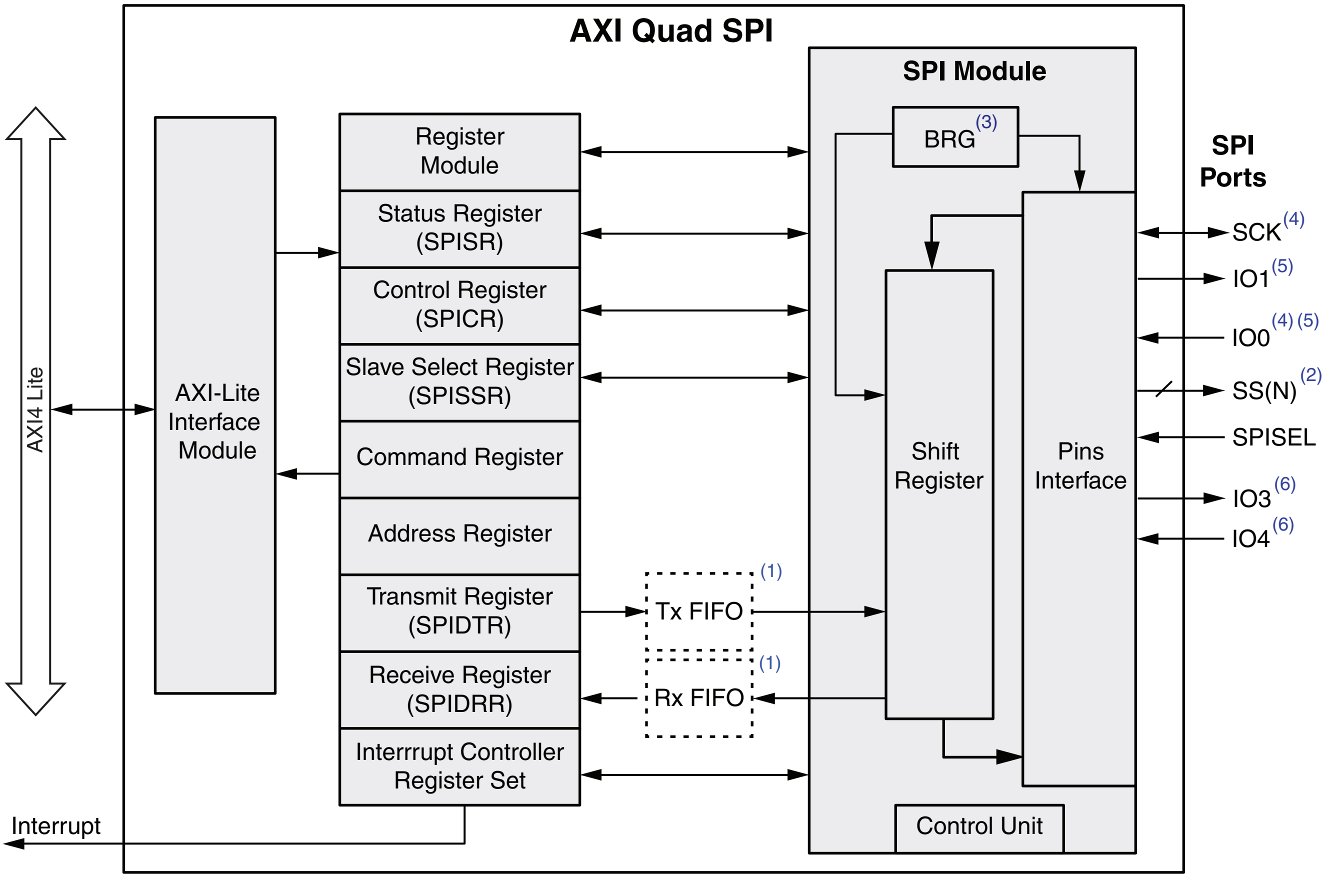
- Master and slave SPI mode
- MSB/LSB first transactions
- SPI transfer length of 8-bits, 16-bits, or 32-bits
- Local loopback capability for testing
- Multiple master and multiple slave environments
- Optional 16 or 256 element deep (an element is a byte, a half-word or a word) transmit and receive FIFOs
- Full-duplex operation

In Dual/Quad SPI mode:

- Master SPI mode
- Manual slave select mode only
- 'MSB first' transaction as per requirements of Quad SPI memories
- SPI transfer length of 8-bits only
- Configurable Dual/Quad SPI mode
- Up to 256 element deep (including option for 16 element deep) (an element is a byte) transmit and receive FIFOs

Functional Description

The top level block diagram for the AXI Quad SPI core is shown in Figure 1.



Notes:
1. The width of Tx FIFO, Rx FiFO, and Shift Register depends on the value of the generic, C_NUM_TRANSER_BITS.
2. The width of SS depends on the value of the generic C_NUM_SS_BITS.
3. BRG (Buad Rate Generator)
4. These ports will be input ports in case the C_SPI_MODE = 0 and the SPI core is cofigured in the slave mode.
5. These ports will be available as MISO and MOSI in case the C_SPI_MODE = 0. When the core C_SPI_MODE = 1. these bits are considered as IO0 and IO1 and will become bi-directional based on the instruction used and the control register bit configuration.
6. These ports will be available only when the core is configured as Quad SPI mode through C_SPI_MODE = 2.

DS843_01

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

When configured in Standard SPI mode, the AXI Quad SPI core 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 upon the control register settings and command used.

The core supports Manual Slave Select as the default mode of operation for slave select mode. This mode allows manual control of the slave select line using 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, the slave select line must be toggled before the start of each new transfer.

In Automatic Slave Select mode, the slave select line is toggled automatically after each element transfer. This mode, supported only in Standard SPI mode, is described in more detail in SPI Protocol with Automatic Slave Select Assertion.

The core functionality is divided into Standard SPI mode and Dual/Quad SPI mode. The functionality for each mode differs in the way the slave memory works. The mode functionality is described in the following sections.

Standard SPI Mode

The standard SPI mode is selected when C_SPI_MODE is assigned a value of 0.

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 upon 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 to use a FIFO depth of 256, as this is the most suitable depth for page program instructions.
2. The valid values for the C_FIFO_DEPTH parameter in this mode are 0, 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 core supports data transfer mode. When configured as master, the transfer continues until the data is available in transmit register or 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 Data Transmit Register (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 Standard SPI mode.

Dual/Quad SPI Mode

The Dual SPI mode is selected when C_SPI_MODE has the value1.

The properties related to the FIFO are:

1. The choice of depth of the FIFO is based upon the C_FIFO_DEPTH parameter. The transmit and receive FIFO depth is limited to 16 or 256 only.
2. The valid values for C_FIFO_DEPTH are 16 or 256.

The behavior of the ports is:

- For standard SPI 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 instruction which has been selected by setting the control register bits.

The Quad SPI mode is selected when C_SPI_MODE has the value 2.

The behavior of the ports in this mode is:

- For standard SPI instruction, the IO0 and IO1 pins are unidirectional and function the same as the 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 the respective SPI mode (Dual or Quad) while the core continues supporting 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 determine the I/O pin availability.

In these modes, the receive and transmit FIFO are part of the core and the depth of these two FIFOs is decided by the setting of the C_FIFO_DEPTH parameter. The allowed values for this parameter in these modes are 16 or 256. The width of the FIFO is 8 bits because the page size of the SPI slave memories is 8 bits only.

Common Information for Both 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, the means to detect all possible conflicts are not implemented with this interface standard. To eliminate conflicts, the system software is required to arbitrate bus control.

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

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

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.

SPI Register Module

The SPI Register Module includes all memory mapped registers, shown in Figure 1. It interfaces to the AXI4-Lite interface, and consists of Status register, Control register, N-bit Slave Select register ($N \leq 32$) and a pair of transmit and receive registers.

Interrupt Controller Register Set Module

The Interrupt Controller Register Set Module consists of interrupt related registers: 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 the Standard SPI mode, this module is the center of the SPI operation.

Optional FIFOs

When enabled by the parameter C_FIFO_DEPTH, the transmit and receive FIFO are implemented on both the transmit and receive paths. The width of the transmit and receive FIFO are the same and depend on the generic

C_NUM_TRANSFER_BITS. When the FIFOs are enabled, their depth is fixed at 256 in Standard SPI mode. In the Dual or Quad SPI mode, the FIFO depth is 256 locations (bytes).

Start-Up Module

The STARTUP is a primitive in the Xilinx FPGA. This primitive can be used in the design, after the FPGA configuration. For more information on the use of this primitive, see the targeted FPGA user guide (see [Ref 11], [Ref 12] or [Ref 13]). This primitive can be included in the design with the C_USE_STARTUP parameter set 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, 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.

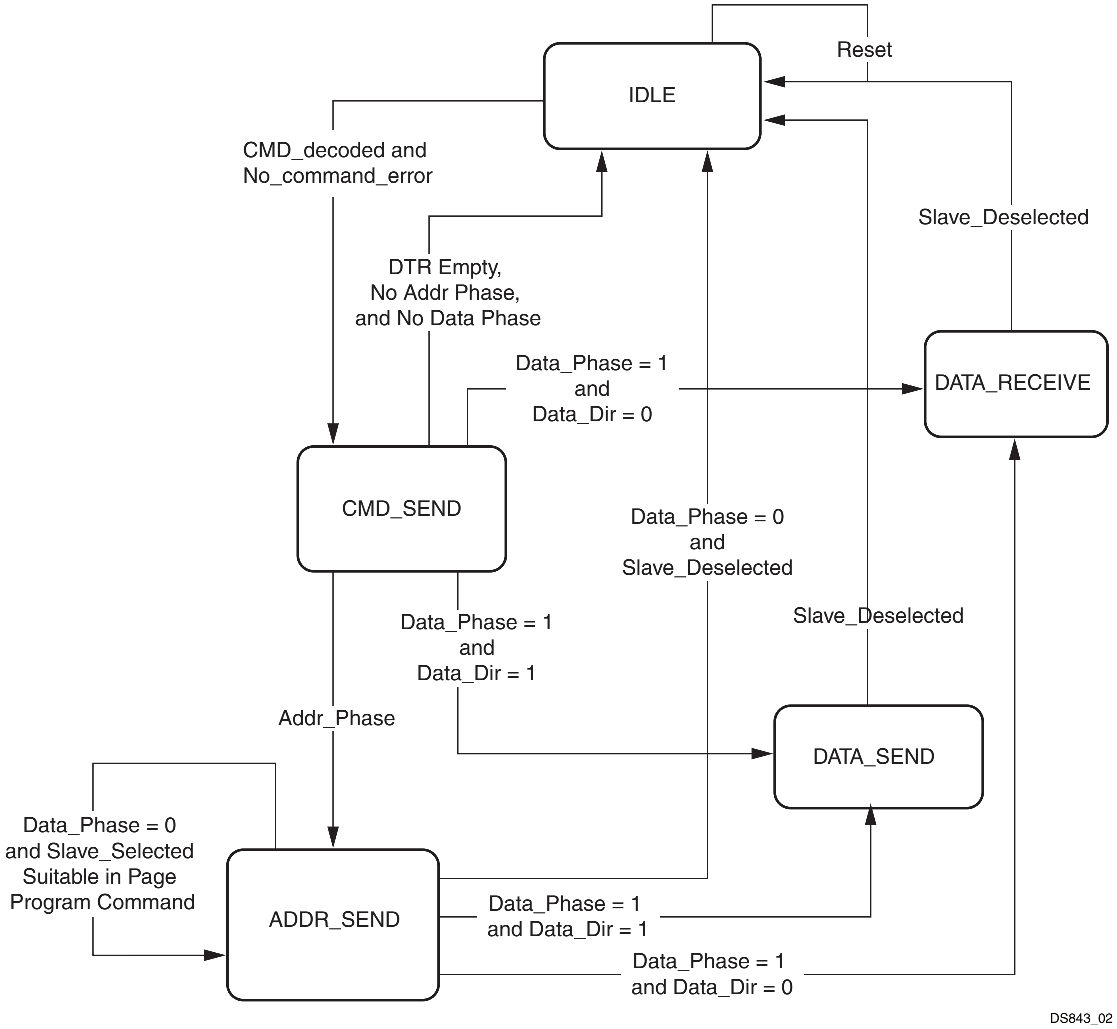


Figure 2: **Logical State Diagram for C_SPI_MODE = 1 and C_SPI_MEMORY = 0 or 1**

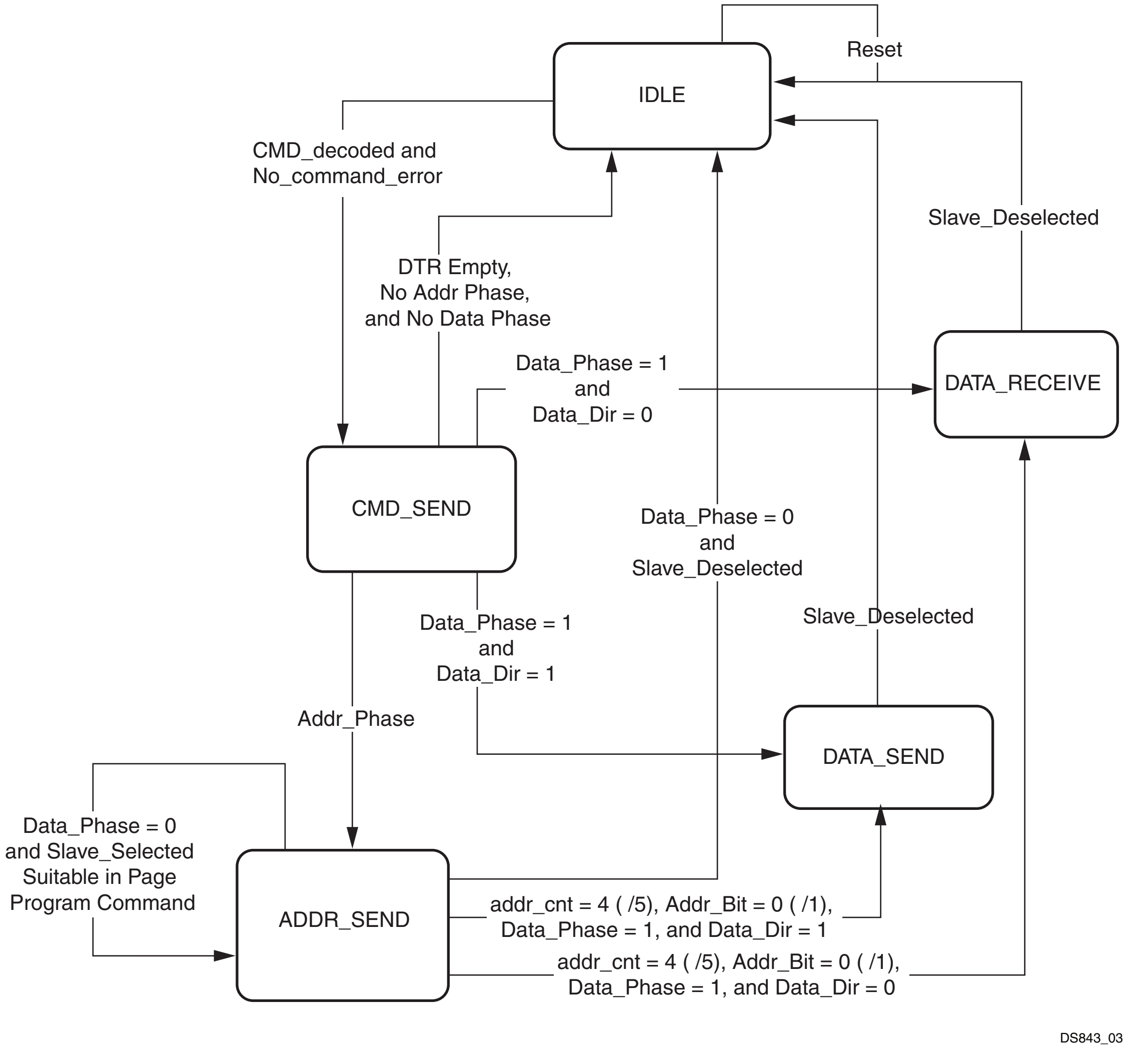


Figure 3: **Logical State Diagram for C_SPI_MODE = 2 and C_SPI_MEMORY = 0 or 1**

Design Parameters

To design an AXI Quad SPI IP core that is uniquely tailored for the 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 [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, 7-series, zynq	virtex6	string
AXI Parameters					
G2	AXI Base Address	C_BASEADDR	Valid Address [(1)]	None [(2)]	std_logic_vector
G3	AXI High Address	C_HIGHADDR	Valid Address [(1)]	None [(2)]	std_logic_vector
G4	AXI Address Bus Width	C_S_AXI_ADDR_WIDTH	32	32	integer
G5	AXI Data Bus Width	C_S_AXI_DATA_WIDTH	32	32	integer
AXI Quad SPI Core Parameters					
G6	Receive and transmit FIFO depth	C_FIFO_DEPTH	0, 16, 256 0 = FIFOs are not included in the design (allowed only in Standard SPI mode) 16 = FIFOs are included in the design (allowed in Standard/Dual/Quad SPI mode) 256 = FIFOs are included in the design (allowed in Standard/Dual/Quad SPI mode)	256 [(3)]	integer
G7	SPI clock frequency ratio	C_SCK_RATIO	2 [(4)], 4, 8, Nx16 for N = 1, 2, 3, ...128	16	integer
G8	Total number of slave select bits	C_NUM_SS_BITS	1 - 32	1	integer
G9	Select number of transfer bits as 8	C_NUM_TRANSFER_BITS	8, 16, 32	8 [(5)]	integer
G10	SPI Modes	C_SPI_MODE	0, 1, 2 0 = Standard SPI mode 1 = Dual SPI mode 2 = Quad SPI mode	0 [(6)]	integer
G11	Use STARTUP primitive	C_USE_STARTUP	0, 1 0 = Exclude the STARTUP primitive from core 1 = Include the STARTUP primitive in the core	0 [(7)]	integer

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

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G12	The SPI memory device used as SPI slave	C_SPI_MEMORY	0, 1, 2 0 = Mixed mode memory 1 = Winbond memories are used as SPI slaves 2 = Numonyx memories are used as SPI slaves	1 (8) (9)	integer

Notes:

1. The range C_BASEADDR to C_HIGHADDR is the address range for the AXI Quad SPI core. 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 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.
2. No default value is specified to ensure that an actual value appropriate to the system is set. You must set the values.
3. In Standard SPI mode, the option to include or exclude the FIFO in the design is allowed. In the Standard SPI mode, if the FIFO is included in the design, the FIFO depth is restricted to 16 or 256 only. In the Dual or Quad SPI mode, this parameter must to be set to 256.
4. The C_SCK_RATIO = 2 parameter is not supported when the AXI Quad SPI core is configured as a slave in Standard SPI mode. See Assigning the C_SCK_RATIO Parameter for information on using this parameter. The SPI clock is generated from the AXI clock of the core.
5. C_NUM_TRANSFER_BITS is allowed to vary between 8, 16 or 32 only when the IP is configured in the Standard SPI mode. When the core operates in Dual or Quad SPI mode, the transfer size is 8 bits because the present SPI slave devices that support the Dual or Quad mode are 8 bits.
6. C_SPI_MODE indicates the operating mode of this device. C_SPI_MODE = 0 means Standard SPI mode, C_SPI_MODE = 1 means Dual SPI mode and C_SPI_MODE = 2 means Quad SPI mode. In C_SPI_MODE = 0, the present AXI Quad SPI core functionality is inferred. This parameter determines the number of ports and their operating mode.
7. Use of the STARTUP primitive is applicable only when the core is intended for use in the Master mode. See the user guide of the 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.
8. The core supports most of the command set for the Winbond and Numonyx memories. When the S_SPI_MEMROY 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 upon the C_SPI_MODE in which the core is configured and the target memory, the command set does vary. Although this core supports most of the commands, ensure that you 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 to use the dedicated memories in place of mixed mode memories while using the core in Dual or Quad SPI mode.
9. This parameter is applicable only when the C_SPI_MODE = 1 or 2. When the parameter C_SPI_MODE = 0, the core is configured in the Standard SPI mode.

I/O Signals

The I/O signals are listed and described in Table 2.

Table 2: **I/O Signal Descriptions**

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
AXI Write Address Channel Signals					
P3	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.
P4	S_AXI_AWVALID	AXI	I	-	Write address valid. This signal indicates that a valid write address and control information are available.
P5	S_AXI_AWREADY	AXI	O	0	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
AXI Write Channel Signals					
P6	S_AXI_WDATA [(C_S_AXI_DATA_WIDTH - 1):0]	AXI	I	-	Write data
P7	S_AXI_WSTB [((C_S_AXI_DATA_WIDTH/8) - 1):0]	AXI	I	-	Write strobes. This signal indicates which byte lanes to update in memory.
P8	S_AXI_WVALID	AXI	I	-	Write valid. This signal indicates that valid write data and strobes are available.
P9	S_AXI_WREADY	AXI	O	0	Write ready. This signal indicates that the slave can accept the write data.
AXI Write Response Channel Signals					
P10	S_AXI_BRESP[1:0]	AXI	O	0	Write response. This signal indicates the status of the write transaction 00 - OKAY (normal response) 10 - SLVERR (error response) 11 - DECERR (not issued by core)
P11	S_AXI_BVALID	AXI	O	0	Write response valid. This signal indicates that a valid write response is available.
P12	S_AXI_BREADY	AXI	I	-	Response ready. This signal indicates that the master can accept the response information.
AXI Read Address Channel Signals					
P13	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.
P14	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.
P15	S_AXI_ARREADY	AXI	O	1	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.

Table 2: **I/O Signal Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Read Data Channel Signals					
P16	S_AXI_RDATA [(C_S_AXI_DATA_WIDTH - 1):0]	AXI	O	0	Read data
P17	S_AXI_RRESP[1:0]	AXI	O	0	Read response. This signal indicates the status of the read transfer. 00 - OKAY (normal response) 10 - SLVERR (error condition) 11 - DECERR (not issued by core)
P18	S_AXI_RVALID	AXI	O	0	Read valid. This signal indicates that the required read data is available and the read transfer can complete.
P19	S_AXI_RREADY	AXI	I	-	Read ready. This signal indicates that the master can accept the read data and response information.
SPI Slave Device Interface Signals					
Global System Signals From the Core					
P20	IP2INTC_Irpt	SPI	O	0	Interrupt control signal from SPI
SPI Interface Signals					
P21	SCK_I	SPI	I	-	SPI bus clock input. This signal is available only when the core is configured in Standard SPI slave mode.
P22	SCK_O	SPI	O	0	SPI bus clock output
P23	SCK_T	SPI	O	1	3-state enable for SPI bus clock. Active low.
P24	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.
P25	SS_O[(C_NUM_SS_BITS - 1):0]	SPI	O	1	Output one-hot encoded, active low slave select vector of length n
P26	SS_T	SPI	O	1	3-state enable for slave select. Active low.
Standard (and Dual) SPI Mode Signals					
P27	IO0_I	SPI	I	-	Behaves similar to Master Output Slave Input (MOSI) input
P28	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 upon certain instructions.
P29	IO0_T	SPI	O	1	3-state enable master output slave input. Active low
P30	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.

Table 2: **I/O Signal Descriptions *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P31	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 upon certain instructions.
P32	IO1_T	SPI	O	1	3-state enable master input slave output. Active low.
P33	SPISEL [(1)]	SPI	I	1	Local SPI slave select active low input. This is an input signal when the core is configured in the Standard SPI slave mode. Must be set to 1 in master mode.
Quad Mode SPI Signals [(2)]					
P34	IO2_I	SPI	I	-	IO2 input based upon commands used. This signal is available only in Quad SPI mode.
P35	IO2_O	SPI	O	-	IO2 output based upon commands used. This signal is available only in Quad SPI mode.
P36	IO2_T	SPI	O	1	3-state enable IO2. This signal is available only in Quad SPI mode. Active low.
P37	IO3_I	SPI	I	-	IO3 input based upon commands used. This signal is available only in Quad SPI mode.
P38	IO3_O	SPI	O	-	IO3 output based upon commands used. This signal is available only in Quad SPI mode.
P39	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 the AXI Quad SPI core 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

The dependencies between the AXI Quad SPI core design parameters and I/O signals are described in Table 3.

Table 3: **Parameters - Signal Dependencies**

Generic or Port	Name	Affects	Depends	Relationship
Design Parameters				
G4	C_S_AXI_ADDR_WIDTH	P3, P13	-	Affects the number of bits in address bus
G5	C_S_AXI_DATA_WIDTH	P6, P7, P16		Affects the number of bits in data bus
G8	C_NUM_SS_BITS	P24, P25	-	Defines the total number of slave select bits
I/O Signals				
P3	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.
P6	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.
P7	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.
P13	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.
P16	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]	-	G8	The number of SS_I pins are generated based on C_NUM_SS_BITS.
P25	SS_O[(C_NUM_SS_BITS - 1):0]	-	G8	The number of SS_O pins are generated based on C_NUM_SS_BITS.

Register Overview Table

Table 4 gives a summary of the AXI Quad SPI core registers.

Table 4: **Core Registers**

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	SPIDTR	Write	0x0	SPI Data Transmit Register A single register or a FIFO
C_BASEADDR + 6C	SPIDRR	Read	NA (1)	SPI Data Receive Register A single register or a FIFO
C_BASEADDR + 70	SPISSR	R/W	No slave is selected	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

Table 4: **Core Registers *(Cont'd)***

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
Interrupt Controller Grouping				
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

Note:

1. The power on reset data in the SPI DRR is unknown data. 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.

Register Details

Software Reset Register (SRR)

The Software Reset register allows you 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 16 AXI clock cycles. Any other write access generates an error condition with undefined results and results in error generation. The bit assignment in the software reset register is shown in Figure 4 and is described in Table 5. Any attempt to read this register returns undefined data.

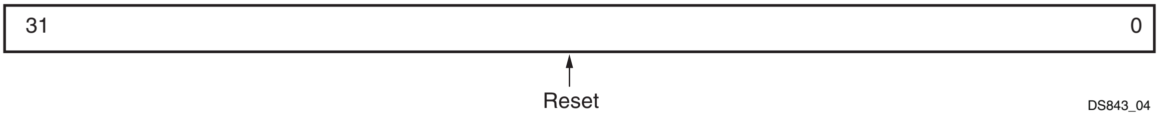


Figure 4: **Software Reset Register (C_BASEADDR + 0x40)**

Table 5: **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 core.

SPI Control Register (SPICR)

The SPI Control register allows you to control various aspects of the AXI Quad SPI core. The bit assignment in the SPICR is shown in Figure 5 and described in Table 6.

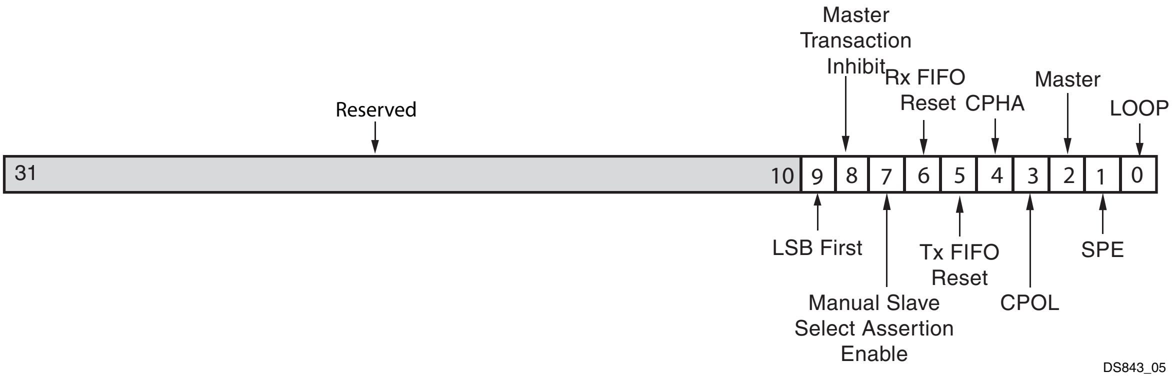


Figure 5: **SPI Control Register (C_BASEADDR + 0x60)**

Table 6: **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'. 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. 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 when the device is configured as a master and the device is enabled (SPE asserted). This bit has no effect on slave operation. 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. 0 = Receive FIFO normal operation 1 = Reset receive FIFO pointer
5	Tx FIFO Reset	R/W	0	**Transmit FIFO Reset**: When written to 1, this bit forces a reset of the transmit FIFO to the empty condition. One AXI clock cycle after reset, this bit is again set to 0. 0 = Transmit FIFO normal operation 1 = Reset transmit FIFO pointer

Table 6: **SPI Control Register (SPICR) Description (C_BASEADDR + 0x60) *(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
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. 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. 0 = Slave configuration 1 = Master configuration ***Note:*** In Dual/Quad SPI mode only Master mode is allowed.
1	SPE	R/W	0	**SPI System Enable:** Setting this bit to 1 enables the SPI devices. 0 = SPI system disabled. Both master and slave outputs are in 3-state and slave inputs are ignored. 1 = SPI system enabled. The master outputs are active (for example, IO0 (MOSI) and SCK in idle state) and slave outputs become active if $\overline{SS}$ is asserted. The master starts a transfer 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. 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. The loopback mode is supported only when C_SPI_MODE = 0.

Notes:

1. Setting this bit ('LSB first') is allowed only in Standard SPI mode. Dual/Quad SPI modes support 'MSB first' mode only. In Dual/Quad SPI mode, if this bit is set then the corresponding error bit is set in the SPISR and an interrupt is generated.
2. In Dual and Quad SPI mode, the allowed values for CPHA and CPOL are either 00 or "11". Setting other configurations cause an error when 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 a 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, the master mode only is supported. 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.
4. Loopback is allowed in Standard SPI mode by default. Loopback in not supported in the Dual or Quad SPI modes.

SPI Status Register (SPISR)

The SPI Status register is a read-only register that provides the status of the AXI Quad SPI core. The bit assignment in the SPISR is shown in Figure 6 and described in Table 7. Writing to the SPISR does not modify the register contents.

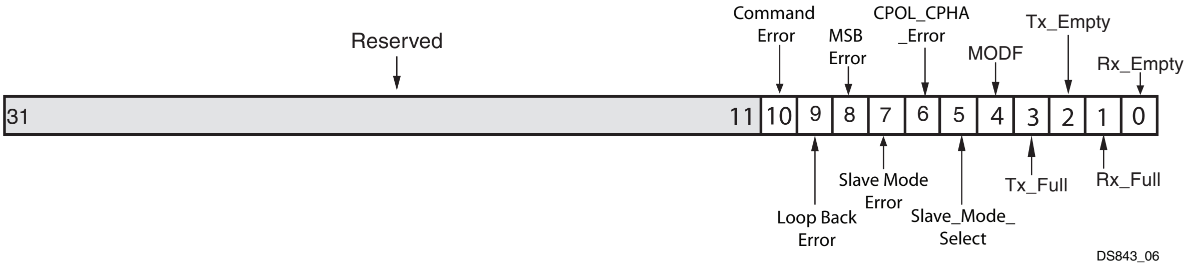


Figure 6: **SPI Status Register (C_BASEADDR + 0x64)**

Table 7: **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** 0 = Default. 1 = When the core is configured in Dual/Quad SPI mode and the first entry in the SPI DTR FIFO (after reset) does not match the supported command list for the particular memory, this bit is set. Command Error is only applicable when the core is configured in either Dual or Quad mode.
9	Loop Back Error	Read	0	**Loopback Error Flag** 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 the control register (SPICR). Loopback is only allowed when the core is configured in C_SPI_MODE = 0 mode. Other values of the bit cause an error and the interrupt bit is set.
8	MSB Error	Read	0	**MSB Error Flag** 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). 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 in either Dual or Quad mode.
7	Slave Mode Error	Read	1	**Slave Mode Error Flag** 1 = This bit is set when the core is configured with Dual or Quad SPI mode and Master is set to 0 in the control register (SPICR). 0 = Master mode is set in the control register (SPICR). In Dual/Quad SPI mode, only the Master mode is allowed. Slave Mode Error Flag is only applicable when the core is configured in either Dual or Quad Mode.

Table 7: **SPI Status Register (SPISR) Description (C_BASEADDR + 0x64)** ***(Cont'd)***

Bit(s)	Name	Core Access	Reset Value	Description
6	CPOL_CPHA_Error	Read	0	**CPOL_CPHA_Error Flag** 0 = Default 1 = The CPOL and CPHA are set to 01 or 10 When the SPI memory is chosen as Winbond or Numonyx, SPI_MODE is set to either 1 or 2 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 in either Dual or Quad Mode.
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 when C_SPI_MODE = 0 0 = Asserted when core configured in slave mode and selected by external SPI master
4	MODF	Read	0	**Mode-Fault Error Flag** This flag is set if the $\overline{SS}$ signal goes active while the SPI device is configured as a master. MODF is automatically cleared by reading the SPISR. A low-to-high MODF transition generates a single-cycle strobe interrupt. 0 = No error 1 = Error condition detected
3	Tx_Full	Read	0	**Transmit Full** When a transmit FIFO exists, this bit is set high when the transmit FIFO is full. 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 C_SPI_MODE = 0: Standard SPI Mode). This bit is cleared when the SPI transfer has 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. When FIFOs do not exist, this bit is set on the completion of an SPI transfer (this option is available only in C_SPI_MODE = 0: Standard SPI mode). Either with or without FIFOs, this bit is cleared on an AXI write to the FIFO or transmit register. For Dual/Quad SPI mode, the FIFO is always present in the core.
1	Rx_Full	Read	0	**Receive Full** When a receive FIFO exists, this bit is set high when the receive FIFO is full. The occupancy of the FIFO is incremented on the completion of each SPI transaction. When FIFOs do not exist, this bit is set high when an SPI transfer has completed (this option is available only in C_SPI_MODE = 0: 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. When FIFOs do not exist, this bit is set high when the receive register has been read (this option is available only in C_SPI_MODE = 0: 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 to with the data to be transmitted on the SPI bus. When 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.

For Dual/Quad mode

The first write must always be a SPI command from AXI transactions, followed by the address (either 24 bit or 32 bit), then filled with the data to be transmitted. When reading the memory status register, (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 the 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 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, while Table 8 shows the specifics of the data format.

When a transmit FIFO exists, data is written directly into the FIFO and the first location in the FIFO is treated as the SPI DTR. The pointer is decremented after completion of each SPI transfer. The choice of inclusion (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 (C_SPI_MODE is set to 0). If the core is configured in Dual or Quad SPI mode, the FIFO always exists. In these modes, 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 to the SPI DTR is not allowed and the read transaction results in undefined data.

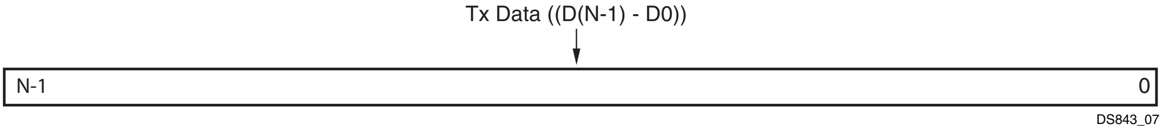


Figure 7: **SPI Data Transmit Register (C_BASEADDR + 0x68)**

Table 8: **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, 16 or 32, based upon 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 occur with a master SPI device as well.

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 (C_SPI_MODE is set to 0). If the core is configured in Dual or Quad SPI mode, the FIFO always exists. In these modes, the FIFO depth is defined by the parameter C_FIFO_DEPTH (allowed values are 16 or 256). For both master and slave SPI mode (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 a successful OK response.

Note: The power on reset values for the SPI DRR are unknown. When known data has been written into the receiver FIFO during core transactions, the data in this register can then be read.

The SPI DRR is shown in Figure 8, while the specifics of the data format is described Table 9.

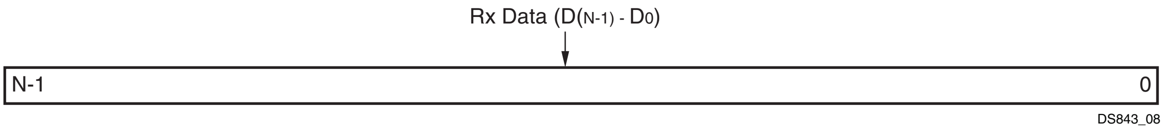


Figure 8: **SPI Data Receive Register (C_BASEADDR + 0x6C)**

Table 9: **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, 16 or 32, based upon the core configuration. In Dual or Quad SPI mode, this register is 8 bits 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 $\overline{SS}$ vector 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 10.

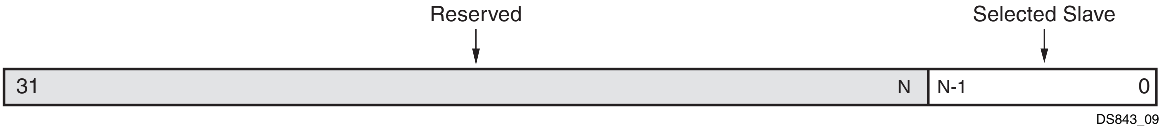


Figure 9: **SPI Slave Select Register (C_BASEADDR + 0x70)**

Table 10: **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 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 (eight-bit for 256 deep memory), 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 and the specifics of the data format are described in Table 11.

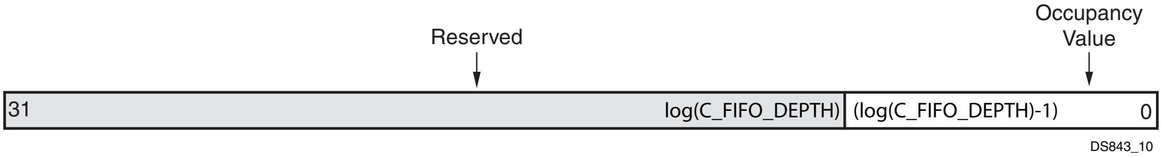


Figure 10: **SPI Transmit FIFO Occupancy Register (C_BASEADDR + 0x74)**

Table 11: **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 bits 3:0 of this register are valid because a FIFO depth of 16 only is allowed.
2. In Dual or Quad SPI mode - the bit position is changed based upon the FIFO depth. A 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 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 (eight-bit for 256 deep memory), 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 receiver 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 12.

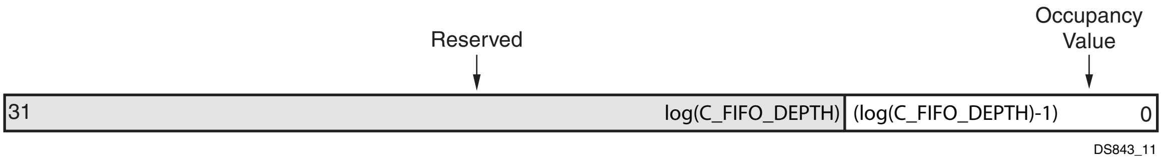


Figure 11: **SPI Receive FIFO Occupancy Register (C_BASEADDR + 0x78)**

Table 12: **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 Dual or Quad SPI mode, the bit position is changed based on the FIFO depth. A maximum of 256 deep, byte wide FIFO is allowed in the design.

Interrupt Register Set Description

The AXI Quad SPI core has a number of distinct interrupts that are sent to the interrupt controller. The AXI Quad SPI 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/Quad SPI mode, when the parameter C_FIFO_DEPTH is set to 16 or 256 and when core is configured in master mode, most of the interrupts shown in Table 14 are available. In Standard SPI mode, when the parameter, C_FIFO_DEPTH is set to 0, all of the interrupts except bit(6), Tx 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, shown in Figure 12 and described in Table 13. This bit is a read/write bit and is cleared upon reset.

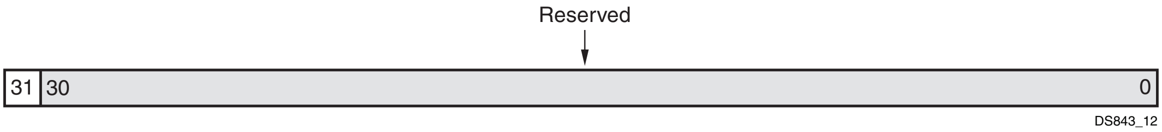


Figure 12: **Device Global Interrupt Enable Register (DGIER) (C_BASEADDR + 0x1C)**

Table 13: **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. 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 upon whether the system is configured with FIFOs or not, and 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 14. 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.

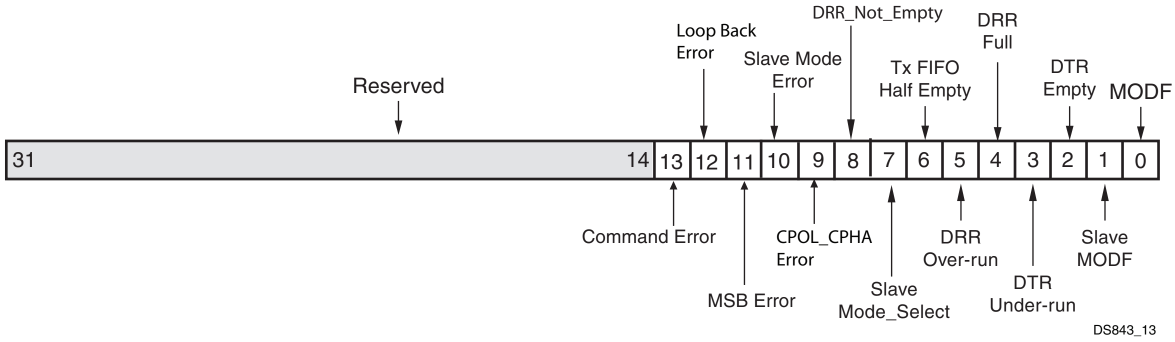


Figure 13: **IP Interrupt Status Register (IPISR) (C_BASEADDR + 0x20)**

Table 14: **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. 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 a particular memory When the SPI command in DTR FIFO does not match with the internally supported command list, the core completes the SPI transactions in Standard SPI format. This bit is set to communicate this behavior. In Standard SPI (C_SPI_MODE = 0) mode, this bit is always in its default state.

Table 14: **IP Interrupt Status Register (IPISR) Description (C_BASEADDR + 0x20)** ***(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
12	Loop Back Error	R/TOW [(1)]	0	**Loopback Error** IPISR bit(12) is the Loopback Error. 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 (C_SPI_MODE = 0) mode, this bit is always in its default state.
11	MSB Error	R/TOW [(1)]	0	**MSB Error** IPISR bit(11) is the MSB Error. 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 (C_SPI_MODE = 0) mode, this bit is always in default 0 state.
10	Slave Mode Error	R/TOW [(1)]	1	**I/O Mode Instruction Error** IPISR bit(10) is the Slave Mode Error. This flag is asserted when: • the core is configured in either Dual or Quad SPI mode and • the core is configured in Master = 0 in control register (SPICR(2)). In Standard SPI (C_SPI_MODE = 0) mode, this bit is always in default 0 state.
9	CPOL_CPHA Error	R/TOW [(1)]	0	**CPOL_CPHA Error** IPISR bit(9) is the CPOL_CPHA Error. This flag is asserted when: • the core is configured in either Dual or Quad SPI mode and • the CPOL - CPHA control register Bits are set to 01 or 10. In Standard SPI (C_SPI_MODE = 0) mode, this bit is always in default 0 state.
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 when C_FIFO_DEPTH = 16/256 and the core is configured in the slave mode of the 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 the first data beat. The assertion of this bit is applicable only when C_FIFO_DEPTH = 16/256 and the core is configured in slave mode in Standard SPI mode (C_SPI_MODE = 0). When C_FIFO_DEPTH = 0 this bit always returns 0. Therefore, it is recommended to use this bit only when C_FIFO_DEPTH = 16/256 and the core is configured in slave mode. This bit has no significance when C_SPI_MODE is 1 or 2.
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 mode (C_SPI_MODE = 0). 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. This bit is applicable only when the core is configured in the slave mode of the Standard SPI mode.

Table 14: **IP Interrupt Status Register (IPISR) Description (C_BASEADDR + 0x20) *(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
6	Tx FIFO Half Empty	R/TOW [(1)]	0	**Transmit FIFO Half Empty** In Standard SPI mode, 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. The same logic applies when the FIFO depth is 256 where 10000000 changes to 01111111. In Dual or Quad SPI mode, based upon the FIFO depth, this bit is set at a half empty condition. This interrupt exists only if the AXI Quad SPI core is configured with FIFOs (In Standard, Dual or Quad SPI mode).
5	DRR Overrun	R/TOW [(1)]	0	**Data Receive Register/FIFO Overrun** IPISR bit(5) is the data receive FIFO overrun interrupt. This bit is set by a one-clock period strobe to the interrupt register when an attempt to write data to a full receive register or FIFO is made by the SPI core logic to complete an SPI transfer. This can occur when the SPI device is in either master or slave mode (in Standard SPI mode (C_SPI_MODE = 0)) 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 occurs only when the SPI device is configured as a slave in Standard SPI mode 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 section 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.
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 when the $\overline{\text{SS}}$ signal goes active while the SPI device is configured as a slave, but is not enabled. This bit is set immediately upon $\overline{\text{SS}}$ going active and is continually set if $\overline{\text{SS}}$ is active and the device is not enabled.

Table 14: **IP Interrupt Status Register (IPISR) Description (C_BASEADDR + 0x20)** ***(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
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 enable register, IPIER, 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, shown in Figure 14 and described in Table 15. All bits are cleared upon reset.

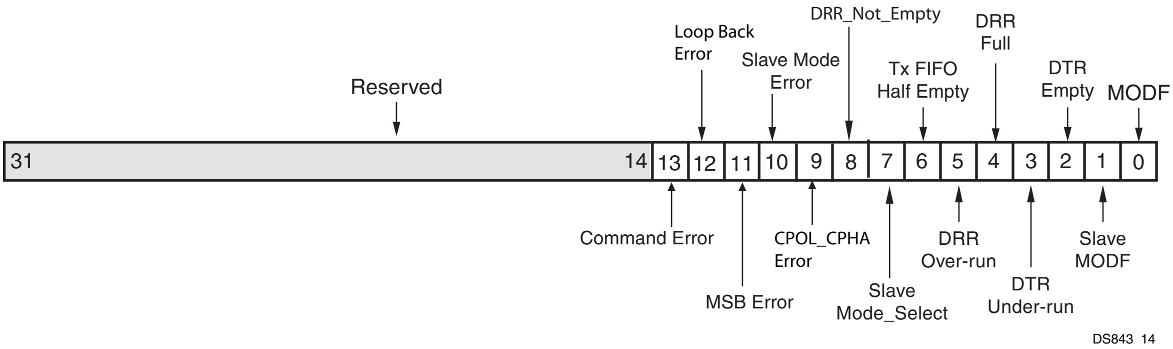


Figure 14: **IP Interrupt Enable Register (IPIER) (C_BASEADDR + 0x28)**

Table 15: **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 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 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 in Dual or Quad SPI mode.
10	Slave Mode Error	R/W	0	**I/O Mode Instruction Error** 0 = Disabled 1 = Enabled This bit is applicable only when the core is in Dual or Quad SPI mode.

Table 15: **IP Interrupt Enable Register (IPIER) Description (C_BASEADDR + 0x28)** ***(Cont'd)***

Bit(s)	Name	Access	Reset Value	Description
9	CPOL_CPHA Error	R/W	0	**CPOL_CPHA Error** 0 = Disabled 1 = Enabled This bit is applicable only when the core is in Dual or Quad SPI mode.
8	DRR_Not_Empty	R/W	0	**DRR_Not_Empty** 0 = Disabled 1 = Enabled This bit is applicable only when C_FIFO_DEPTH = 1 and the core is configured in slave mode of the Standard SPI mode. If C_FIFO_DEPTH = 0, the setting of this bit has no effect. This is allowed only in Standard SPI mode. It means this bit is not set in IPIER. Therefore, it is recommended to use this bit only in C_FIFO_DEPTH /= 0 condition when the core is configured in slave mode. This bit has no significance when C_SPI_MODE is 1 or 2.
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 This bit is applicable 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

Design Description

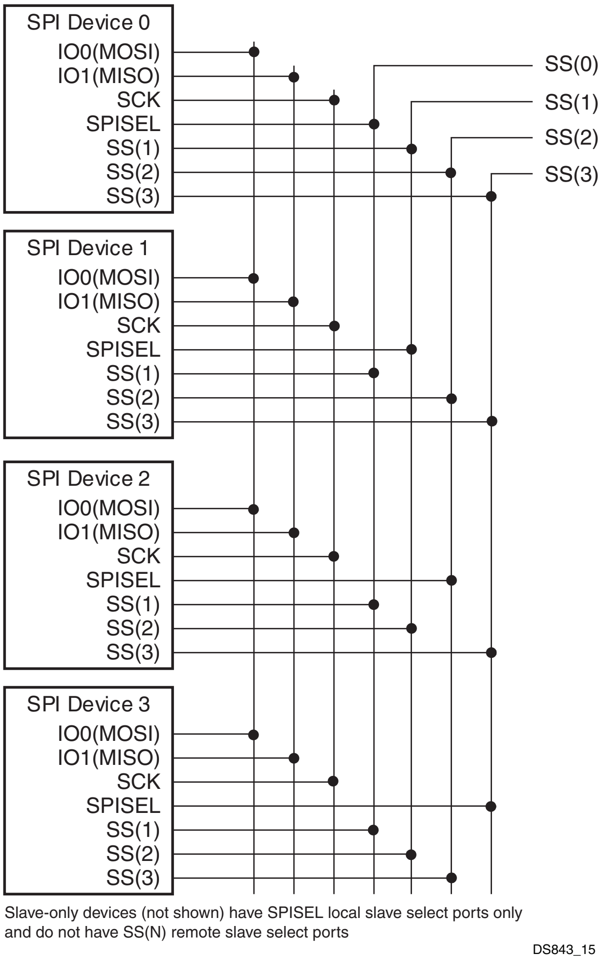
Standard SPI Device Features

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

- 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.
- Supports Multi-master environment (implemented with 3-state drivers and requires software arbitration for possible conflict). See AXI Quad SPI Core in Standard SPI Multi-Master Configuration section.
- Supports Multi-slave environment (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.

AXI Quad SPI Core 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 15 for more reference.



Slave-only devices (not shown) have SPISEL local slave select ports only and do not have SS(N) remote slave select ports

Figure 15: **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 15. The same configuration diagram is applicable for Dual mode.

Standard SPI Mode - Optional FIFOs

Optional FIFOs, either 16 or 256 deep, can be included in the design if the core is configured in Standard SPI mode. Because SPI is full-duplex (both transmit and receive FIFOs are instantiated as a pair), there is the option to include FIFOs in the AXI Quad SPI core, 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 by 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 Master Transaction Inhibit bit, bit(23) of 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 results in lost data.

When the AXI Quad SPI core is configured with FIFOs, SPI transfers can be started in two different ways depending on when the enable bit in the SPICR is set. If the enable bit is set prior to the first data being loaded in the FIFO, the SPI transfer begins immediately after the write to the master transmit FIFO. If the FIFO is emptied by 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.

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 option to choose the depth of FIFO. The depth of this FIFO can be 16 or 256. The width of the FIFO in this mode is always 8 bits wide.

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.

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 $\overline{SS}$ 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 a SPI device configured as a slave, but is not enabled, is selected (that is, its $\overline{SS}$ 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 provided as well. 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 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 (Dual and Quad modes are tested only with Winbond memories). See the data sheet of the targeted SPI slave flash memory or EEPROMs for maximum speed of operation. Ensure that you use the correct values when selecting the AXI clock and the C_SCK_RATIO parameter. The AXI clock and the C_SCK_RATIO set up the clock at the SCK pin of AXI Quad SPI core. When 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.

SPI Slave Mode - Standard SPI Configuration (C_SPI_MODE = 0)

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 SPI Control Register (SPICR) to '0'. The slave mode is allowed only in the Standard SPI mode. In Dual or Quad SPI mode, the core supports only the master mode of operation.

All the incoming signals are synchronized to the AXI clock when C_SCK_RATIO is greater than 4. Because of the tight timing requirements when C_SCK_RATIO equals 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 equals 4. For other C_SCK_RATIO values, it is preferred, but might not be necessary to have such synchronization.

During the slave mode operation it is strongly recommended to use the FIFO by setting C_FIFO_DEPTH = 16 or 256 in Standard SPI mode. In the 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 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

The system 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 (the clock's idle state = low) or active low clock (the clock's idle state = 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 = 0, data is valid on the first SCK edge (rising or falling) after $\overline{SS}$(N) has been asserted. If CPHA = 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 upon a write to the transmit register. Consequently, it is important 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.

In Standard SPI mode, any of the 00, 01, 10, 11 values are allowed for the CPHA and CPOL bits. In Dual or Quad SPI modes, only 00 or 11 values are allowed for the CPHA and CPOL bits,. If any other modes are set in the SPICR, an interrupt is set indicating an error and the core does not perform as expected.

CPHA Equals Zero Transfer Format

Figure 16 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.

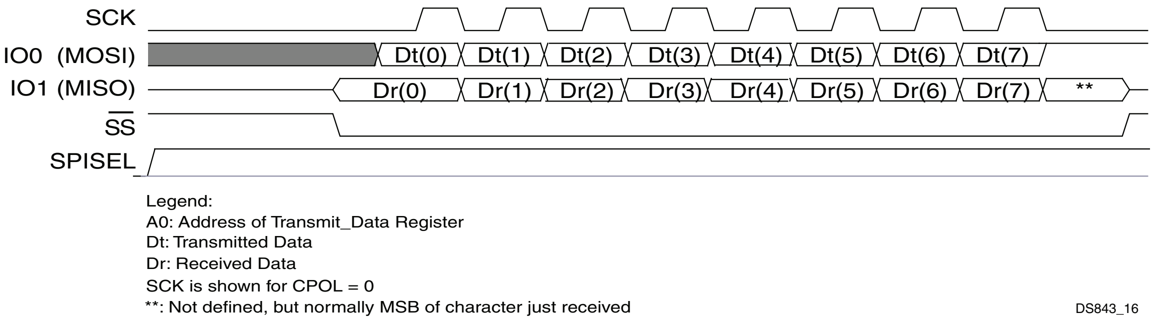


Figure 16: **Data Write-Read Cycle on SPI Bus with CPHA = 0 and SPICR(24) = 0 for 8-bit data**

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 de-asserted 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 the 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, and the required $\overline{\text{SS}}$(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 and 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), shown in Figure 17. 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.

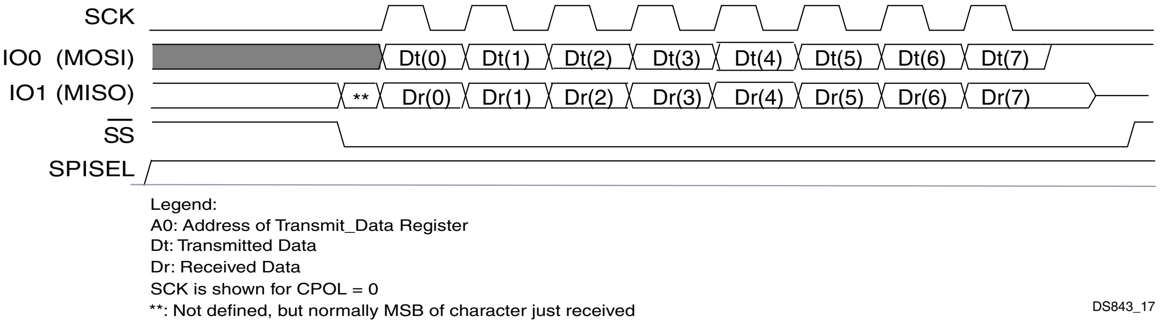


Figure 17: **Data Write-Read Cycle on SPI Bus with CPHA = 1 and SPICR(24) = 0 for 8-bit Data**

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 the slave select ($\overline{SS}$(N)) is asserted automatically by the SPI master device (SPICR bit(7) = 0). This mode is not supported in Dual or Quad SPI mode. This configuration mode is 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 the SPISSR register again appears on the $\overline{SS}$(N) output at the beginning of a new transfer. The slave select signal does not need to be manually controlled.

SPI Protocol with Manual Slave Select Assertion

This section describes the SPI protocol where the slave select ($\overline{SS}$(N)) is manually asserted (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. SCK must be stable before the assertion of slave select. When manual slave select mode is used, the SPI master must be enabled first (SPICR bit(7) = 1) to set SCK to 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 utilized 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 the assertion of the 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 to use the Manual Slave Select mode.

Beginning and Ending SPI Transfers

The details of the begin and end periods depend on the CPHA format selected and whether the SPI is configured as a master or a slave. This section describes the begin and end period for SPI transfers.

Transfer Begin Period

The definition of the transfer begin period for the AXI Quad SPI core is consistent with the M68HC11 reference manual, which 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 $\overline{SS}$ 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 as 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 also asserted. 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 the 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 16 and Figure 17. 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 the SPISR and IPISR bits for both master and slave SPI devices. Note that external SPI slave devices might use an internal AXI clock that is asynchronous to the SCK clock. This can cause status bits in the SPISR and IPISR to be inconsistent with each other. Therefore, the AXI Quad SPI core cannot be used in a system with external SPI slave devices that do not use the AXI clock.

When the AXI Quad SPI core is configured with FIFOs and a series of consecutive SPI 8-bit/16-bit/32-bit element transfers are performed (based upon 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.

Designing with the C_USE_STARTUP Parameter

The C_USE_STARTUP parameter is applicable only for Virtex®-6 and 7 series FPGA devices and if the core is intended to be used 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 upon 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 Xilinx Answer Records or the device user guides ([Ref 11], [Ref 12] and [Ref 13]) to understand more about the functionality and use of the STARTUP primitives for the respective FPGA devices.

C_USE_STARTUP = 1

In 7 series FPGA designs:

- 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.

In Virtex-6 FPGA designs:

- The SCK and IO1_I port from the core are 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 = 0

In Virtex-6 and 7 Series FPGA designs:

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

Core Behavior

The AXI Quad SPI core supports Winbond and Numonyx memories. Check the commands if other memories need to be tested with the core. If the commands, address and data behavior is the same for a different memory, then this memory can be chosen as the base memory to test the core.

The core is designed to understand the commands and expected behavior for the targeted memory. The commands that are not supported by the Winbond or Numonyx memory part, are marked with a Command Error. When the Command Error is set, the core does not execute the SPI transaction for that command and a command error interrupt is generated. If an address phase is included, after the command phase the next DTR contents are transferred on 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 the SPI DTR along with the number of bytes intended to read from memory. For more information on the number of Dummy Bytes needed for a particular instruction, see the data sheet for the targeted memory, [Ref 6] and [Ref 7].

For read commands, after the transmission of the address bits, the core immediately reverts to input mode and starts storing the data in DRR. Therefore, you must be aware of how many dummy bytes are to be ignored in the DRR. For example, in Fast Read Dual Output command in Winbond memory, the DTR should be filled with 1 command byte + 3 address bytes + 2 dummy bytes for dummy cycles + 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 IO0_I and IO_1 lines and stored in the SPI DRR, which includes the two dummy cycles + valid data. Therefore, while reading the SPI DRR, ignore the first 6 bytes in the SPI DRR; the 7th byte onwards is the real data available in FIFO. This also applies to other dual or quad read commands.

For each fresh transaction, 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.

SPI Command, Address and Data Flow Description

For proper 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 18.

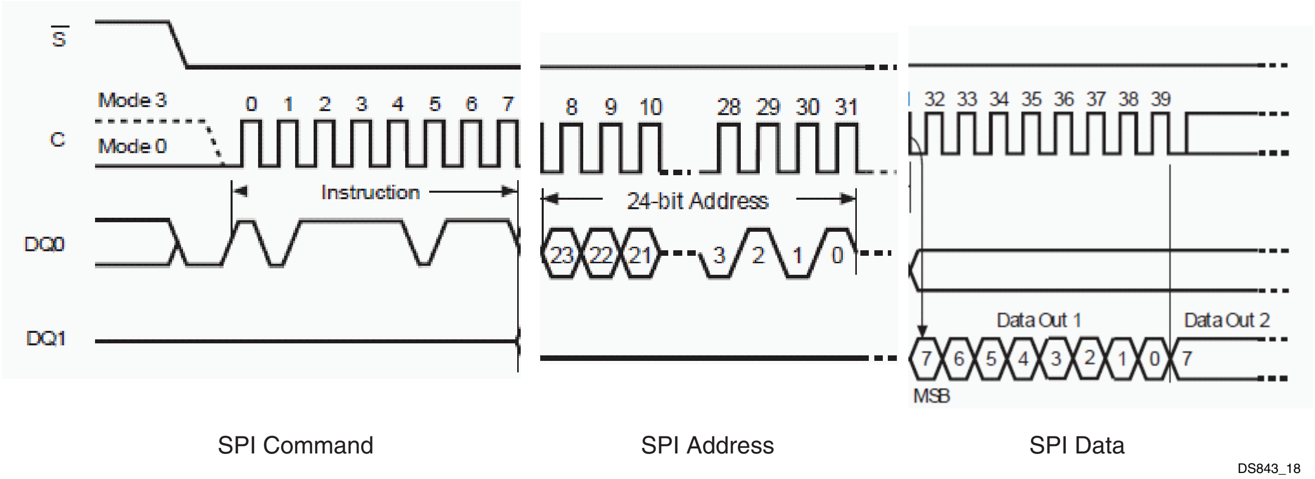


Figure 18: **SPI Transactions - CMD, ADDR and Data Phase**

SPI Command

If the first entry in the 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 IO0 (MOSI) line along with the SPI clock.

- When the C_SPI_MEMORY = 2 (Numonyx memories) then, based upon the supported command and C_SPI_MODE, the IO0, IO1, IO2, IO3 lines are used. When the first entry in the 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 = 1 (Winbond memories) then, based upon the supported command, the IO0 line is used.
- When the C_SPI_MEMORY = 0 (mixed mode memories; Winbond and Numonyx), then the common command set between these two memories is supported. In this case, based upon C_SPI_MODE setting (1 or 2) the behavior for 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 C_SPI_MODE = 1 is selected, the Dual and Standard SPI commands are supported. If C_SPI_MODE = 2, the Quad, Dual and Standard SPI commands are supported.

SPI Address

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

- When the C_SPI_MEMORY = 2 (Numonyx memories) and C_SPI_MODE is 1 or 2), then based upon the supported command, the IO0, IO1 (IO2, IO3 in Quad mode) lines are used.
- When the C_SPI_MEMORY = 1 (Winbond memories) and C_SPI_MODE is 1 or 2, based upon 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: Winbond and Numonyx), then the common command set between these two memories is supported. In this case, based upon 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 the memories. 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 = 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, in the case of 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 turns 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. Ensure that you 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

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

- When the C_SPI_MEMORY = 2 (Numonyx memories) and the C_SPI_MODE is 1 or 2, then based upon the supported command, the IO0, IO1 (IO2, IO3 in Quad mode) lines are used.
- When the C_SPI_MEMORY = 1 (Winbond memories) and C_SPI_MODE is 1 or 2, based upon the supported command, the IO0, IO1 (IO2, IO3 in Quad mode) lines are used to transfer the data bits.
- When the C_SPI_MEMORY = 0 (mixed mode memories: Winbond and Numonyx), then the common command set between these two memories is supported. In this case, based upon C_SPI_MODE setting (1 or 2), the behavior for Winbond memory for the data flow is taken as reference. This is applicable for both the 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 = 2, the Quad, Dual and Standard SPI commands are supported.

AXI Quad SPI Core Behavior with Different Configurations

When the Numonyx SPI memory is targeted as a slave, different types of instruction sets are supported. At present, the Numonyx memory part N25Q_256_3 data sheet supports only Extended SPI Instructions. You must read the data sheet to match the expected behavior of the core in different modes set through C_SPI_MODE (1 or 2), when C_SPI_MEMORY is set to 2.

C_SPI_MODE = 1

The core is configured to support the Dual mode and Standard mode SPI commands. The configurations, modes and their behavior for C_SPI_MEMORY set to 0, 1 and 2 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. Because the core supports only the Winbond and Numonyx memories at present, the slave is one of these two memory types.

This mode is considered to be a mixed mode memory mode, where Winbond memories are taken as the base for defining behavior of the core. The instructions which are common to Winbond and Numonyx memories (from the Extended SPI protocol instructions set) are supported. Table 16 shows the generalized core behavior.

Table 16: **Core Behavior for C_SPI_MODE = 1 and C_SPI_MEMORY = 0**

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. In C_SPI_MODE = 1, the Dual and 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 17 shows the generalized core behavior.

Table 17: **Core Behavior for C_SPI_MODE = 1 and C_SPI_MEMORY = 1 (Winbond Memory)**

Command Type	Winbond Memory	Command Error	Core Behavior
Standard SPI	Supported	No	Standard Format
Standard SPI	Not Supported	Yes	No SPI Transaction
Dual Mode	Supported	No	Dual Mode Instruction Format as given in the data sheet
Dual Mode	Not Supported	Yes	No SPI Transaction
Quad Mode	Supported	Yes	No SPI Transaction
Quad Mode	Not Supported	Yes	No SPI Transaction

Notes:

1. The 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 also for information on how the command, address and data bits operate.

C_SPI_MEMORY = 2

This mode is a 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 18 shows the generalized core behavior.

Table 18: **Core Behavior for C_SPI_MODE = 1 and C_SPI_MEMORY = 2 (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 information on how the command, address and data bits operate.

C_SPI_MODE = 2

The core is configured to support the Quad mode, Dual mode and Standard mode SPI commands based upon the type of memory used as the SPI slave. The configurations, modes and their behavior for C_SPI_MEMORY set to 0, 1 and 2 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 memory types.

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. Most of the instructions which are common to Winbond and Numonyx memories (Extended SPI commands) are supported. Table 19 shows the generalized core behavior.

Table 19: **Core Behavior for C_SPI_MODE = 2 and C_SPI_MEMORY = 0**

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

Table 19: **Core Behavior for C_SPI_MODE = 2 and C_SPI_MEMORY = 0 *(Cont'd)***

Command Type	Winbond	Numonyx	Command Error	Core Behavior
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 and 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 a dedicated mode and supports only Winbond memories as SPI slave devices. Most of the SPI commands from the Winbond memories (W25Q64BV) are supported. Table 20 shows the generalized core behavior.

Table 20: **Core Behavior for C_SPI_MODE = 1 and C_SPI_MEMORY = 1 (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	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 also for information on how the command, address and data bits operate.

C_SPI_MEMORY = 2

This mode is a 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 21 shows the generalized core behavior.

Table 21: **Core Behavior for C_SPI_MODE = 1 and C_SPI_MEMORY = 2 (Numonyx Memory)**

Command Type	Numonyx Memory	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.

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. The following steps show how to successfully complete an SPI transaction:

1. Start from a known state, including SPI bus arbitration.
2. Configure the 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 the 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.
5. Ensure that the SPISSR register contains all ones.
6. Write configuration data to the master SPI device SPICR as required, including setting bit(7) for manual assertion of the $\overline{SS}$ vector and setting both the enable bit and the master transfer inhibit bit. This initializes SCK and IO0, but inhibits transfer.
7. Write to the SPISSR to manually assert $\overline{SS}$ 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 the 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 device, then clear the master transaction inhibit bit to continue N 8-bit element transfer. Note that an overrun of the SPI DRR register/FIFO can occur if the SPI DRR register/FIFOs are not read properly. Also note that SCK stretches the idle levels between element transfers (or groups of element transfers if utilizing FIFOs) and that IO0 can transition at the end of a element transfer (or group of transfers), but will be stable at least one-half SCK period prior to sampling edge of SCK.
11. Repeat step 9 and step 10 until all the data is transferred.
12. Write all ones to the SPISSR or exit the manual slave select assert mode to de-assert $\overline{SS}$ vector while SCK and IO0 are in the idle state.
13. Disable devices as required.

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

The following steps show how 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 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 the SPISR of both master and slave SPI devices as required.
12. Perform actions as required or dictated by the SPISR data.

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

The following steps show how to successfully complete an SPI transaction.

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 the 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 a 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 the SPISR of both master and slave SPI devices as required.
13. Perform actions as required or dictated by the 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).

The following steps show how 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 the IPISR of both master and slave SPI devices as required.
12. Perform interrupt requests as required.
13. Read the SPISR of both master and slave SPI devices as required.
14. Perform actions as required or dictated by the SPISR data.

Dual/Quad SPI Mode Transactions

In this mode, the core need be configured in Master mode only, or else the error interrupt is generated.

The sequence flow to operate the core in Dual mode is shown in the following steps. Take care in predicting the behavior of the core and the memory. Check the inter-dependency between the commands before filling the SPI DTR and enabling the SPI core to start the transaction.

1. Ensure that the C_SPI_MODE = 1 and C_SPI_MEMORY parameter is configured for proper SPI slave memory.
2. Ensure that the instructions that are operated on the required SPI clock, set by C_SCK_RATIO parameter, are listed.
3. Configure the core with C_FIFO_DEPTH parameter. This parameter can be either 16 or 256.
4. Write into the Soft Reset register to reset the core. This reset is active for 16 AXI cycles, during which each FIFO's register is in the reset state.
5. Write into the SPICR to make the core in master mode, set proper CPOL, CPHA values and make sure that the Master Transaction Inhibit bit is set.
6. Write the IPIER, IPISR register to enable the required interrupts.
7. Write into the SPI DTR with the Command, Address, Dummy and Data bytes to be transmitted in the same sequence which is provided in the data sheet of the target device.
8. Write into the SPI DTR with the number of data bytes intended to be read or written into memory along with command, address and dummy bytes.
9. Write into the SPISSR to assert the Chip Select signal from the core.
10. Write into the SPICR to enable the Master Transaction Inhibit bit, so that the core starts the SPI clock.
11. For a write, wait till the DTR empty interrupt is generated.
12. When the DTR empty signal is generated, you can still write the data into the SPI DTR for further transactions, if 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 has contents, then again the SPI clock is enabled and data is transmitted.
14. Steps 11, 12 and 13 apply while reading also. You must fill the DTR with any random data beats while reading the SPI slave memory.
15. When the selected slave is disabled by setting all the SPISSR bits to 1 (or setting the Master Transactions Inhibit bit in the SPICR to 1), 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 the DTR with the supported command.
16. In between new transactions, ensure that the SPI DTR and the SPI DRR are reset by writing to 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 Mode FIFO Fill Order

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

Check the supported and unsupported command list (see Unsupported Commands for Dual/Quad SPI Mode and Winbond/Numonyx Memory Mode for Winbond and Numonyx memories. If non-supported 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 behavior of the core are shown in Figures 19 and 20. In Figure 19, the core is configured in C_SPI_MODE = 1 and C_SPI_MEMORY = 1 mode with C_SCK_RATIO = 4. In Figure 20 the core is configured in C_SPI_MODE = 2 and C_SPI_MEMORY = 1 mode with C_SCK_RATIO = 4.

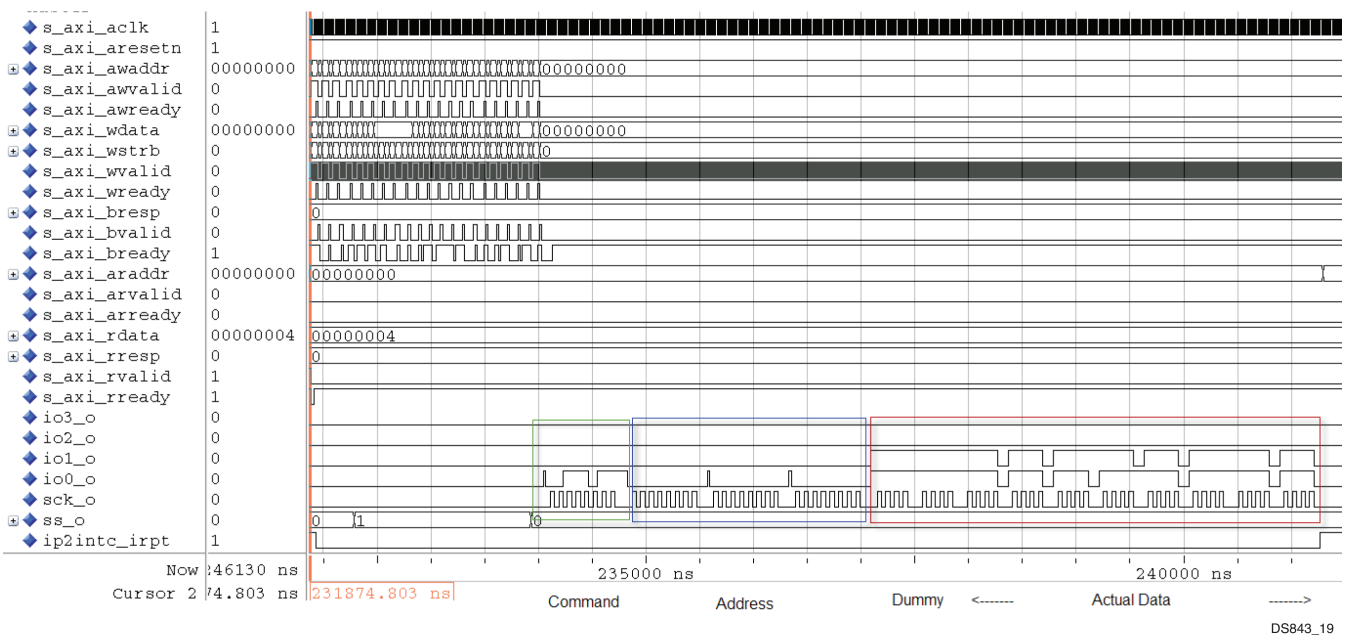
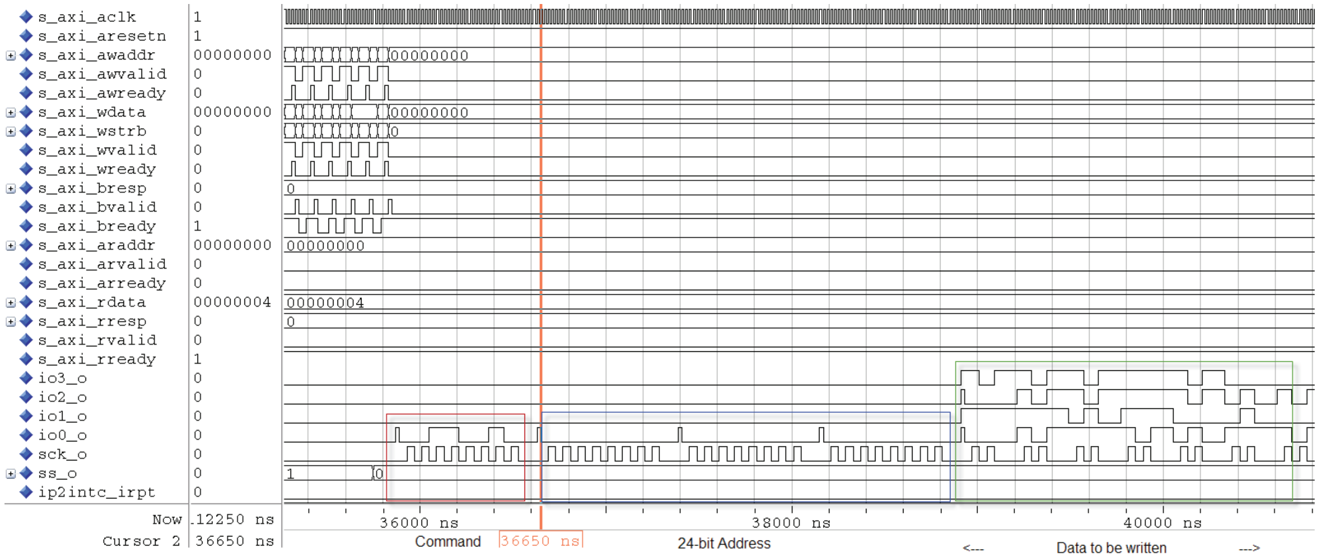


Figure 19: **Fast Read Dual Output command (3B h) Winbond memory**



DS843_20

Figure 20: **Quad Input Page Program command (32 h) Winbond memory**

Design Implementation

Target Technology

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

Device Utilization and Performance Benchmarks

Core Performance

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

The AXI Quad SPI core resource utilization for various parameter combinations measured with a Spartan-6 FPGA as the target device are detailed in Table 22. The utilization figures should be considered as reference only.

Table 22: **Performance and Resource Utilization Benchmarks on a Spartan-6 FPGA (xc6slx45tfgg484-3)**

Parameter Values (other parameters at default values)						Device Resources			Performance
C_SPI_MODE	C_FIFO_DEPTH	C_SPI_MEMORY	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	0	0	16	2	8	98	184	202	150
0	16	0	16	2	8	105	184	275	150
0	256	0	16	2	8	151	194	388	150
1	16	0	16	2	8	153	217	489	150
1	16	1	16	2	8	118	194	279	150
1	16	2	16	2	8	117	206	333	150
2	16	0	16	2	8	133	199	307	150
2	16	1	16	2	8	140	204	327	150
2	16	2	16	2	8	138	200	315	150
1	256	0	16	2	8	131	207	408	150
1	256	1	16	2	8	165	207	417	150
1	256	2	16	2	8	167	222	527	150
2	256	0	16	2	8	153	217	489	150
2	256	1	16	2	8	149	216	487	150
2	256	2	16	2	8	158	218	492	150

The core resource utilization for various parameter combinations measured with a Virtex-6 FPGA as the target device are detailed in Table 23. The utilization figures should be considered as reference only.

Table 23: **Performance and Resource Utilization Benchmarks on a Virtex-6 FPGA (xc6vlx130tff1156-1)**

Parameter Values (other parameters at default values)							Device Resources			Performance
C_SPI_MODE	C_FIFO_DEPTH	C_FIFO_MEMORY	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	C_USE_STARTUP	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	0	0	16	2	8	0	132	181	221	202
0	16	0	16	2	8	0	149	182	262	202
0	256	0	16	2	8	0	183	191	400	217
1	16	0	16	2	8	0	175	201	387	218
1	16	1	16	2	8	0	132	193	281	203
1	16	2	16	2	8	0	140	195	263	205
2	16	0	16	2	8	0	154	198	316	225
2	16	1	16	2	8	0	162	198	319	223
2	16	2	16	2	8	0	151	199	288	212
1	256	0	16	2	8	0	189	203	406	201
1	256	1	16	2	8	0	182	203	411	202
1	256	2	16	2	8	0	187	207	434	201
2	256	0	16	2	8	0	175	201	387	218
2	256	1	16	2	8	0	178	200	385	203
2	256	2	16	2	8	0	174	203	391	211
0	16	0	16	2	8	1	153	184	257	204
0	256	0	16	2	8	1	178	193	401	212
0	16	0	2	2	8	1	126	170	242	209
0	256	0	2	2	8	1	182	176	389	192

The core resource utilization for various parameter combinations measured with an Artix-7 FPGA as the target device are detailed in Table 24. The utilization figures should be considered as reference only.

Table 24: **Performance and Resource Utilization Benchmarks on an Artix-7 FPGA (XC7A100T-3)**

Parameter Values (other parameters at default values)							Device Resources			Performance
C_SPI_MODE	C_FIFO_DEPTH	C_FIFO_MEMORY	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	C_USE_STARTUP	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	0	0	16	2	8	0	116	180	229	167
0	16	0	16	2	8	0	162	182	282	186
0	256	0	16	2	8	0	146	197	398	169
1	16	0	16	2	8	0	161	203	409	170
1	16	1	16	2	8	0	135	194	262	179
1	16	2	16	2	8	0	134	196	290	172
2	16	0	16	2	8	0	126	197	273	169
2	16	1	16	2	8	0	142	197	277	190
2	16	2	16	2	8	0	172	200	319	171
1	256	0	16	2	8	0	186	209	423	162
1	256	1	16	2	8	0	182	209	427	170
1	256	2	16	2	8	0	187	207	431	174
2	256	0	16	2	8	0	161	203	409	170
2	256	1	16	2	8	0	174	202	407	168
2	256	2	16	2	8	0	176	204	409	191
0	16	0	16	2	8	1	111	183	284	217
0	256	0	16	2	8	1	168	198	396	200
0	16	0	2	2	8	1	124	169	259	196
0	256	0	2	2	8	1	164	183	383	178

The core resource utilization for various parameter combinations measured with a Virtex-7 FPGA as the target device are detailed in Table 25. The utilization figures should be considered as reference only.

Table 25: **Performance and Resource Utilization Benchmarks on a Virtex-7 FPGA (xc7v285tffg784-3)**

Parameter Values (other parameters at default values)							Device Resources			Performance
C_SPI_MODE	C_FIFO_DEPTH	C_FIFO_MEMORY	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	C_USE_STARTUP	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	0	0	16	2	8	0	112	181	225	250
0	16	0	16	2	8	0	106	182	251	222
0	256	0	16	2	8	0	169	192	399	244
1	16	0	16	2	8	0	181	204	417	229
1	16	1	16	2	8	0	114	194	264	261
1	16	2	16	2	8	0	145	196	290	274
2	16	0	16	2	8	0	120	197	276	225
2	16	1	16	2	8	0	144	197	280	263
2	16	2	16	2	8	0	168	200	331	275
1	256	0	16	2	8	0	181	204	417	229
1	256	1	16	2	8	0	180	203	411	230
1	256	2	16	2	8	0	168	200	331	275
2	256	0	16	2	8	0	158	208	426	250
2	256	1	16	2	8	0	187	206	432	224
2	256	2	16	2	8	0	192	201	431	251
0	16	0	16	2	8	1	109	183	254	242
0	256	0	16	2	8	1	158	208	426	250
0	16	0	2	2	8	1	140	169	260	241
0	256	0	2	2	8	1	143	177	391	239

The core resource utilization for various parameter combinations measured with a Kintex-7 FPGA as the target device are detailed in Table 26. The utilization figures should be considered as reference only.

Table 26: **Performance and Resource Utilization Benchmarks on a Kintex-7 FPGA (xc7k325tffg900-3)**

Parameter Values (other parameters at default values)							Device Resources			Performance
C_SPI_MODE	C_FIFO_DEPTH	C_FIFO_MEMORY	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	C_USE_STARTUP	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	0	0	16	2	8	0	125	181	225	241
0	16	0	16	2	8	0	146	182	254	258
0	256	0	16	2	8	0	182	192	394	237
1	16	0	16	2	8	0	176	204	413	228
1	16	1	16	2	8	0	93	194	265	258
1	16	2	16	2	8	0	158	196	290	252
2	16	0	16	2	8	0	147	197	275	250
2	16	1	16	2	8	0	151	197	281	261
2	16	2	16	2	8	0	187	207	435	260
1	256	0	16	2	8	0	192	208	427	228
1	256	1	16	2	8	0	187	203	411	237
1	256	2	16	2	8	0	186	205	413	253
2	256	0	16	2	8	0	192	208	427	228
2	256	1	16	2	8	0	100	183	254	273
2	256	2	16	2	8	0	188	177	388	222
0	16	0	16	2	8	1	100	183	254	273
0	256	0	16	2	8	1	164	193	395	234
0	16	0	2	2	8	1	139	169	260	234
0	256	0	2	2	8	1	188	177	388	221

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 21). 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 and timing will 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 27.

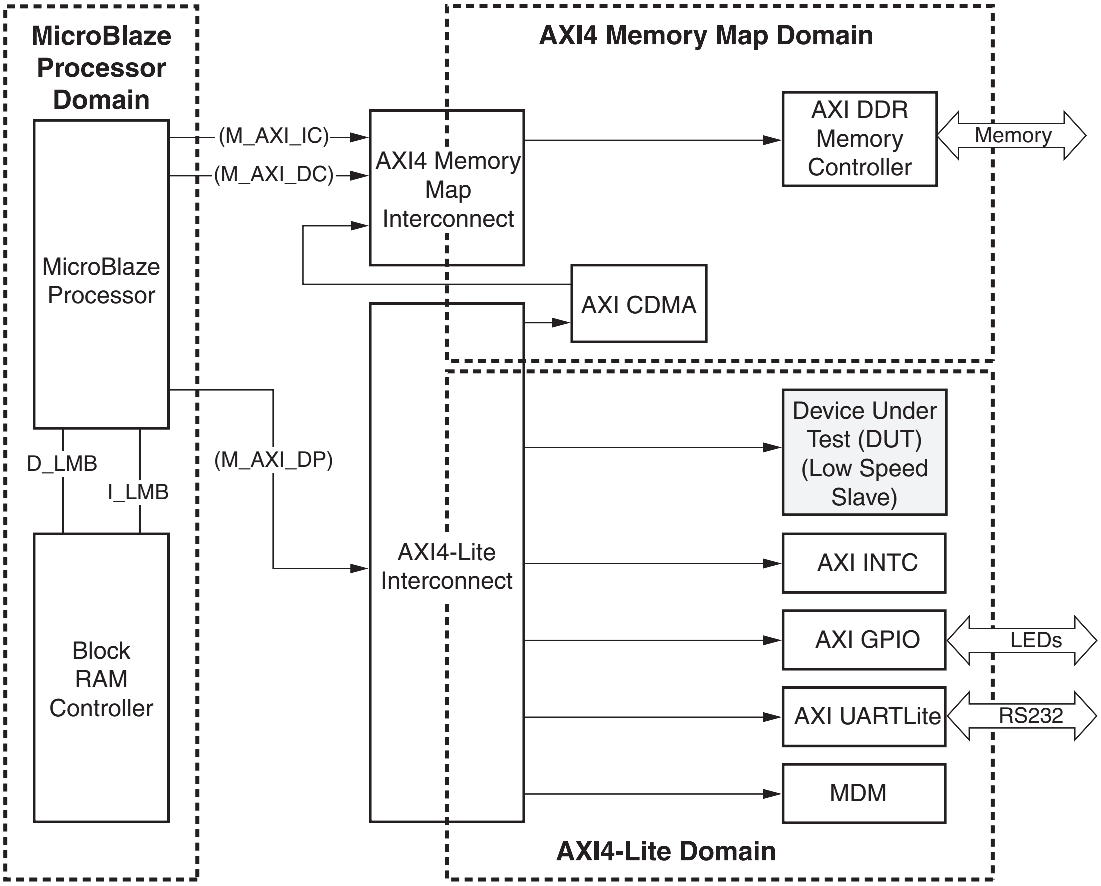


Figure 21: **Virtex-6 and Spartan-6 Devices F_{MAX} Margin System**

Table 27: **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

The following list contains 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 the SPI DRR registers have independent addresses. This is an exception to the M68HC11 specification which calls for two registers to have the same address.
3. All $\overline{SS}$ signals are required to be routed between SPI devices internally to the FPGA. This is because toggling of the $\overline{SS}$ signal is utilized in slaves to minimize FPGA resources.
4. Manual control of the $\overline{SS}$ signals is provided by setting bit(7) in the SPICR register. When the device is configured as a master and is enabled 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 $\overline{SS}$ vector.
5. A control bit is provided to inhibit master transfers. This bit is effective in any master mode, but its main use is for manual control of the $\overline{SS}$ signals.
6. In the M68HC11 implementation, the transmit register is transparent to the shift register which necessitates the write collision error (WCOL) detection hardware. This 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. The 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 modes 0 and 3. For some of these devices, the data valid time of 8 ns from the falling edge of SCK is applicable. While operating with these devices at a higher speed of 50 MHz (most of the instructions supports this speed), the core should be configured in C_SCK_RATIO = 2 mode (where the AXI clock 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 modes 1 and 3. 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 the IP core only. The AXI Quad SPI core, when configured in master mode, changes data on each falling edge and this behavior is as per the M68HC11 standard.
12. When the AXI Quad SPI core is configured in slave mode (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 the AXI clock. The core changes the data on the SCK falling edge + 1 AXI clock cycle.
13. When the AXI Quad SPI core is configured in slave mode and C_SPI_MODE = 0, 4 AXI clock cycles of idle time are required between each SPI transfer from the external SPI master. This idle time is required for the core to register the received data in the SPI DRR and load the new data to be transferred in the SPI DTR. It is important to note this idle time requirement when using the core in slave mode. In Dual/Quad mode (C_SPI_MODE=1/2), the slave mode configuration is not supported.

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 operations on only one slave at a time.
2. The core is based upon the specific memory parts from Winbond (W25Q80) and Numonyx(N25Q256). To test the core with different memory parts (not mentioned in the data sheet), it is important to understand the internal command decoding logic and the bit positions for correct operation of the core. Also, check the common command set between the specified 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. For a list of unsupported commands for both Winbond and Numonyx memories, see Unsupported Commands for Dual/Quad SPI Mode and Winbond/Numonyx Memory Mode.
4. It is recommended that the dedicated memory be used as AXI Quad SPI slaves because the core supports a limited set of commands. These commands 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 commands for the selected memory and gives better performance.

Common Supported Commands for Dual SPI Mode/Mixed Memory Mode)

For the setup when C_SPI_MODE =1 (Dual) and C_SPI_MEMORY=0 (Mixed Memory), the core supports commands that are common and identical (in terms of command, address and data behavior) in Winbond and Numonyx memories. See the data sheets for both memories for these commands. Some of the commands which are not supported in this mode include Fast Read, Dual IO Fast Read, Dual Output Fast Read. These commands are not supported by the core in mixed mode because the dummy bytes or dummy cycles are different in Winbond and Numonyx memories. Also, in mixed mode the volatile configuration register of Numonyx is not supported because this command is not present in the Winbond memory data sheet.

See the Unsupported Commands for Dual/Quad SPI Mode and Winbond/Numonyx Memory Mode for a list of unsupported commands.

Table 28: **Supported Commands for Dual SPI Mode/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
C7	Bulk Erase
D8	Sector Erase

Common Supported Commands for Quad SPI Mode/Mixed Memory Mode

For the setup when C_SPI_MODE=2 (Quad) and C_SPI_MEMORY=0 (Mixed Memory), the core supports the commands listed in Table 29, which are common and identical (in terms of command, address and data behavior) in Winbond and Numonyx memories. See the data sheet for both memories for such commands. The commands which are not supported in this mode include Fast Read, Dual Output Fast Read, Quad Output Fast Read, Dual IO Fast Read, Quad IO Fast Read. These commands are not supported by the core in mixed mode because the dummy bytes or dummy cycles are different in Winbond and Numonyx memories. Also in mixed mode, the volatile configuration register of Numonyx is not supported because this command is not present in the Winbond memory data sheet. See Unsupported Commands for Dual/Quad SPI Mode and Winbond/Numonyx Memory Mode for a list of unsupported commands.

Table 29: **Supported Commands for Quad SPI Mode/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	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

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

The unsupported commands for C_SPI_MODE = 1 or 2 and C_SPI_MEMORY = 1 or 2 (Winbond or Numonyx Memory) are listed in this section.

Winbond Memory

Unsupported commands/features for Winbond memory part, W25Q64VSFIG, are:

1. Fast Read Dual IO Continuous Read mode.
2. Fast Read Quad IO Continuous Read mode.
3. 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:

1. Release Power Down/High Performance mode (AB h): This command supports Release Power Down/High Performance mode OR reading the Device ID with a different combination of dummy bytes. The core supports

only "Release Power Down/High Performance mode" where only one command byte is required to be put in the DTR. The other mode of command is not supported, as there is another command (90h) to read the device ID. Note this exception.

Numonyx Memory

Unsupported commands/features for Numonyx memory part, N25Q256, are:

1. XIP mode or continuous read mode in both memories is not supported.
2. All the commands in C_SPI_MODE = 1 or 2 are supported in Extended SPI mode. DIO and QIO modes are not supported.
3. 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.

List of Acronyms

Acronym	Definition
AMBA	Advanced Microcontroller Bus Architecture
ARM®	Advanced RISC Machine
AXI	Advanced eXtensible Interface
BRG	Baud Rate Generator
CPHA	Clock Phase
CPOL	Clock Polarity
DGIER	Device Global Interrupt Enable Register
EDK	Embedded Development Kit
EEPROM	Electrically Erasable Programmable Read-Only Memory
FF	Flip-Flop
FIFO	First In First Out
FPGA	Field Programmable Gate Array
I/O	Input/Output
IP	Intellectual Property
IPIC	IP Interconnect
IPIER	IP interrupt enable register
IPIF	IP Interface
IPISR	IP Interrupt Status Register
ISE	Integrated Software Environment
LSB	Least Significant Bit
LUT	Lookup Table
IO1(MISO - in Standard SPI mode)	Master In Slave Out
IO0 (MOSI - in Standard SPI mode)	Master Out Slave In
MODF	Mode-fault Error
MSB	Most Significant Bit
RAM	Random Access Memory
Rx	Receive

Acronym	Definition
SCK	Serial Clock
SPI	Synchronous Serial Peripheral Interface
SPICR	SPI Control Register
SPI DRR	SPI Data Receive Register
SPI DTR	SPI Data Transmit Register
SPIE	SPI Interrupt Enable
SPISEL	SPI Slave Select Line
SPISR	SPI Status Register
SPISSR	SPI Slave Select Register
$\overline{\text{SS}}$(N)	Slave Select
Tx	Transmit
UCF	User Constraints File
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits)
WCOL	Write Collision error
XST	Xilinx Synthesis Technology

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. AXI4 AMBA® AXI Protocol Version: 2.0 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)

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 ISE® Design Suite Embedded Edition software under the terms of the Xilinx End User License. The core is generated using the Xilinx ISE Embedded Edition software (EDK).

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 modules and software, contact your local Xilinx sales representative.

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.

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.