LogiCORE IP AXI Serial Peripheral Interface (AXI SPI) (v1.02.a)

DS742 January 18, 2012 **Product Specification**

Introduction

The AXI Serial Peripheral Interface (SPI) connects to the Advanced eXtensible Interface (AXI4). This core provides a serial interface to SPI devices such as SPI Electrically Erasable Programmable Read-Only Memories (EEPROMs) and SPI serial flash devices. The SPI protocol, as described in the Motorola M68HC11 data sheet, provides a simple method for a master and a selected slave to exchange data. This 32-bit soft Intellectual Property (IP) core is designed to interface with the AXI4-Lite interface.

Features

- AXI4-Lite interface is based on the AXI4 specification
- Connects as a 32-bit AXI4-Lite slave
- Supports four signal interfaces:
  - Master Out Slave In (MOSI)
  - Master In Slave Out (MISO)
  - Serial Clock (SC)
  - $\overline{SS}$
- Slave select ($\overline{SS}$) bit for each slave on the SPI bus
- Full-duplex operation
- Master and slave SPI modes
- Programable clock phase and polarity
- Continuous transfer mode for automatic scanning of a peripheral
- Back-to-back transactions
- Automatic or manual slave select modes
- MSB/LSB first transactions
- Transfer length of 8-bits, 16-bits or 32-bits
- Local loopback capability for testing
- Multiple master and multiple slave environment
- Optional 16 element deep (an element is a byte, a half-word or a word) transmit and receive First In First Out (FIFO)

LogiCORE IP Facts				
Core Specifics				
Supported Device Family (1)	Artix-7(2), Virtex-7(2), Zynq™-7000, Kintex-7(2), Virtex-6(3), Spartan-6(4)			
Supported User Interfaces	AXI4-Lite			
Configuration	**Resources and Frequency**			
Configuration 1	LUTs	FFs	Freq.	Block RAMS
	See Table 20, Table 19, Table 18, Table 16 and Table 17.			0
Provided with Core				
Documentation	Product Specification			
Design Files	VHSIC Hardware Description Language (VHDL)			
Example Design	N/A			
Test Bench	N/A			
Constraints File	N/A			
Supported S/W Driver(5)	Standalone and Linux			
Tested Design Tools (6)				
Design Entry Tools	Xilinx Platform Studio (XPS)			
Simulation	Mentor Graphics ModelSim			
Synthesis Tools	Xilinx Synthesis Technology (XST)			
Support				
Provided by Xilinx @ www.xilinx.com/support				

Notes:

1. For a complete listing of supported derivative devices, see the IDS Embedded Edition Derivative Device Support.
2. For more information, see *7 Series FPGAs Overview* DS180.
3. For more information, see the Virtex-6 Family Overview Product Specification (DS150).
4. For more information, see Spartan-6 Family Overview Product Specification (DS160).
5. Standalone driver details can be found in the EDK or SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from http://wiki.xilinx.com.
6. For a listing of the supported tool versions, see the ISE Design Suite 13: Release Note Guide.

Functional Description

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

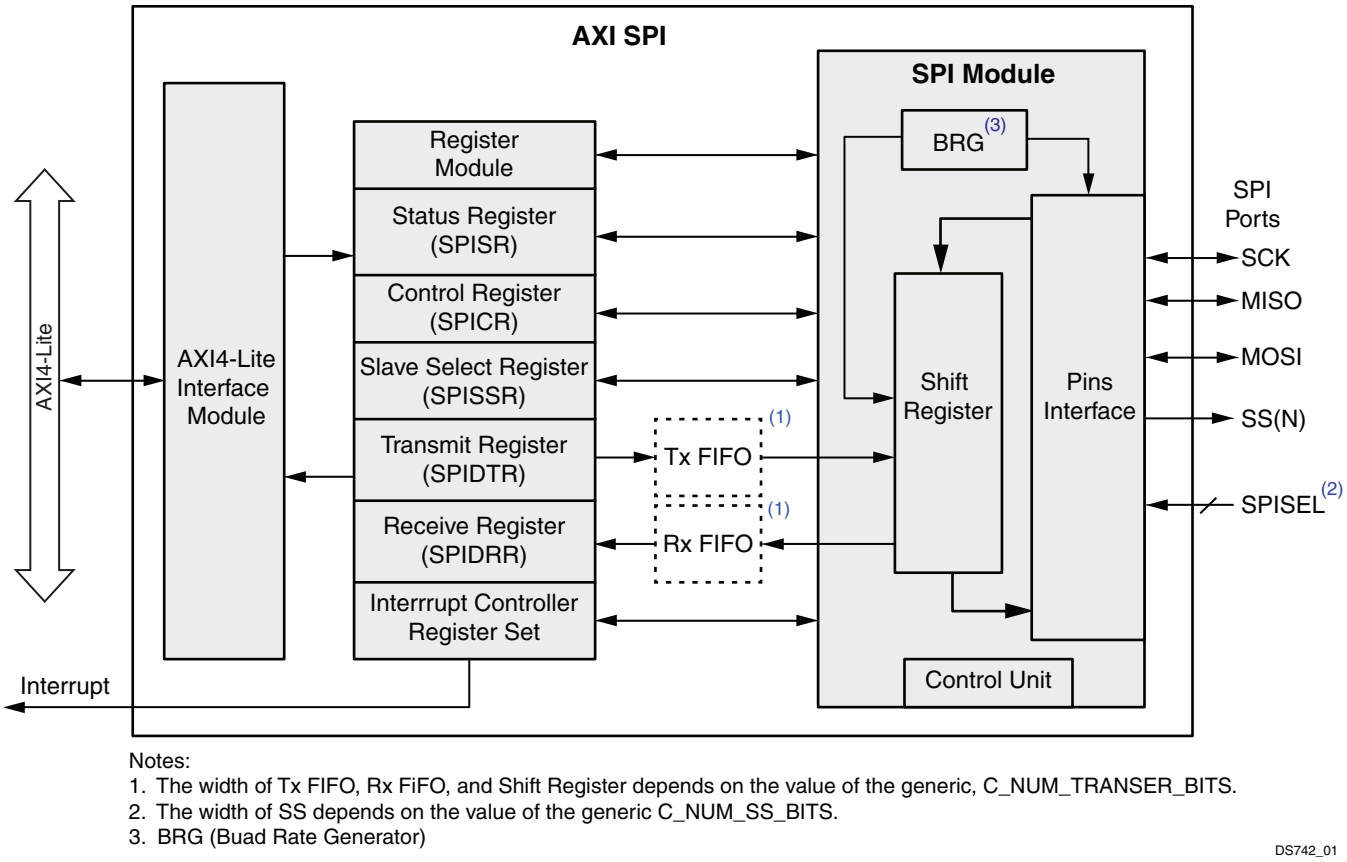

Figure 1: **AXI SPI IP Core Top-Level Block Diagram**

The AXI SPI IP core is a full-duplex synchronous channel that supports a four-wire interface (receive, transmit, clock and slave-select) between a master and a selected slave. The core supports Manual Slave Select Mode as the default mode of operation. This mode allows the user to manually control the slave select line using the data written to the slave select register. This allows transfers of an arbitrary number of elements without toggling the slave select line between elements. However, the user must toggle the slave select line before starting a new transfer.

The other mode of operation is Automatic Slave Select Mode. In this mode the slave select line is toggled automatically after each element transfer. This mode is described in more detail in SPI Protocol with Automatic Slave Select Assertion.

The AXI SPI IP core supports continuous transfer mode; when configured as a master the transfer continues until the data is available in transmit register/FIFO. This capability is provided in both manual and automatic slave select modes.

When the core is configured as a slave and if inadvertently its slave select line (SPISEL) goes high (inactive state) in between the data element transfer, then the current transfer is aborted. Again if the slave select line goes low then the aborted data element is transmitted again. The core allows 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, 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 does impact the achievable performance in terms of frequency and resource utilization. All of the SPI and INTR registers are 32-bit wide. The core supports only 32-bit word access to all SPI and INTR register modules.

AXI4-Lite IP IPIF Interface (IPIF): The AXI4-Lite IP Interface (IPIF) provides the interface to the AXI4-Lite to IP Interconnect (IPIC). The read and write transactions at the AXI4-Lite interface are translated into equivalent IP Interconnect (IPIC) transactions. See [Ref 4] for more information about the IPIC.

SPI Register Module: The SPI Register Module includes all memory mapped registers (as shown in Figure 1). It interfaces to the AXI. It consists of Status Register, Control Register, N-bit Slave Select Register (N ≤ 32) and a pair of Transmit/Receive Registers.

Interrupt Controller Register set Module: The Interrupt Controller Register set Module consists of interrupt related registers, namely: 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. It is the heart of core.

Optional FIFOs: The Tx FIFO and Rx FIFO are implemented on both transmit and receive paths when enabled by the parameter C_FIFO_EXIST. The width of Tx FIFO and Rx FIFO are the same and depend on the generic C_NUM_TRANSFER_BITS. When the FIFOs are enabled, their depth is fixed at 16.

Design Parameters

To allow the user to obtain an AXI 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 SPI IP core. The features that can be parameterized are as shown in Table 1. In addition to the parameters listed in this table, there are also parameters that are inferred for each AXI interface in the Embedded Development Kit (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 6].

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, 7series, 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, 64	32	integer
AXI SPI IP Core Parameters					
G6	Include receive and transmit FIFOs	C_FIFO_EXIST	0 = FIFOs not included 1 = FIFOs included	1	integer

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

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G7	SPI clock frequency ratio	C_SCK_RATIO	2[3], 4, 8, Nx16 for N = 1, 2, 3, ...	32	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	integer

Notes: Notes:

1. The range C_BASEADDR to C_HIGHADDR is the address range for the AXI SPI IP. This range is subject to restrictions to accommodate the simple address decoding scheme that is employed. The size, C_HIGHADDR - C_BASEADDR + 1, must be a power of two and must be at least 0x80 to accommodate all AXI SPI IP core registers. However, a larger power of two can be chosen to reduce decoding logic. C_BASEADDR must be aligned to a multiple of the range size.
2. No default value is specified to ensure that an actual value appropriate to the system is set. The values must be set by the user.
3. C_SCK_RATIO = 2 is not supported when the AXI SPI IP core is configured as slave. Read the Precautions to be Taken while Assigning the C_SCK_RATIO Parameter section carefully when using this parameter.

Input/Output (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)

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

Port	Signal Name	Interface	I/O	Initial State	Description
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. When HIGH, this signal indicates 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.
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 Interface Signals					
P20	IP2INTC_Irpt	SPI	O	0	Interrupt control signal from SPI
P21	SCK_I	SPI	I	-	SPI bus clock input
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	MOSI_I	SPI	I	-	Master output slave input
P25	MOSI_O	SPI	O	1	Master output slave input
P26	MOSI_T	SPI	O	1	3-state enable master output slave input. Active Low.
P27	MISO_I	SPI	I	-	Master input slave output
P28	MISO_O	SPI	O	1	Master input slave output
P29	MISO_T	SPI	O	1	3-state enable master input slave output. Active Low.
P30	SPISEL[(1)]	SPI	I	1	Local SPI slave select active Low input Must be set to 1 in idle state
P31	SS_I[(C_NUM_SS_BITS - 1):0]	SPI	I	-	Input one-hot encoded. This signal is a dummy signal and is used in the design as chip-select input.
P32	SS_O[(C_NUM_SS_BITS - 1):0]	SPI	O	1	Output one-hot encoded, active Low slave select vector of length n.
P33	SS_T	SPI	O	1	3-state enable for slave select. Active Low.

1. SPISEL signal is used as a slave select line when AXI SPI is configured as a slave.

Parameters - I/O Signal Dependencies

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

Table 3: **Parameters - Signal Dependencies**

Generic or Port	Name	Affects	Depends	Relationship Description
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	P31, P32	-	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 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.
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.
P31	SS_I[(C_NUM_SS_BITS - 1):0]	-	G8	The number of SS_I pins are generated based on C_NUM_SS_BITS.
P32	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 SPI IP core registers. The Transmit FIFO Occupancy Register and the Receive FIFO Occupancy Register exist only when C_FIFO_EXIST = 1.

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	0x25	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	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 (1)	Read	0x0	Transmit FIFO Occupancy Register
C_BASEADDR + 78	SPI Receive FIFO Occupancy Register(1)	Read	0x0	Receive FIFO Occupancy Register
Interrupt Controller Grouping				
C_BASEADDR + 1C	DGIER	R/W	0x0	Device Global Interrupt Enable Register

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

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_BASEADDR + 20	IPISR	R/TOW(2)	0x0	IP Interrupt Status Register
C_BASEADDR + 28	IPIER	R/W	0x0	IP Interrupt Enable Register

Note:

1. Exists only when C_FIFO_EXIST = 1.
2. 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 the programmer to reset the core independent of other cores in the systems. To activate software generated reset, the value of 0x0000_000A must be written to this register. 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 2 and described in Table 5. An attempt to read this register returns undefined data.

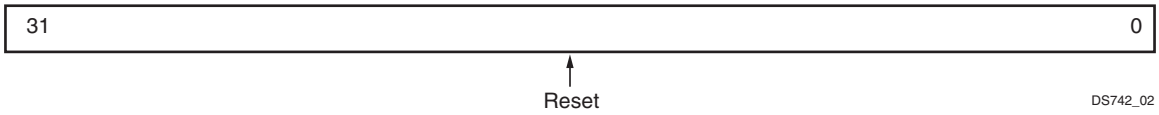

Figure 2: **Software Reset Register (C_BASEADDR + 0x40)**

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 SPI IP core.

SPI Control Register (SPICR)

The SPI Control Register (SPICR) gives the programmer control over various aspects of the AXI SPI IP core. The bit assignment in the SPICR is shown in Figure 3 and described in Table 6.

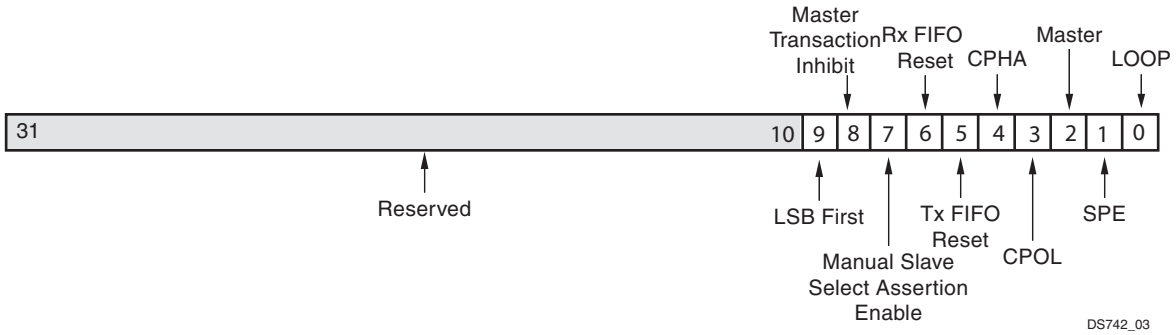

Figure 3: **SPI Control Register (C_BASEADDR + 0x60)**

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

Bit(s)	Name	Core Access	Reset Value	Description
31 - 10	Reserved	N/A	N/A	Reserved
9	LSB First	R/W	0	**LSB First** 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
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 anytime 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
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. This bit is unassigned when the AXI SPI IP core is not configured with FIFOs. 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. This bit is unassigned when the AXI SPI IP core is not configured with FIFOs. 0 = Transmit FIFO normal operation 1 = Reset transmit FIFO pointer
4	CPHA	R/W	0	**Clock Phase (CPHA)** 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 (CPOL)** 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)** Setting this bit configures the SPI device as a master or a slave. 0 = Slave configuration 1 = Master configuration
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 ignored. 1 = SPI system enabled. Master outputs active (for example, MOSI and SCK in idle state) and slave outputs become active if $\overline{SS}$ becomes asserted. Master starts a transfer when transmit data is available.
0	LOOP	R/W	0	**Local Loopback Mode** Enables local loopback operation and is functional only in 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.

SPI Status Register (SPISR)

The SPI Status Register (SPISR) is a read-only register that gives the programmer visibility of the status of some aspects of the AXI SPI IP core. The bit assignment in the SPISR is shown in Figure 4 and described in Table 7. Writing to the SPISR does not modify the register contents.

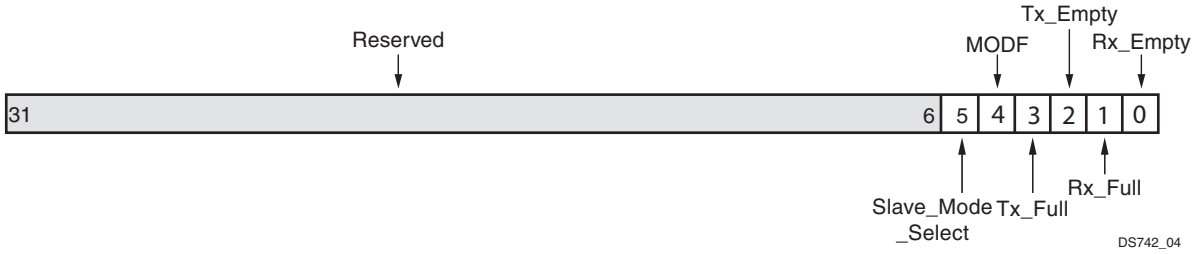

Figure 4: **SPI Status Register (C_BASEADDR + 0x64)**

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

Bit(s)	Name	Core Access	Reset Value	Description
31 - 6	Reserved	N/A	N/A	Reserved
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 0 = Asserted when core configured in slave mode and selected by external SPI master
4	MODF	Read	0	**Mode-Fault Error Flag (Mode-fault Error)** This flag is set if the $\overline{SS}$ signal goes active while the SPI device is configured as a master. MODF is automatically cleared by reading the SPISR. A low-to-high MODF transition generates a single-cycle strobe interrupt. 0 = No error 1 = Error condition detected
3	Tx_Full	Read	0	**Transmit Full** When a transmit FIFO exists, this bit is set high when the transmit FIFO is full. ***Note:*** When FIFOs do not exist, this bit is set high when an AXI write to the transmit register has been made. This bit is cleared when the SPI transfer is completed.
2	Tx_Empty	Read	1	**Transmit Empty** When a transmit FIFO exists, this bit is set high when the transmit FIFO is empty. The occupancy of the FIFO is decremented with the completion of each SPI transfer. ***Note:*** When FIFOs do not exist, this bit is set with the completion of an SPI transfer. Either with or without FIFOs, this bit is cleared upon a AXI write to the FIFO or transmit register.
1	Rx_Full	Read	0	**Receive Full** When a receive FIFO exists, this bit is set high when the receive FIFO is full. The occupancy of the FIFO is incremented with the completion of each SPI transaction. ***Note:*** When FIFOs do not exist, this bit is set high when an SPI transfer has completed. Rx_Empty and Rx_Full are complements in this case.
0	Rx_Empty	Read	1	**Receive Empty** When a receive FIFO exists, this bit is set high when the receive FIFO is empty. The occupancy of the FIFO is decremented with each FIFO read operation. ***Note:*** When FIFOs do not exist, this bit is set high when the receive register has been read. This bit is cleared at the end of a successful SPI transfer.

SPI Data Transmit Register (SPIDTR)

This register is written to with the data to be transmitted on the SPI bus. After the SPE bit is set to 1 in master mode or SPISEL is active in the slave mode, the data is transferred from the SPIDTR to the shift register. If a transfer is in progress, the data in the SPIDTR is loaded in the shift register as soon as the data in the shift register is transferred to the SPIDRR and a new transfer starts. The data is held in the SPIDTR until a subsequent write overwrites the data. The SPIDTR is shown in Figure 5, while Table 8 shows specifics of the data format. When a transmit FIFO exists, data is written directly in the FIFO and the first location in the FIFO is treated as the SPIDTR. The pointer is decremented after completion of each SPI transfer.

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, then the AXI write transaction completes with an error condition. Reading to the SPIDTR is not allowed and the read transaction results in undefined data.

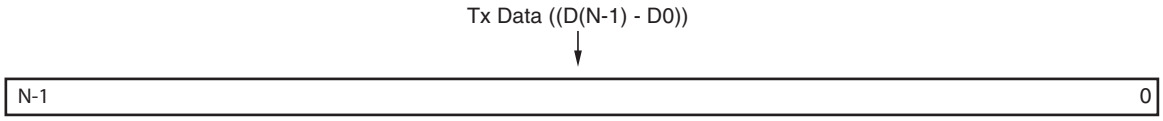

Figure 5: **SPI Data Transmit Register (C_BASEADDR + 0x68)**

Table 8: **SPI Data Transmit Register (SPIDTR) 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. N = 8 when C_NUM_TRANSFER_BITS = 8 N = 16 when C_NUM_TRANSFER_BITS = 16 N = 32 when C_NUM_TRANSFER_BITS = 32

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.

SPI Data Receive Register (SPIDRR)

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 SPIDRR is updated following each completed transaction only if the SPIDRR was read prior to the last SPI transfer. If the SPIDRR was not read, and therefore 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.

For both master and slave SPI devices 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, then the AXI read transaction completes successfully with undefined data. Writes to the SPIDRR do not modify the register contents and return with a successful OK response. The SPIDRR is shown in Figure 6, while the specifics of the data format is described Table 9.

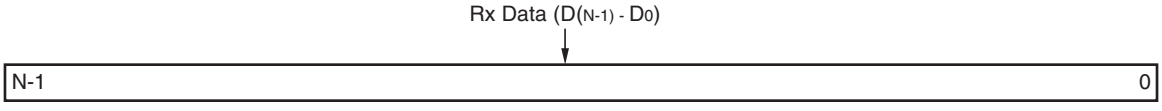

Figure 6: **SPI Data Receive Register (C_BASEADDR + 0x6C)**

Table 9: **SPI Data Receive Register (SPIDRR) 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	0	N-bit SPI receive data. N can be 8, 16 or 32. N = 8 when C_NUM_TRANSFER_BITS = 8 N = 16 when C_NUM_TRANSFER_BITS = 16 N = 32 when C_NUM_TRANSFER_BITS = 32

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.

SPI Slave Select Register (SPISSR)

This register contains an active Low, one-hot encoded slave select vector $\overline{SS}$ of length N, where N is the number of slaves set by parameter C_NUM_SS_BITS. The $\overline{SS}$ occupies the right-most bits of the register. At most, one bit can be asserted low. This bit denotes the slave with whom the local master communicates. The bit assignment in the SPISSR is shown in Figure 7 and described in Table 10.

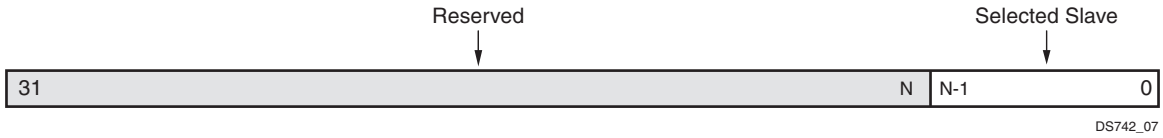

Figure 7: **SPI Slave Select Register (C_BASEADDR + 0x70)**

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 SPI IP core is configured with FIFOs (C_FIFO_EXIST = 1). If it is present and if the Transmit FIFO is not empty, the register contains a four-bit, right-justified value that is one less than the number of elements in the FIFO (occupancy minus one). This register is read-only. A write to it (or a read when the FIFO is empty) does not affect the register contents. The only reliable way to determine that the Tx FIFO is empty is by reading the Tx_Empty status bit in the SPI Status Register or the Data Transmit Register (DTR) Empty bit in the Interrupt Status Register. The Transmit FIFO Occupancy register is shown in Figure 8, while the specifics of the data format are described in Table 11.

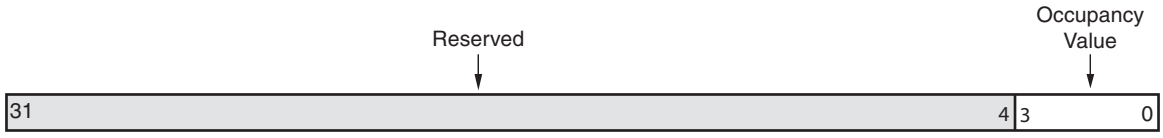

Figure 8: **SPI Transmit FIFO Occupancy Register (C_BASEADDR + 0x74)**

Table 11: **SPI Transmit FIFO Occupancy Register Description (C_BASEADDR + 0x74)**

Bit(s)	Name	Core Access	Reset Value (hex)	Description
31 - 4	Reserved	N/A	N/A	Reserved
3 - 0	Occupancy Value	Read	0	Bit 3 is the MSB. The binary value plus 1 yields the occupancy.

SPI Receive FIFO Occupancy Register (Rx_FIFO_OCY)

The SPI Receive FIFO Occupancy Register is present if and only if the AXI SPI IP core is configured with FIFOs (C_FIFO_EXIST = 1). If it is present and if the Receive FIFO is not empty, the register contains a four-bit, right-justified value that is one less than the number of elements in the FIFO (occupancy minus one). This register is read-only. A write to it (or of a read when the FIFO is empty) does not affect the register contents. The only reliable way to determine that the Rx 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 9, while the specifics of the data format are described in Table 12.

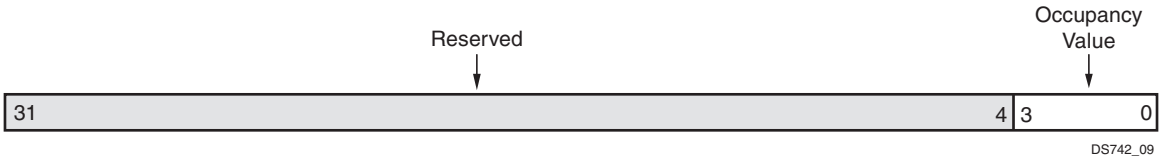

Figure 9: **SPI Receive FIFO Occupancy Register (C_BASEADDR + 0x78)**

Table 12: **SPI Receive FIFO Occupancy Register Description (C_BASEADDR + 0x78)**

Bit(s)	Name	Core Access	Reset Value (hex)	Description
31- 4	Reserved	N/A	N/A	Reserved
3 - 0	Occupancy Value	Read	0	Bit 3 is the MSB. The binary value plus 1 yields the occupancy.

Interrupt Register Set Description

The AXI SPI IP core has a number of distinct interrupts that are sent to the interrupt controller submodule. The AXI SPI IP interrupt controller allows each interrupt to be enabled independently (via 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. Setting the parameter C_FIFO_EXIST = 1 makes available almost all of the interrupts shown in Table 14 when the core is configured in the master mode. Setting the parameter C_FIFO_EXIST = 0 makes available all of the interrupts except bit(6), Tx FIFO Half Empty and bit(8) Data Receive Register (DRR) Not Empty, which is not present in this case.

Device Global Interrupt Enable Register (DGIER)

The Device Global Interrupt Enable Register is used to globally enable the final interrupt output from the interrupt controller as shown in Figure 10 and described in Table 13. This bit is a read/write bit and is cleared upon reset.

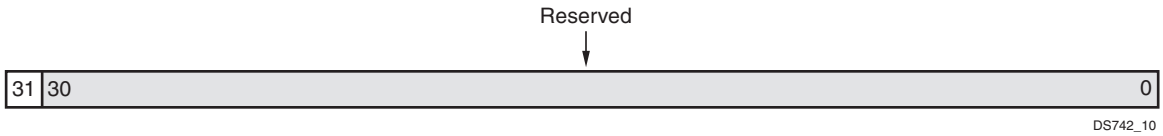

Figure 10: **Device Global Interrupt Enable Register (DGIER) (C_BASEADDR + 0x1C)**

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 nine unique interrupt conditions are possible depending upon whether the system is configured with FIFOs or not as well as if it is configured in master mode or slave mode. A system without FIFOs has seven interrupts. The interrupt controller has the 32-bit Interrupt Status Register that can enable each interrupt independently. This register collects all of the interrupt events. Bit assignments are shown in Figure 11 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.

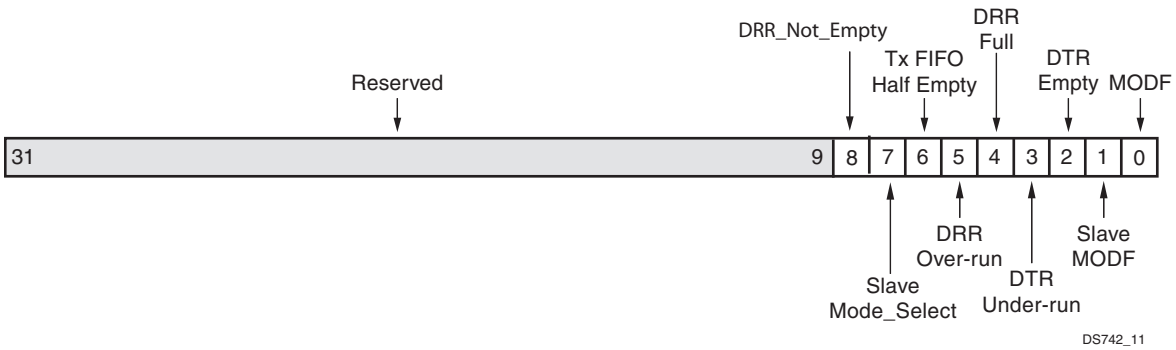

Figure 11: **IP Interrupt Status Register (IPISR) (C_BASEADDR + 0x20)**

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

Bit(s)	Name	Access	Reset Value	Description
31 - 9	Reserved	N/A	N/A	Reserved
8	DRR_Not_ Empty	R/TOW(1)	0	**DRR Not Empty** IPISR bit(8) is the DRR Not Empty bit. The assertion of this bit is applicable only in the case where C_FIFO_EXIST = 1 and the core is configured in slave mode. This bit is set when the DRR FIFO receives the first data value during the SPI transaction. This bit is set by one-clock period strobe to the interrupt register when the core receives first data beat. ***Note:*** The assertion of this bit is applicable only when the C_FIFO_EXIST = 1 and the core is configured in slave mode. In C_FIFO_EXIST = 0, this bit always return 0. It is recommended to use this bit only when C_FIFO_EXIST = 1 and the core is configured in slave mode.
7	Slave_ Select_ Mode	R/TOW(1)	0	**Slave Select Mode** IPISR bit(7) is the Slave Select Mode bit. The assertion of this bit is applicable only when the core is configured in slave mode. This bit is set when the other SPI master core selects the core by asserting the Slave Select line. This bit is set by one-clock period strobe to the interrupt register. ***Note:*** This bit is applicable only when the core is configured in the slave mode.
6	Tx FIFO Half Empty	R/TOW(1)	0	**Transmit FIFO Half Empty** IPISR bit(6) is the transmit FIFO half-empty interrupt. This bit is set by a one-clock period strobe to the interrupt register when the occupancy value is decremented from "1000" to "0111". The value "0111" means there are 8 elements in the FIFO to be transmitted. ***Note:*** This interrupt exists only if the AXI SPI IP core is configured with FIFOs.
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.

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

Bit(s)	Name	Access	Reset Value	Description
4	DRR Full	R/TOW(1)	0	**Data Receive Register/FIFO Full** IPISR bit(4) is the data receive register full interrupt. Without FIFOs, this bit is set at the end of an SPI element (An element can be a byte, half-word or word depending on the value of 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 filled by a one-clock period strobe to the interrupt register.
3	DTR Under run	R/TOW(1)	0	**Data Transmit Register/FIFO Underrun** IPISR bit(3) is the data transmit register/FIFO underrun interrupt. This bit is set at the end of an SPI element transfer by a one-clock period strobe to the interrupt register when data is requested from an "empty" transmit register/FIFO by the SPI core logic to perform an SPI transfer. This can occur only when the SPI device is configured as a slave and is enabled by the SPE bit as set. All zeros are loaded in the shift register and transmitted by the slave in an underrun 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 Ending 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 SPIF interrupt bit. In master mode if this bit is set to 1, no more SPI transfers are permitted.
1	Slave MODF	R/TOW(1)	0	**Slave Mode-Fault Error** IPISR bit(1) is the slave mode-fault error flag. This interrupt is generated if the SS signal goes active while the SPI device is configured as a slave but is not enabled. This bit is set immediately upon SS going active and continually set if SS is active and the device is not enabled.
0	MODF	R/TOW(1)	0	**Mode-Fault Error** IPISR bit(0) is the mode-fault error flag. This interrupt is generated if the SS signal goes active while the SPI device is configured as a master. This bit is set immediately upon SS going active.

Notes:

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

IP Interrupt Enable Register (IPIER)

The interrupt controller IPIER register allows the system interrupt output to be active. This interrupt is generated if the enabled bit in IPIER detects any activity on the corresponding IPISR bit. The IPIER has an enable bit for each defined bit of the IPISR as shown in Figure 12 and described in Table 15. All bits are cleared upon reset.

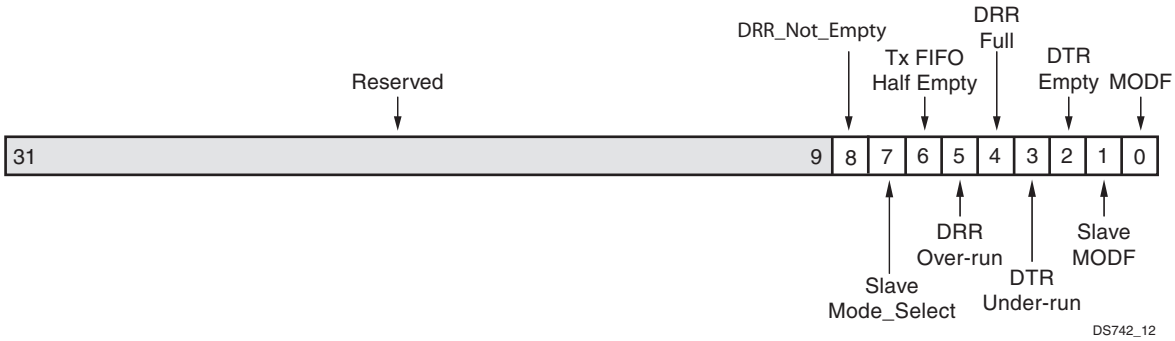

Figure 12: **IP Interrupt Enable Register (IPIER) (C_BASEADDR + 0x28)**

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

Bit(s)	Name	Access	Reset Value	Description
31 - 9	Reserved	N/A	N/A	Reserved
8	DRR_Not_Empty	R/W	0	**DRR_Not_Empty** 0 = Disabled 1 = Enabled ***Note:*** The setting of this bit is applicable only when C_FIFO_EXIST = 1 and the core is configured in slave mode. If C_FIFO_EXIST = 0, setting of this bit has no effect, which means that this bit is not set in IPIER. Therefore, it is recommended to use this bit only in C_FIFO_EXIST = 1 condition when the core is configured in slave mode.
7	Slave_Select_Mode	R/W	0	**Slave_Select_Mode** 0 = Disabled 1 = Enabled This bit is applicable only when the core is configured in slave mode. In master mode, setting this bit has no effect.
6	Tx FIFO Half Empty	R/W	0	**Transmit FIFO Half Empty** 0 = Disabled 1 = Enabled ***Note:*** This bit is meaningful only if the AXI SPI IP core is configured with FIFOs.
5	DRR Overrun	R/W	0	**Receive FIFO Overrun** 0 = Disabled 1 = Enabled
4	DRR Full	R/W	0	**Data Receive Register/FIFO Full** 0 = Disabled 1 = Enabled
3	DTR Underrun	R/W	0	**Data Transmit FIFO Underrun** 0 = Disabled 1 = Enabled
2	DTR Empty	R/W	0	**Data Transmit Register/FIFO Empty** 0 = Disabled 1 = Enabled

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

Bit(s)	Name	Access	Reset Value	Description
1	Slave MODF	R/W	0	**Slave Mode-Fault Error Flag** 0 = Disabled 1 = Enabled
0	MODF	R/W	0	**Mode-Fault Error Flag** 0 = Disabled 1 = Enabled

Design Description

SPI Device Features

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

- Supports multi-master configuration within the Field Programmable Gate Array (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 via 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 via a bit in the SPICR for SPI master devices.
- Multi-master environment supported (implemented with 3-state drivers and requires software arbitration for possible conflict). See the SPI in Multi-Master Configuration section.
- Multi-slave environment supported (automatic generation of additional slave select output signals for the master).
- Supports maximum SPI clock rates up to one-half of the AXI clock rate in master mode and one-fourth of the AXI clock rate in slave modes. C_SCK_RATIO = 2 is not supported in Slave Mode (due to 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.
- The Write Collision error (WCOL) flag is not supported as a write collision error as described in the M68HC11 reference manual. The user must not write to the transmit register when an SPI data transfer is in progress.
- Back-to-back transactions are supported, which means 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.
- This IP cannot be used for FPGA bitstream programming through the SPI interface during power-on reset state.
- The data transfer and registering mechanism of this core is synchronized with the AXI clock. User should take care while configuring the IP. See the timing parameters for the targeted device while configuring the core and the C_SCK_RATIO parameter.

SPI in Multi-Master Configuration

The SPI bus to a given slave device (N-th device) consists of four wires, Serial Clock (SCK), Master Out Slave In (MOSI), Master In Slave Out (MISO) and Slave Select ($\overline{SS}$(N)). The signals SCK, MOSI and MISO are shared for all slaves and masters. See Figure 13.

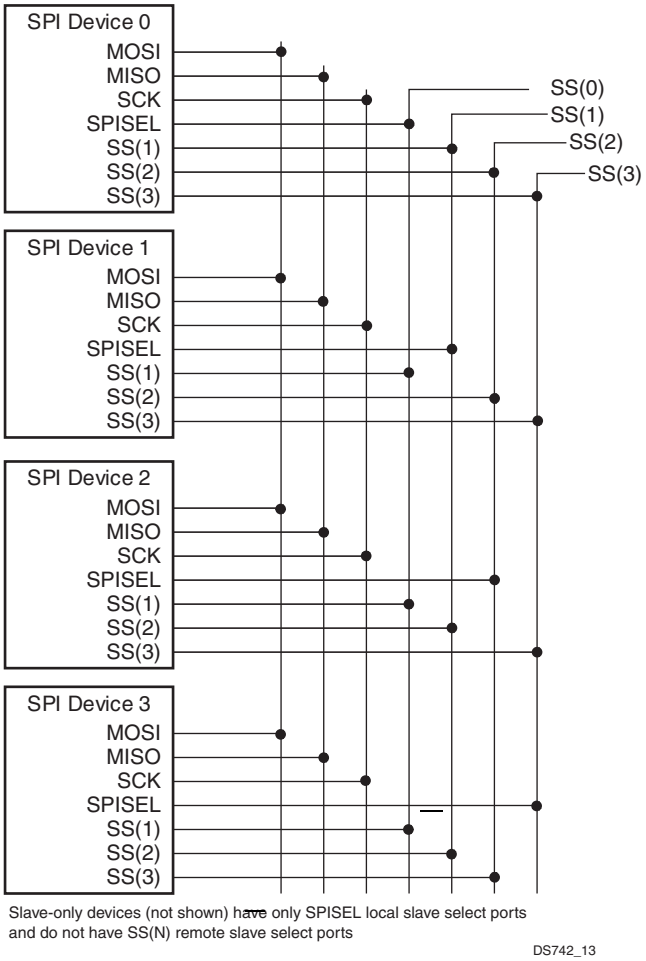

Figure 13: **Multi-Master Configuration Block Diagram**

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 Xilinx Platform Generator when interfacing to an external SPI slave device. Multiple SPI master/slave devices are shown in Figure 13.

Optional FIFOs

The user has the option to include FIFOs in the AXI SPI IP core as shown in Figure 1. Because SPI is full-duplex, both transmit and receive FIFOs are instantiated as a pair.

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. Because burst mode is not supported, both transmit and receive FIFOs are 16 elements deep and are accessed via single AXI transactions.

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, then 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 underrun. See the section on 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, then 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, then a receive FIFO overflow interrupt is issued. When this happens, all data attempted to be written to the full receive FIFO by the SPI module is lost.

SPI transfers, when the AXI SPI IP core is configured with FIFOs, 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, then the SPI transfer begins immediately after the write to the master transmit FIFO. If the FIFO is emptied via 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 up to 16 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.

Local Master Loopback Operation

Local master loopback operation, although not included in the M68HC11 reference manual, has been implemented to expedite testing. This operation is selected via setting the loopback bit in the SPICR; 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.

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 detects when an SPI device configured as a master is selected (that is, its $\overline{SS}$ bit is asserted) by another SPI device 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, this might require clearing the FIFOs.

A similar error detection mechanism has been implemented for SPI slave devices. The error detected is when a SPI device configured as a slave but is not enabled and 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 and overrun conditions error detection is also provided. Underrun conditions can happen only in slave mode operation. This happens when a master commands a transfer but the slave does not have data in the transmit register or FIFO for transfer. In this case, the slave underrun interrupt is asserted and the slave shift register is loaded with all zeros for transmission. Overrun can happen 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.

Precautions to be Taken while Assigning the C_SCK_RATIO Parameter

The AXI SPI IP core is tested in hardware with the SPI slave devices like serial EEPROMs, ATMEL, STMicro-Electronics and Intel flash memories. Read the data sheet of the targeted SPI slave flash memory or EEPROMs for maximum speed of operation. It is the responsibility of the user to mention the correct values while deciding the AXI clock and selecting the C_SCK_RATIO parameter of the core. The AXI clock and the C_SCK_RATIO decide the clock at SCK pin of AXI SPI IP core. While using different external SPI slave devices, the C_SCK_RATIO should be set carefully and the maximum clock frequencies supported by all the external SPI slave devices should be taken into account.

SPI Slave Mode

The AXI SPI core can be configured in the slave mode by connecting the external master's slave select line to SPISEL and by setting bit 2 of SPI Control Register (SPICR) to '0'. All the incoming signals are synchronized to the AXI when C_SCK_RATIO > 4. Due to the tight timing requirements when C_SCK_RATIO = 4 the incoming SCK clock signal and its synchronized signals are used directly in the internal logic. Therefore it is required that the external clock be synchronized with the AXI clock when C_SCK_RATIO = 4. For other C_SCK_RATIO values, it is preferred, but might not be necessary, to have such synchronization.

During the slave mode operation it is strongly recommended to use the FIFO by setting C_FIFO_EXIST = 1. 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. After the master starts communication, with the core configured in slave mode, the core will transfer data until the data exists in its transmit FIFO. At the end of last data beat transmitted from slave FIFO, the core (in slave mode) generates the DTR Empty signal to notify that new data beats needed to be filled in its transmit FIFO before further communication is started.

SPI Transfer Formats

SPI Clock Phase and Polarity Control

Software can select any of four combinations of serial clock (SCK) phase and polarity with programmable bits in the SPICR. The clock polarity (CPOL) bit selects an active High (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, 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 for the master SPI device and the selected slave device.

The first SCK cycle begins with a transition of SCK signal from its idle state and this 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 the user's responsibility to ensure that the data is properly loaded in the SPISSR register prior to the first SCK edge.

The $\overline{SS}$ signal is toggled for all CPHA configurations and there is no support for SPISEL being held low. It is required that all $\overline{SS}$ signals be routed between SPI devices internally to the FPGA. Toggling the $\overline{SS}$ signal reduces FPGA resources. The different transfer formats are described in the following sections.

CPHA Equals Zero Transfer Format

Figure 14 shows the timing diagram for an SPI 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.

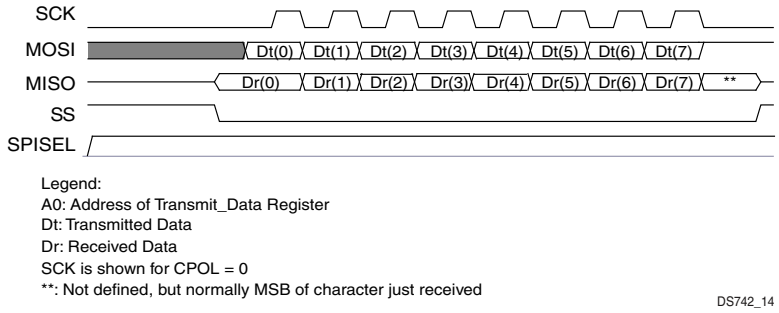

Figure 14: **Data Write-Read Cycle on SPI Bus with CPHA = 0 and SPICR(24) = 0 for 8-bit Data**

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

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

The MOSI and MISO ports behave differently depending on whether the SPI device is configured as a master or a slave. When configured as a master, the MOSI port is a serial data output port and the MISO is a serial data input port. The opposite is true when the device is configured as a slave; the MISO port is a slave serial data output port and the MOSI 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 MISO and MOSI signals) and requires the software to provide arbitration to prevent possible contention errors.

All SCK, MOSI, and MISO 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 MOSI pins to the SCK and MOSI pins of the slaves. The uniquely selected slave device drives data out from its MISO pin to the MISO master pin, thus realizing full-duplex communication. The Nth bit of the $\overline{SS}$(N) signal selects the Nth SPI slave with an active Low signal. All other slave devices ignore both SCK and MOSI signals. In addition, the non-selected slaves (that is, $\overline{SS}$ pin high) drive their MISO pin to 3-state so as not to interfere with SPI bus activities.

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

The user must 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. Recall that the I/O drivers can be configured for different drive strengths as well as internal pull-ups. The 3-state signals for multiple external slaves can be implemented as per system design requirements, but the external bus must follow the SPI M68HC11 specifications.

CPHA Equals One Transfer Format

With CPHA = 1, the first SCK cycle begins with an edge on the SCK line from its inactive level to active level (rising or falling depending on CPOL) as shown in Figure 15. 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 SAXI_Clk and SCK, respectively.

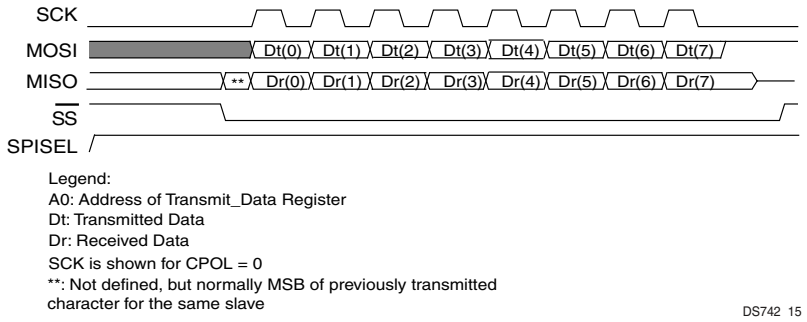

Figure 15: **Data Write-Read Cycle on SPI Bus with CPHA = 1 and SPICR(24) = 0 for 8-bit Data**

SPI Protocol Slave Select Assertion Modes

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

SPI Protocol with Automatic Slave Select Assertion

This section describes the SPI protocol where slave select ($\overline{SS}$(N)) is asserted automatically by the SPI master device (SPICR bit(7) = 0). This is the configuration mode provided to allow transfer of data with automatic toggling of the slave select ($\overline{SS}$) signal until all the elements are transferred. In this mode the data in the SPISSR register appears on the $\overline{SS}$(N) output when the new transfer starts. After every byte (or element) transfer, the $\overline{SS}$(N) output goes to 1. The data in SPISSR register again appears on the $\overline{SS}$(N) output at the beginning of a new transfer. The user does not need to manually control the slave select signal.

SPI Protocol with Manual Slave Select Assertion

This section briefly describes the SPI protocol where the slave select ($\overline{SS}$(N)) is manually asserted by the user (that is, SPICR bit(7) = 1). This is the configuration mode provided to allow transfers of an arbitrary number of elements without toggling the 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. Therefore, when manual slave select mode is used, the SPI master must be enabled first (SPICR bit(7) = 1) to assert 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 used before the first transaction and after any transaction that is allowed to complete. When the preceding rules are followed, the timing is the same as presented for the automatic slave select assertion mode with the exception that assertion of the slave select signal and the number of elements transferred is controlled by the user.

Beginning and Ending SPI Transfers

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

Transfer Beginning Period

The definition of the transfer beginning period for the AXI SPI IP core is consistent with the M68HC11 reference manual. This manual can be referenced for more details. All SPI transfers are started and controlled by a master SPI device. As a slave, the processor considers a transfer to begin with the first SCK edge or the falling edge of $\overline{SS}$, depending on the CPHA format selected. When CPHA equals zero, the falling edge of $\overline{SS}$ indicates the beginning of a transfer. When CPHA equals one, the first edge on the SCK indicates the start of the transfer. In either CPHA format, a transfer can be aborted by deasserting the $\overline{SS}$(N) signal. This 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 SPIDTR 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 Ending Period

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

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

When the SPI device is configured as a master without FIFOs, the following occurs:

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

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 14 and Figure 15. However, the internal master clock provides this SCK edge which prompts the setting/clearing of the bits noted.

In this design, a counter was implemented which allows the simultaneous setting of SPISR and IPISR bits for both master and slave SPI devices. External SPI slave devices can use an internal 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 SPI IP core cannot be used with external slave devices that do not use the AXI clock.

When the AXI SPI IP core is configured with FIFOs and a series of consecutive SPI 8-bit/16-bit/32-bit element transfers are performed, 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/clearing of the master bits for both cases of CPHA = 0 and CPHA = 1. Recall 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.

SPI Registers Flow Description

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 via SPICR bit(24) assertion.

This flow allows the transfer of N number of byte/half-word/word by toggling of the slave select vector just once. This is the default mode of operation. The user can follow the subsequent steps to successfully complete an SPI transaction:

1. Start from proper state including SPI bus arbitration.
2. Configure Device Global Interrupt Enable Register (DGIER) and IPIER registers as desired.
3. Configure target slave SPI device as required. This includes configuration of the DTR and Control Register of slave SPI core and enabling it.
4. Write initial data to master SPIDTR register/FIFO. This assumes that the SPI master is disabled.
5. Ensure the SPISSR register has all ones.
6. Write configuration data to master SPI device SPICR as desired including setting bit(7) for manual asserting of $\overline{SS}$ vector and setting both enable bit and master transfer inhibit bit. This initializes SCK and MOSI but inhibits transfer.
7. Write to SPISSR to manually assert $\overline{SS}$ vector.
8. Write the preceding configuration data to master SPI device SPICR, but clear inhibit bit which starts transfer.
9. Wait for interrupt (typically IPISR bit(4)) or poll status for completion. Wait time depends on SPI clock ratio.
10. Set master transaction inhibit bit to service interrupt request. Write new data to master register/FIFOs and slave device and then clear master transaction inhibit bit to continue N 8-bit element transfer. An overrun of the SPIDRR register/FIFO can occur if the SPIDRR register/FIFOs are not read properly. In addition, SCK will have *stretched* idle levels between element transfers (or groups of element transfers if utilizing FIFOs) and MOSI 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 previous two steps until all data is transferred.
12. Write all ones to SPISSR or exit manual slave select assert mode to deassert $\overline{SS}$ vector while SCK and MOSI are in the idle state.
13. Disable devices as desired.

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

Follow these steps to complete an SPI transaction:

1. Start from proper state including SPI bus arbitration.
2. Configure master DGIER and IPIER. Also configure slave DGIER and IPIER registers as desired.

3. Write configuration data to master SPI device SPICR as required.
4. Write configuration data to slave SPI device SPICR as required.
5. Write the active Low, one-hot encoded slave select address to the master SPISSR.
6. Write data to slave SPIDTR register as required.
7. Write data to master SPIDTR register to start transfer.
8. Wait for interrupt (typically IPISR bit(4)) or poll status for completion.
9. Read IPISR of both master and slave SPI devices as required.
10. Perform interrupt requests as required.
11. Read SPISR of both master and slave SPI devices as required.
12. Perform actions as required or dictated by SPISR data.

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

User can follow the subsequent steps to successfully complete an SPI transaction:

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

SPI master and slave devices with FIFOs where some initial data is written to FIFOs, the SPI transfer is started, data is written to the FIFOs as fast or faster than the SPI transfer and multiple discrete 8-bit transfers are performed (optional mode).

The user can follow the subsequent steps to successfully complete an SPI transaction:

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

11. Read IPISR of both master and slave SPI devices as required.
12. Perform interrupt requests as required.
13. Read SPISR of both master and slave SPI devices as required.
14. Perform actions as required or dictated by SPISR data.

Steps to be followed when the core is configured in slave mode

1. Set the desired clock ratio using C_SCK_RATIO = up to 4 is allowed.
2. Fill the SPIDTR with the data; enable the interrupts as required for slave mode.
3. Enable the slave mode through SPICR and enable the core through SPE.
4. Connect the SPISEL input of the core to the Chip Select signal of external master SPI core.
5. Select the core using the active Low SPISEL bit .
6. After the core is selected by asserting the SPISEL, the core waits for the master's clock on SCK line and inputs on the MOSI line.
7. When the master starts the clock, data is exchanged between the master and slave on MISO and MOSI line respectively.
8. After each exchange of 8 bit of data, the core performs local housekeeping work. This includes storing the received data in the SPIDRR, loading the new data from SPIDTR into the local shift register for a new transfer and resetting the internal counter for the next transfer.
9. All the internal processes in step 8 take approximately 4 AXI clock cycles. It is preferable to allow an idle time of 6 clocks in between 2 consecutive transactions.
10. The core transfers the data until its DRR FIFO is empty. When the complete transfer is finished from the core the interrupt sets to indicate that the DRR is empty and the DTR is full. At this point, unless the DTR is re-filled, the core cannot communicate with master.
11. Read the DRR and re-fill the DTR and repeat the steps above.

Design Constraints

Timing Constraints

When the core is added in the MHS of XPS build, the timing constraints for the core are taken care at the system level by the XPS tool.

Design Implementation

Target Technology

The target FPGA technologies for the core are the supported device families listed in the LogiCORE IP Facts.

Device Utilization and Performance Benchmarks

Core Performance

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

The core resource utilization for various parameter combinations measured with a Artix™-7 FPGA as the target device are detailed in Table 16.

Table 16: **Performance and Resource Utilization Benchmarks on a Artix-7 FPGA (XC7A175Tdie-3)**

Parameter Values (other parameters at default values)				Device Resources			Performance
C_FIFO_EXIST	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	2	2	8	101	157	200	150
1	2	2	8	124	156	256	150
0	4	2	8	106	167	233	150
1	4	2	8	110	167	273	150
0	32	2	8	101	171	235	150
1	32	2	8	150	171	276	150

The core resource utilization for various parameter combinations measured with a Virtex®-7 FPGA as the target device are detailed in Table 17.

Table 17: **Performance and Resource Utilization Benchmarks on a Virtex-7 FPGA (xc7v285tffg484-3)**

Parameter Values (other parameters at default values)				Device Resources			Performance
C_FIFO_EXIST	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	2	2	8	101	156	200	200
1	2	2	8	96	156	253	200
0	4	2	8	111	167	222	200
1	4	2	8	115	167	273	200
0	32	2	8	101	171	232	200
1	32	2	8	128	171	271	200

The core resource utilization for various parameter combinations measured with a Kintex™-7 FPGA as the target device are detailed in Table 18.

Table 18: **Performance and Resource Utilization Benchmarks on a Kintex-7 FPGA (xc7v285tffg484-31)**

Parameter Values (other parameters at default values)				Device Resources			Performance
C_FIFO_EXIST	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	2	2	8	109	156	200	200
1	2	2	8	119	156	253	200
0	4	2	8	92	167	222	200
1	4	2	8	118	167	270	200
0	32	2	8	107	171	223	200
1	32	2	8	119	171	270	200

The core resource utilization for various parameter combinations measured with a Virtex®-6 FPGA as the target device are detailed in Table 19.

Table 19: **Performance and Resource Utilization Benchmarks on a Virtex-6 FPGA (xc6vlx130tff1156-1)**

Parameter Values (other parameters at default values)				Device Resources			Performance
C_FIFO_EXIST	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	2	2	8	79	138	166	209
1	2	2	8	92	139	208	192
0	4	2	8	96	149	290	193
1	4	2	8	101	149	224	206
0	32	2	8	82	153	200	192
1	32	2	8	112	153	224	182

The AXI SPI IP core resource utilization for various parameter combinations measured with a Spartan®-6 FPGA as the target device are detailed in Table 20.

Table 20: **Performance and Resource Utilization Benchmarks on a Spartan-6 FPGA (xc6slx45tfgg484-3)**

Parameter Values (other parameters at default values)				Device Resources			Performance
C_FIFO_EXIST	C_SCK_RATIO	C_NUM_SS_BITS	C_NUM_TRANSFER_BITS	Slices	Slice Flip-Flops	LUTs	Fmax (MHz)
0	2	2	8	85	138	155	125
1	2	2	8	96	138	206	128
0	4	2	8	109	153	192	133
1	4	2	8	105	150	230	127
0	32	2	8	92	155	182	135
1	32	2	8	109	154	235	130

System Performance

To measure the system performance (F_{MAX}) of this core, the core was added to a Virtex-6 FPGA system and a Spartan-6 FPGA system as the device under test (DUT) as illustrated in Figure 16.

Because the AXI 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 design's FPGA resources and timing usage can 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 21.

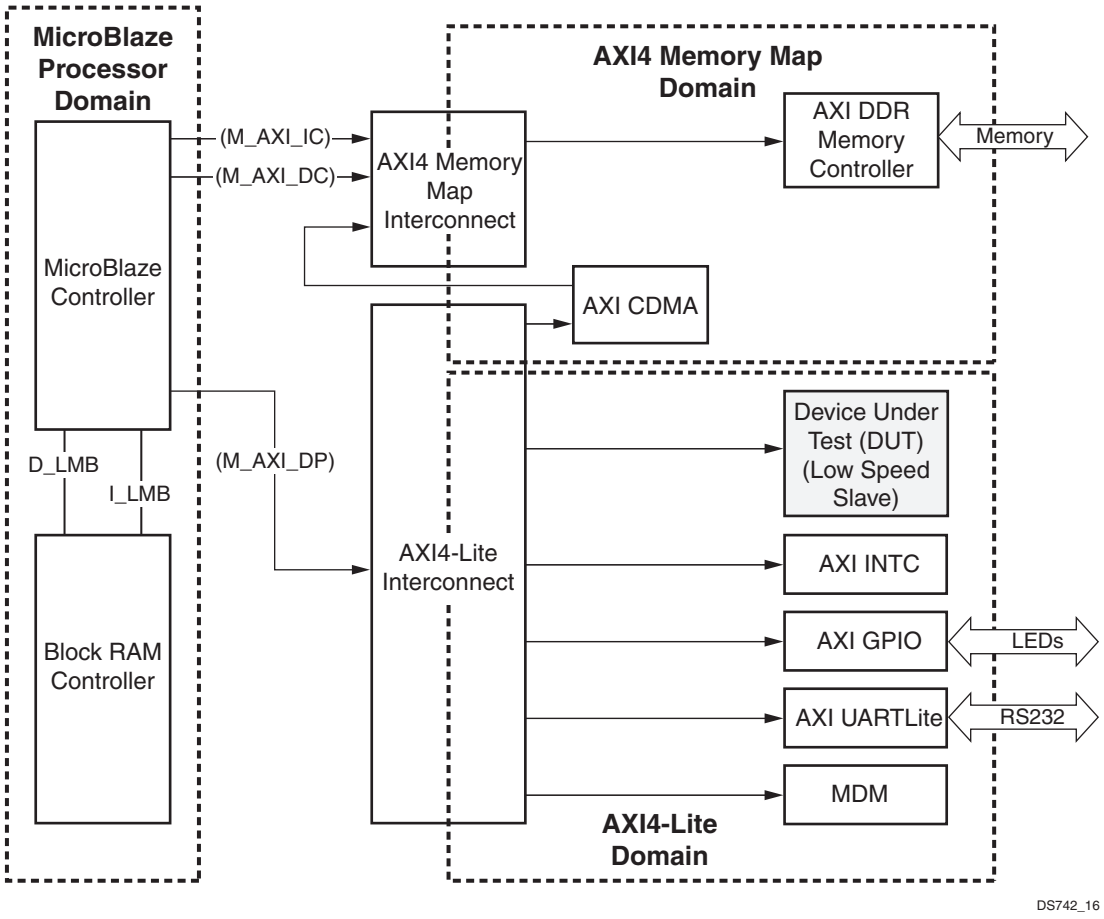

Figure 16: **Virtex-6 and Spartan-6 Devices F_{MAX} Margin System**

Table 21: **AXI SPI System Performance**

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

Notes:

1. S6 LUT utilization ~60%, block RAM utilization ~70%, I/O utilization 80%, MB not on AXI4 interconnect, AXI4 interconnect configured with a single clock of 120MHz.
2. V6 LUT utilization ~70%, block RAM utilization ~70%, I/O utilization ~80%.

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

Specification Exceptions

Exceptions from the Motorola M68HC11-Rev. 4.0 Reference Manual

1. A slave mode-fault error interrupt is added to provide an interrupt if a SPI device is configured as a slave and is selected when not enabled.
2. In this design, the SPIDTR and SPIDRR 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 utility 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. M68HC11 implementation supports only byte transfer. In this design either a byte, half-word or word transfer can be configured via a generic C_NUM_TRANSFER_BITS.
10. The baud rate generator is specified by Motorola to be programmable via bits in the control register; however, in this FPGA design the baud rate generator is programmable via parameters in the VHDL implementation. Therefore, 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. The AXI SPI IP core is tested with Atmel AT45DB161D and ST Microelectronics M25P16 serial SPI slave devices. These devices support SPI modes 0 and 3. These devices have data valid time of 8 ns from the falling edge of SCK. While operating with these devices at higher speed of 50 MHz (most instructions supports this speed), the core should be configured in C_SCK_RATIO = 2 mode (where the AXI is configured to operate at 100 MHz). Due to limited time availability in the design as well as real SPI slave behavior for data change, the data in the SPI core is registered in the middle of each falling edge and the next consecutive rising edge. As per the M68HC11 document, the master should register data on each rising edge of SCK in SPI modes 1 and 3. Note that the data registering mechanism when C_SCK_RATIO =2 follows a different pattern than specified in the standard (this is applicable to the data registering mechanism in the IP core only). The SPI core when configured in master mode changes data on each falling edge and this behavior is as per the M68HC11 standard.
12. When the AXI SPI IP core is configured in slave mode, the data in the core is registered on the SCK rising edge + 1 AXI clock signal. Internally, this data is registered on the next rising edge of AXI. The core changes the data on the SCK falling edge + AXI clock cycle.

List of Acronyms

Acronym	Description
AMBA	Advanced Microcontroller Bus Architecture
ARM	Advanced RISC Machine
AXI	Advanced eXtensible Interface
BRG	Baud Rate Generator
CPHA	Clock Phase
CPOL	Clock Polarity
DDR	Double Data Rate
DGIER	Device Global Interrupt Enable Register
DRR	Data Receive Register
DTR	Data Transmit Register
DUT	Device Under Test
EDK	Embedded Development Kit
EEPROM	Electrically Erasable Programmable Read-Only Memory
FF	Flip-Flop
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
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
MISO	Master In Slave Out
MODF	Mode-fault Error
MOSI	Master Out Slave In
MSB	Most Significant Bit
RAM	Random Access Memory
Rx	Receive
SCK	Serial Clock
SPI	Serial Peripheral Interface
SPICR	SPI Control Register
SPIDRR	SPI Data Receive Register
SPIDTR	SPI Data Transmit Register
SPIE	SPI Interrupt Enable

Acronym	Description *(Cont'd)*
SPISEL	SPI Slave Select Line
SPISR	SPI Status Register
SPISSR	SPI Slave Select Register
SRR	Software Reset Register
$\overline{SS}$(N)	Slave Select
TOW	Toggle On Write
Tx	Transmit
UART	Universal Asynchronous Receiver Transmitter
UCF	User Constraints File
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits)
WCOL	Write Collision error
XPS	Xilinx Platform Studio
XST	Xilinx Synthesis Technology

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 Integrated Software Environment (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, please contact your local Xilinx sales representative.

Reference Documents

To search for Xilinx documentation, go to http://www.xilinx.com/support

The following documents contain reference information important to understanding the AXI SPI IP 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 (v1.01.a) Data Sheet (DS765)
5. Spartan-3AN FPGA In-System Flash User Guide (UG333)
6. AXI Interconnect IP Data Sheet (DS768)

Revision History

Date	Version	Revision
09/21/10	1.0	First release of the core with AXI interface support. The previous release of this document was ds570.
09/21/10	1.0.1	Documentation only. Added inferred parameters text on page 4.
09/28/10	1.1	Updated version and utilization table.
12/14/10	1.2	Updated to v1.01.a version; updated tools to 12.4.
6/22/11	1.3	Updated for 13.2 release; removed Design Constraints section; added 7 series support; modified Timing Constraints section; modified Allowable value for C_SCK_RATIO in Table 1.
7/6/11	1.3.1	Corrected verbiage for LogiCORE Facts table footnote 1.
10/19/11	1.4	Summary of Major Core Version Changes • Updated to v1.02.a version • Updated tools to 13.3 • Fixed CR610995 - SPI slave Select Endianness is corrected Summary of Major Documentation Changes • Updated Notice of Disclaimer • Updated List of Acronyms • In Core Performance section, listed the latest device to earliest device: Artix-7, Virtex-7, Kintex-7, Virtex-6, and Spartan-6 • Added information about IPIC on page 3 • Corrected Figures 14 and 15 • Updated to newest FrameMaker template
01/18/12	1.5	Added Steps to be followed when the core is configured in slave mode, page 25.

Notice of Disclaimer