

Introduction

The Xilinx OPB Serial Peripheral Interface (SPI) connects to the OPB and provides the controller interface to any SPI device such as SPI EEPROMs. It is assumed that the reader is familiar with the SPI EEPROMs and its operation. The OPB SPI is based on the Motorola M68HC11 device. However, there are differences in Xilinx OPB SPI implementation and the M68HC11 specification that should be reviewed, see the [Specification Exceptions](#) section.

Features

The Xilinx OPB SPI is a soft IP core designed for the Xilinx FPGAs and contains following features:

- Supports four signal interface (MOSI, MISO, SCK and \overline{SS})
- Supports slave select (\overline{SS}) bit for each slave on the SPI bus
- Supports full-duplex operation
- Supports master and slave SPI modes
- Supports programable clock phase and polarity
- Optional transmit and receive FIFOs
- Supports continuous transfer mode for automatic scanning of a peripheral
- Supports back-to-back transactions
- Supports automatic or manual slave select modes
- Supports local loopback capability for testing

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	QPro™-R Virtex-II™, QPro Virtex-II, Virtex-II, Virtex-II Pro, Virtex-4, Spartan™-3 and Virtex-5	
Version of Core	opb_spi	v1.00e
Resources Used.		
	Min	Min
Slices	See Table 16 and Table 17	
LUTs		
FFs		
Block RAMs		
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Reference Designs	None	
Design Tool Requirements		
Xilinx Implementation Tools	8.2i or later	
Verification	ModelSim SE 6.0 or later	
Simulation	ModelSim SE 6.0 or later	
Synthesis	XST 8.2i or later	
Support		
Support provided by Xilinx, Inc.		

© 2006 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at <http://www.xilinx.com/legal.htm>. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice. NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

OPB IPIF Module: The OPB IPIF module provides the interface to the OPB and implements OPB bus protocol logic.

Register Module: The Register module includes all memory mapped registers (as shown in [Figure 1](#)). It interfaces to the Xilinx OPB through the OPB IPIF module. It consists of an 8-bit status register, an 8-bit control register, an n-bit slave select register and a pair of 8-bit transmit/receive registers. All registers are accessed directly from the Xilinx OPB which is a subset of IBM's 64-bit OPB.

SPI Module: The SPI module consists of a shift register, a parameterized baud rate generator (BRG) and a control unit. It provides the SPI EEPROM interface, including the control logic state machines, and initialization logic.

Optional FIFOs: The Tx FIFO and Rx FIFO are implemented on both transmit and receive paths when enabled by the parameter C_FIFO_EXIST.

OPB SPI I/O Signals

Table 1 provides a summary of all OPB SPI input/output (I/O) signals, the interfaces under which they are grouped, and the brief description of the signals.

Table 1: OPB SPI I/O signal Descriptions

Port	Signal Name	Interface	I/O	Initial State	Description
SPI Signals					
P1	SCK_I	SPI	I	-	SPI bus clock input
P2	SCK_O	SPI	O	0	SPI bus clock output
P3	SCK_T	SPI	O	1	3-state enable for SPI bus clock
P4	MOSI_I	SPI	I	-	Master output slave input
P5	MOSI_O	SPI	O	1	Master output slave input
P6	MOSI_T	SPI	O	1	3-state enable master output slave input
P7	MISO_I	SPI	I	-	Master input slave output
P8	MISO_O	SPI	O	1	Master input slave output
P9	MISO_T	SPI	O	1	3-state enable master input slave output
P10	SPISEL	SPI	I	1	Local SPI slave select active low input. Must be set to 1
P11	SS_I[0:C_NUM_SS_BITS - 1] ⁽¹⁾	SPI	I	-	Input slave select vector of length n where there are n SPI devices
P12	SS_O[0:C_NUM_SS_BITS - 1]	SPI	O	1	Output one-hot encoded, active low slave select vector of length n
P13	SS_T[0:C_NUM_SS_BITS - 1]	SPI	O	1	3-state enable for slave select vector of length n
System Signals					
P14	Freeze	System	I	-	Brings OPB SPI to freeze state
OPB Slave Signals					
P15	IP2INTC_Irpt	OPB	O	0	System interrupt

Table 1: OPB SPI I/O signal Descriptions (Contd)

Port	Signal Name	Interface	I/O	Initial State	Description
P16	OPB_Select	OPB	I	-	OPB select
P17	OPB_RNW	OPB	I	-	OPB read not write
P18	OPB_ABus[0:C_OPB_AWIDTH - 1]	OPB	I	-	OPB address bus
P19	OPB_DBus[0:C_OPB_DWIDTH - 1]	OPB	I	-	OPB data bus
P20	OPB_BE[0:C_OPB_DWIDTH/8 - 1]	OPB	I	-	OPB byte enables
P21	OPB_seqAddr	OPB	I	-	OPB sequential address
P22	SPI_xferAck	OPB	O	0	OPB SPI transfer acknowledge
P23	SPI_errAck	OPB	O	0	OPB SPI error acknowledge
P24	SPI_toutSup	OPB	O	0	OPB SPI time-out suppress
P25	SPI_retry	OPB	O	0	OPB SPI retry
P26	SPI_DBus[0:C_OPB_DWIDTH-1]	OPB	O	0	OPB SPI slave data bus
P27	OPB_Clk	OPB	I	-	OPB clock
P28	OPB_Rst	OPB	I	-	OPB reset

Notes:
 1. Signal is not used in this design.

OPB SPI Design Parameters

To allow the users to obtain an OPB SPI that is uniquely tailored for their system, certain features are the parameterizable in the OPB SPI design. This allows the user to have a design that only utilizes the resources required by their system and gives the best possible performance. The features that are parameterizable in the OPB SPI are shown in Table 2.

Table 2: OPB SPI Design Parameters

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
SPI Features					
G1	Both receive and transmit FIFOs	C_FIFO_EXIST	0 = FIFOs not included 1 = FIFOs included	1	integer
G2	OPB to SPI clock frequency ratio	C_OPB_SCK_RATIO ⁽¹⁾	2, 4, 16, 32, Nx16 for N = 1, 2, 3,...,128	32	integer
G3	Slave-only mode	C_SPI_SLAVE_ONLY	0 = Disable slave-only mode 1 = Not supported	0	integer
G4	Number of off-chip slave select bits	C_NUM_OFFCHIP_SS_BITS	0 - 32 ⁽²⁾	0	integer

Table 2: OPB SPI Design Parameters (Contd)

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G5	Total number of slave select bits	C_NUM_SS_BITS	1 - 32	1	integer
G6	Depth of transmit and receive FIFOs	C_DEPTH	16	16	integer
G7	Width of SPI registers	C_NUM_BITS_REG	8	8	integer
G8	Width of SPI shift register	C_NUM_BITS_SR	8	8	integer
G9	Length of SPI FIFO occupancy vectors	C_OCCUPANCY_NUM_BITS	4	4	integer
G10	Target FPGA family	C_FAMILY	qrvirtex2, qvirtex2, virtex2, virtex2p, virtex4, spartan3, virtex5	virtex2	string
System					
G11	Device block ID	C_DEV_BLK_ID	Note ⁽³⁾	4	integer
G12	Module identification register enable	C_DEV_MIR_ENABLE	Note ⁽³⁾	0	integer
Interrupt Interface					
G13	Enable/Disable interrupt support	C_INTERRUPT_PRESENT	0 = Disable interrupt 1 = Enable interrupt	1	integer
OPB Interface					
G14	OPB address bus width	C_OPB_AWIDTH	32	32	integer
G15	OPB data bus width	C_OPB_DWIDTH	32	32	integer
Address Space					
G16	Base address for OPB SPI	C_BASEADDR	Valid address range ⁽⁴⁾	None	std_logic_vector
G17	High address for OPB SPI	C_HIGHADDR	Valid address range ^(4, 5)	None	std_logic_vector
G18	Offset parameter to OPB IPIF	C_IP_REG_BAR_OFFSET	x"00000060"	x"00000060"	std_logic_vector
Notes:					
1. Ratios of 2 and 4 are not supported in this release of the core.					
2. Must be <= to C_NUM_SS_BITS..					
3. See the <i>Processor IP Reference Guide</i> under Part 1: Embedded Processor IP, under IPIF, under OPB IPIF Architecture.					
4. The range specified by C_BASEADDR and C_HIGHADDR must comprise of complete, contiguous power of two range such that range = 2 ^m , and the m least significant bits of C_BASEADDR must be zero.					
5. This value should be = (C_BASEADDR + any 2 ^m -1) and must be >= x7F.					

Parameter-Port Dependencies

The dependencies between the OPB SPI design parameters and I/O signals are shown in Table 3. It gives information about how ports and parameters get affected by changing certain parameters. When

certain features are parameterized away, the related logic will be part of design, signals are unused and the related output signals are set to default values.

Table 3: Parameter-Port Dependencies

Generic or Port	Parameter	Affects	Dependencies	Description
Design Parameters				
G3	C_SPI_SLAVE_ONLY	P1, P2, P3, P5	-	Slave only mode not supported in the current implementation
G4	C_NUM_OFFCHIP_SS_BITS	-	G5	Defines the number of slave select bits that go off-chip (i.e. off-chip SPI slave devices). Must be less than or equal to C_NUM_SS_BITS
G5	C_NUM_SS_BITS	P11, P12, P13	G15	Defines the total number of slave select bits
G13	C_INTERRUPT_PRESENT	P15	-	Utilized only when interrupt mode is used
G14	C_OPB_AWIDTH	P18	-	OPB address bus width
G15	C_OPB_DWIDTH	P19, P20, P26	-	OPB data bus width
I/O Signals				
P11	SS_I[0:C_NUM_SS_BITS - 1]	-	G5	The number of SS_I pins are generated based on C_NUM_SS_BITS
P12	SS_O[0:C_NUM_SS_BITS - 1]	-	G5	The number of SS_O pins are generated based on C_NUM_SS_BITS
P13	SS_T[0:C_NUM_SS_BITS - 1]	-	G5	The number of SS_T pins are generated based on C_NUM_SS_BITS
P15	IP2INTC_Irpt	-	G13	Interrupt output signal is used when C_INTERRUPT_PRESENT = 1

OPB SPI Register Descriptions

The OPB SPI contains addressable registers as shown in Table 4. The base address is set by the parameter C_BASEADDR and all SPI register addresses are calculated by an offset from C_BASEADDR.

The IPIF also contains the Interrupt registers and software reset register which are both optional registers. The addresses of both register sets are calculated by an offset from C_BASEADDR. In this document, bit assignment will be made assuming a 32-bit OPB interface. The highest bit index is the LSB. Assignment for either wider or narrower buses follows the same convention.

Table 4 shows addresses of all the SPI registers, IPIF interrupt registers and the IPIF software reset register. The transmit FIFO occupancy register and the receive FIFO occupancy register only exist when the OPB SPI is configured via parameters to include FIFOs. Note that the transmit and receive

registers have independent addresses which differs from the M68HC11 specification where the same address is to be assigned to the two independent registers.

Table 4: OPB SPI Register Summary

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
SPI Core Grouping				
C_BASEADDR + 40	IPIF Software Reset Register ⁽¹⁾	Write	N/A	IPIF software reset
C_BASEADDR + 60	SPICR	R/W	00000180	SPI control register
C_BASEADDR + 64	SPISR	Read	00000005	SPI status register
C_BASEADDR + 68	SPIDTR	Write	00000000	SPI data transmit register A single register or a FIFO
C_BASEADDR + 6C	SPIDRR	Read	00000000	SPI data receive register A single register or a FIFO
C_BASEADDR + 70	SPISSR	R/W	Note ⁽²⁾	SPI slave select register
C_BASEADDR + 74	SPI Transmit FIFO Occupancy Register ⁽³⁾	Read	00000000	Transmit FIFO occupancy register
C_BASEADDR + 78	SPI Receive FIFO Occupancy Register ⁽³⁾	Read	00000000	Receive FIFO occupancy register
OPB IPIF ISC Grouping				
C_BASEADDR + 1C	DGIE	R/W	00000000	Device global interrupt enable register
C_BASEADDR + 20	IPIISR	R/TOW ⁽⁴⁾	00000000	IP interrupt status register
C_BASEADDR + 28	IPIER	R/W	00000000	IP interrupt enable register
Notes:				
1. See the Processor IP Reference Guide under Part1: Embedded Processor IP, under OPB IPIF Architecture, under OPB IPIF Register Descriptions for a complete description of these registers.				
2. The number of 1 = number of slaves present. This default value is set depending on the parameter C_NUM_SS_BITS.				
3. This register does not exist if C_FIFO_EXIST = 0.				
4. TOW = Toggle On Write. Writing a 1 to a bit position within the register causes the corresponding bit position in the register to toggle.				

IPIF Software Reset Register

The IPIF software reset register is instantiated in the OPB SPI IPIF module. This reset module permits software reset of the SPI module independent of other modules in the system and has another register for test purposes. Figure 2 shows the bit assignment for this register.

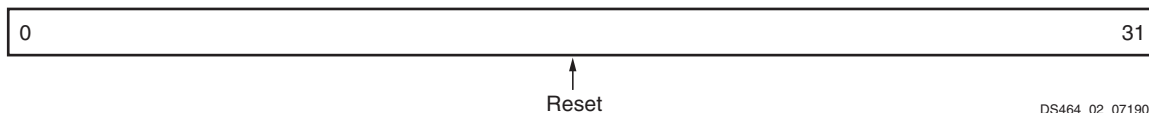


Figure 2: IPIF Software Reset Register

DS464_02_071906

SPI Control Register (SPICR)

Bit assignment in the SPICR is shown in Figure 3 and described in Table 5. Bit assignment was made to follow the assignment pattern of Xilinx OPB IPIF specifications and when possible, follow the M68HC11 assignment pattern.

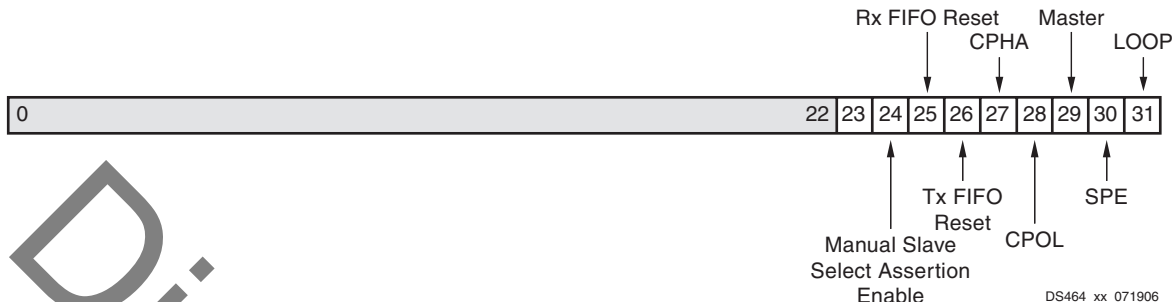


Figure 3: SPI Control Register

Table 5: SPICR Description

Bit(s)	Name	Core Access	Reset Value	Description
0-22	Reserved			
23	Master Transaction Inhibit	R/W	'1'	Master Transaction Inhibit. This bit inhibits master transactions in the same way freeze functions. This bit has no effect on slave operation. '0' = Master transactions enabled '1' = Master transactions disabled
24	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 state machine '1' = Slave select output follows data in slave select register
25	Rx FIFO Reset	R/W	'0'	Receive FIFO Reset. This bit forces a reset of the FIFO pointer and asserts the FIFO empty flag. FIFO contents are unchanged. One OPB clock cycle after reset, this bit is again set to '1'. This bit is unassigned when the OPB SPI is not configured with FIFOs. '0' = Receive FIFO normal operation '1' = Reset receive FIFO pointer
26	Tx FIFO Reset	R/W	'0'	Transmit FIFO Reset. This bit forces a reset of the FIFO pointer and asserts the FIFO empty flag. FIFO contents are unchanged. One OPB clock cycle after reset, this bit is again set to '1'. This bit is unassigned when the OPB SPI is not configured with FIFOs. '0' = Transmit FIFO normal operation '1' = Reset transmit FIFO pointer
27	CPHA	R/W	'0'	Clock Phase. Setting this bit selects one of two fundamentally different transfer formats. See timing diagrams and discussion of diagrams.

Table 5: SPICR Description (Contd)

Bit(s)	Name	Core Access	Reset Value	Description
28	CPOL	R/W	'0'	Clock Polarity. Setting this bit defines clock polarity. '0' = Active high clock; SCK idles low '1' = Active low clock; SCK idles high
29	MASTER	R/W ⁽¹⁾	'0'	Master. Setting this bit configures the SPI device as a master or a slave. '0' = Slave configuration '1' = Master configuration
30	SPE	R/W	'0'	SPI System Enable. Setting this bit high enables the SPI devices as noted below. '0' = SPI system disabled. Both master and slave outputs are in "3-state" and slave inputs ignored '1' = SPI system enabled. Master outputs active (e.g. MOSI and SCK in idle state) and slave outputs will become active if SS becomes asserted. Master will start transfer when transmit data is available
31	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 Note that the interrupt enable bit which resides at this bit position of the M68HC11 specification resides in the interrupt enable register in this implementation; see Specification Exceptions
Notes: 1. Will be read-only if slave-only option is implemented.				

SPI Status Register (SPISR)

The bit assignment in the SPISR is shown in Figure 4 and described in Table 6. Bit assignment was made to follow the assignment pattern of Xilinx OPB IPIF specifications and when possible, follow the M68HC11 assignment pattern. The SPISR is a read-only register.

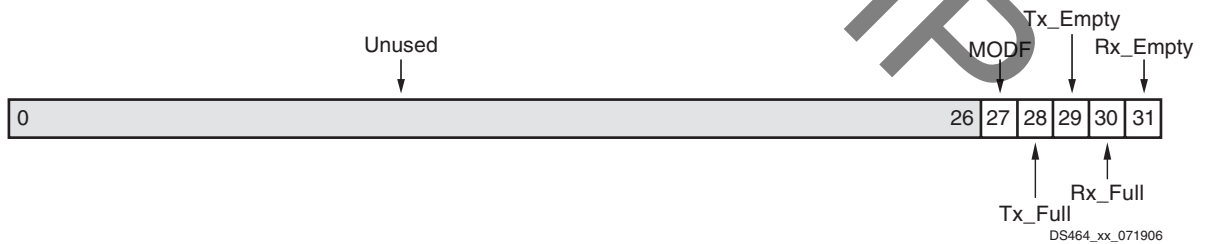


Figure 4: SPI Status Register

Table 6: SPISR Description

Bit(s)	Name	Core Access	Reset Value	Description
0-26	Reserved			
27	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. MODF does generate an interrupt with a single cycle strobe when the MODF bit transitions from a low to high. '0' = No error '1' = Error condition detected
28	Tx_Full	Read	'0'	Transmit Full. When a transmit FIFO exists, this bit will be set high when the transmit FIFO is full. When FIFOs don't exist, this bit is set high when an OPB write to the register has been made. This bit is cleared when the SPI transfer is completed.
29	Tx_Empty	Read	'1'	Transmit Empty. When a transmit FIFO exists, this bit will be set high when the transmit FIFO is empty. The occupancy of the FIFO is decremented with the completion of each SPI transfer. When FIFOs don't exist, this bit is cleared with the completion of an SPI transfer. Either with or without FIFOs, this bit is cleared upon an OPB write to the FIFO or transmit register.
30	Rx_Full	Read	'0'	Receive Full. When a receive FIFO exists, this bit will be set high when the receive FIFO is full. The occupancy of the FIFO is incremented with the completion of each SPI transaction. When FIFOs don't exist, this bit is set high when an SPI transfer has completed. Rx_Empty and Rx_Full are complements in this case.
31	Rx_Empty	Read	'1'	Receive Empty. When a receive FIFO exists, this bit will be set high when the receive FIFO is empty. The occupancy of the FIFO is decremented with each FIFO read operation. When FIFOs don't 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 a write only register and contains data to be transmitted on the SPI bus. It is double buffered with the shift register. Once the enable bit is set high 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 7 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.

The hardware that forwards data from the register or FIFO to the shift register will never cause a write collision error. Attempting to write in a full register or FIFO will not result in a write acknowledgement for the OPB transaction, but rather an OPB time-out.

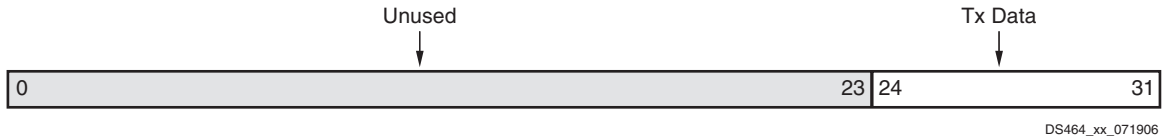


Figure 5: SPI Data Transmit Register

Table 7: SPI Data Transmit Register Bit Definitions (Bit assignment assumes 32-bit bus)

Bit(s)	Name	Core Access	Reset Value	Description
0-23				Reserved
24-31	Tx Data D0 - D7	Write only	0	SPI transmit data.

SPI Data Receive Register (SPIDRR)

This double buffer receive register contains the data received from the SPI bus, 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 (i.e. is full), then the most recently transferred data will be lost and a receive over-run interrupt will occur. 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.

If an SPI transfer occurs with the FIFO full, then the most recently transferred data will be lost and a receive over-run interrupt will occur. The receive FIFO is a read only buffer. If an attempt to read an empty receive register or FIFO is made, then an OPB time-out error will occur because an acknowledgement will not be issued. The SPIDRR is shown in Figure 6, while the specifics of the data format is described in Table 8.



Figure 6: SPI Data Receive Register

Table 8: SPI Data Receive Register Bit Definitions (Bit assignment assumes 32-bit bus)

Bit(s)	Name	Core Access	Reset Value	Description
0-23				Reserved
24-31	Rx Data D0 - D7	Read only	0	SPI receive data.

SPI Slave Select Register (SPISSR)

This field contains an n-length vector specifying the slave with whom the local master will communicate. This vector is an active-low, one-hot encoded vector ($\overline{SS}(n)$, where n is the index number). Actual assignment of slaves to specific bits is performed by Xilinx Platform Generator. This

register is a read/write register and is shown in **Figure 7**. The specifics of the data format is described in **Table 9**.

The index of $\overline{SS}(n)$ increments in the opposite direction to that of the OPB bit index. OPB bit index 31 is bit(0) of $\overline{SS}(n)$, OPB bit index 30 is bit(1) of $\overline{SS}(n)$ and so on.

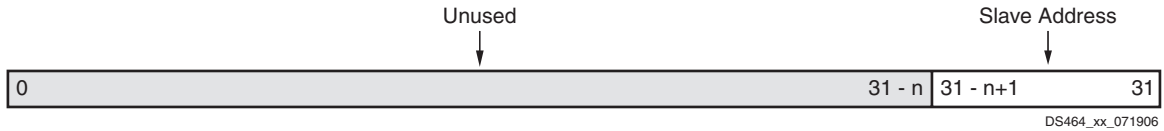


Figure 7: SPI Slave Select Register

Table 9: SPI Slave Select Address Register Bit Definitions (Bit assignment assumes 32-bit bus)

Bit(s)	Name	Core Access	Reset Value	Description
0 to (31-n)				Reserved
(31-n+1) to 31	Slave Address	R/W	1	Active-low, one-hot encoded slave select vector of length n-bits. n must be less than or equal to the databus width. Note that $\overline{SS}(n)$ increments in the opposite direction to that of the OPB bit index.

SPI Transmit FIFO Occupancy Register (Tx_FIFO_OCY)

When the OPB SPI is configured with FIFOs, this field contains the occupancy number greater than one for the transmit FIFO. The actual occupancy is the binary value plus 1. This register is a read only register and does not exist when the OPB SPI is configured without FIFOs i.e. C_FIFO_EXIST = 0.

The transmit FIFO empty interrupt or status bit is the only reliable way to determine if the FIFO is empty; reading this register cannot be used to determine if the FIFO is empty. An attempt to write to this register does not yield an acknowledgement, but rather an OPB time-out. The transmit FIFO occupancy register is shown in **Figure 8**, while the specifics of the data format is described in **Table 10**.

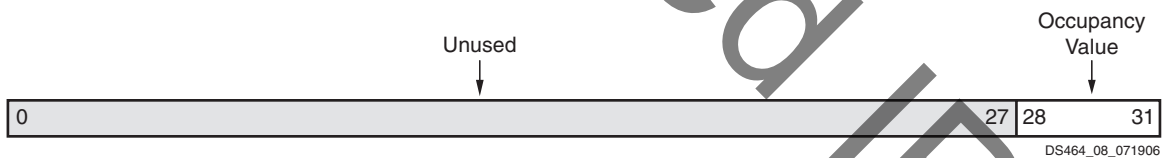


Figure 8: SPI Transmit FIFO Occupancy Register

Table 10: Transmit FIFO Occupancy Register Bit Definitions (Bit assignment assumes 32-bit bus)

Bit(s)	Name	Core Access	Reset Value	Description
0-27				Reserved
28-31	Occupancy Value	Read	0	Bit 28 is the MSB. The binary value plus 1 yields the occupancy.

SPI Receive FIFO Occupancy Register (Rx_FIFO_OCY)

This field contains the occupancy number greater than one for the receive FIFO when the OPB SPI is configured with FIFOs. The actual occupancy is the binary value plus 1. This register is a read only register and does not exist when the OPB SPI is configured without FIFOs i.e. C_FIFO_EXIST = 0.

The receive FIFO empty status bit is the only reliable way to determine if the FIFO is empty, reading this register cannot be used to determine if the FIFO is empty. An attempt to write to this register does not yield an acknowledgement, but rather an OPB time-out. The receive FIFO occupancy register is shown in Figure 9, while the specifics of the data format is described in Table 11.



Figure 9: SPI Receive FIFO Occupancy Register

Table 11: Receive FIFO Occupancy Register Bit Definitions (Bit assignment assumes 32-bit bus)

Bit(s)	Name	Core Access	Reset Value	Description
0-27				Reserved
28-31	Occupancy Value	Read	0	Bit 28 is the MSB. The binary value plus 1 yields the occupancy.

OPB SPI Interrupt Descriptions

The OPB SPI module has seven distinct interrupts that are sent to the IPIF. The IPIF utilizes the IP interrupt service controller (IPISC) and allows each interrupt to be enabled independently (via the IP interrupt enable register (IPIER)).

The interrupt registers are in the interrupt module which is instantiated in the OPB SPI IPIF module. The OPB SPI has the option to include the interrupt module, which permits multiple conditions for an interrupt, or not to include the interrupt module and have an interrupt strobe occur only after the completion of a transfer. All status register bits are available for detailed information independent of the interrupt choice.

The OPB SPI permits the exclusion of the interrupt module by setting the parameter C_INTERRUPT_PRESENT = 0. If the interrupt module is not implemented, then the single interrupt condition occurs when the receive register or FIFO is full. At this time, Xilinx does not provide software for utilizing the single interrupt scheme.

Device Global Interrupt Enable Register (DGIE)

The Device Global Interrupt Enable register is used to globally enable the final interrupt output from the IPIF interrupt service as shown in Figure 10 and described in Table 12. This bit is essentially AND'ed with the input to the interrupt controller. This bit is a read/write bit and is cleared upon reset.

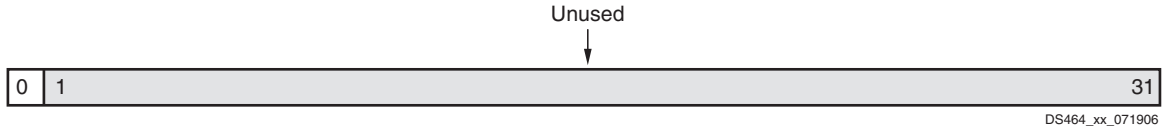


Figure 10: Device Global Interrupt Enable (DGIE) Register

Table 12: DGIE Register Description (Bit assignment assumes 32-bit bus)

Bit(s)	Name	Access	Reset Value	Description
0	GIE	R/W	'0'	Global Interrupt Enable. It enables all individually enabled interrupts to be passed to the interrupt controller. '0' = Disabled '1' = Enabled
1-31	Reserved			

IP Interrupt Status Register (IPISR)

When the interrupt module is included, up to seven unique interrupt conditions are possible depending upon whether the system is configured with FIFOs or not. A system without FIFOs has six interrupts.

The IPIF interrupt module has a register that can enable each interrupt independently. Bit assignment in the Interrupt register for a 32-bit data bus is shown in Figure 11 and described in Table 13. The interrupt register is a read/toggle on write register and by writing a '1' to a bit position within the register causes the corresponding bit position in the register to 'toggle'. All register bits are cleared upon reset.

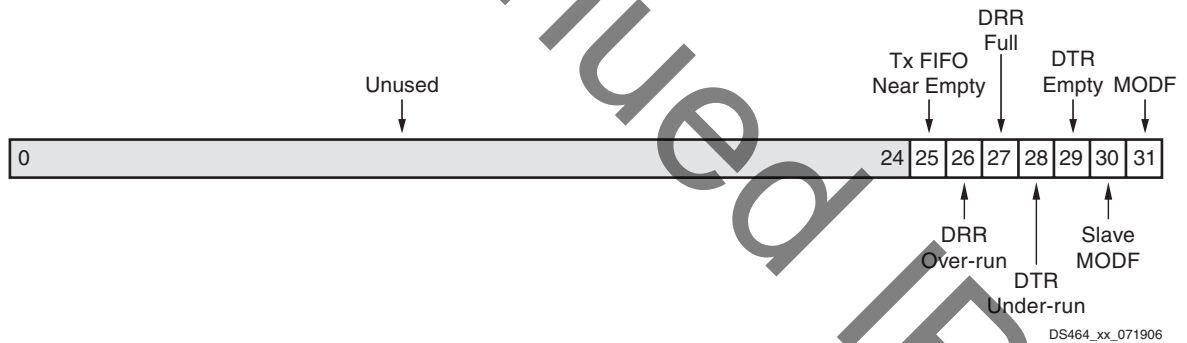


Figure 11: IP Interrupt Status Register (IPISR)

Table 13: IPISR Description (Bit assignment assumes 32-bit bus)

Bit(s)	Name	Access	Reset Value	Description
0-24	Reserved			

Table 13: IPISR Description (Bit assignment assumes 32-bit bus) (Contd)

Bit(s)	Name	Access	Reset Value	Description
25	Tx FIFO Near Empty	R/TOW ⁽¹⁾	'0'	Transmit FIFO Less than or Equal to Half Empty. IPISR bit(25) is the transmit FIFO near 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". Note that "0111" means there are 8 characters in the FIFO to be transmitted. This interrupt exists only if the OPB SPI is configured with FIFOs.
26	DRR Over-run	R/TOW ⁽¹⁾	'0'	Data Receive Register/FIFO Over-run. IPISR bit(26) is the data receive FIFO over-run 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 state machine in order to complete an SPI transfer. This can occur when the SPI device is in either master or slave mode.
27	DRR Full	R/TOW ⁽¹⁾	'0'	Data Receive Register/FIFO Full. IPISR bit(27) is the data receive register full interrupt. Without FIFOs, this bit is set at the end of an SPI byte transfer by a one-clock period strobe to the interrupt register. With FIFOs, this bit is set at the end of the SPI byte transfer when the receive FIFO has been filled by a one-clock period strobe to the interrupt register.
28	DTR Under-run	R/TOW ⁽¹⁾	'0'	Data Transmit Register/FIFO Under-run. IPISR bit(28) is the data transmit register/FIFO under-run interrupt. This bit is set by a one-clock period strobe to the interrupt register when data is request from an "empty" transmit register/FIFO by the SPI state machine in order to perform an SPI transfer. This can occur only when the SPI device is in slave mode. All zeros are loaded in the shift register and transmitted by the slave in an under-run condition.
29	DTR Empty	R/TOW ⁽¹⁾	'0'	Data Transmit Register/FIFO Empty. IPISR bit(29) is the data transmit register/FIFO empty interrupt. Without FIFOs, this bit is set at the end of an SPI byte transfer by a one-clock period strobe to the interrupt register. With FIFOs, this bit is set at the end of the SPI byte 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 SPI transfer complete flag (SPIF) interrupt bit.
30	Slave MODF	R/TOW ⁽¹⁾	'0'	Slave Mode-Fault Error. IPISR bit(30) is the slave mode-fault error flag. This interrupt is generated if the \overline{SS} signal goes active while the SPI device is configured as a slave but is not enabled. This bit is set immediately upon \overline{SS} going active and continually set if \overline{SS} is active and the device is not enabled.
31	MODF	R/TOW ⁽¹⁾	'0'	Mode-Fault Error. IPISR bit(31) 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 IPIER has an enable bit for each defined bit of the IPISR as shown in Figure 12 and described in Table 14. All bits are cleared upon reset.

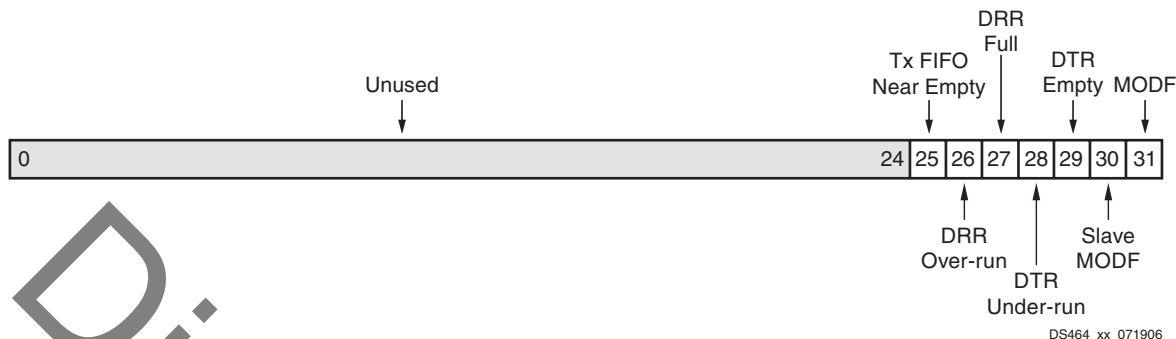


Figure 12: IP Interrupt Enable Register (IPIER)

Table 14: IPIER Description (Bit assignment assumes 32-bit bus)

Bit(s)	Name	Access	Reset Value	Description
0-24	Reserved			
25	Tx FIFO Near Empty enable	R/W	'0'	Transmit FIFO Less Than or Equal to Half Empty Enable. '0' = Disabled '1' = Enabled
26	DRR Over-run enable	R/W	'0'	Receive FIFO Over-run Enable. '0' = Disabled '1' = Enabled
27	DRR Full enable	R/W	'0'	Data Receive Register/FIFO Full Enable. '0' = Disabled '1' = Enabled
28	DTR Under-run enable	R/W	'0'	Data Transmit FIFO Under-run Enable. '0' = Disabled '1' = Enabled
29	DTR Empty enable	R/W	'0'	Data Transmit Register (FIFO) Empty Enable. '0' = Disabled '1' = Enabled
30	Slave MODF enable	R/W	'0'	Slave Mode-Fault Error Flag Enable. '0' = Disabled '1' = Enabled
31	MODF set enable	R/W	'0'	Mode-Fault Error Flag Enable. '0' = Disabled '1' = Enabled

OPB SPI Design Description

SPI Device Features

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

- Three signal in/out (in, out, 3-state) for implementing 3-state SPI device in/out to support multi-master configuration within the FPGA.
- 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. In addition, an optional operation where the \overline{SS} output is toggled automatically with each 8-bit character transfer by the master device. This 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).
- Multi-slave environment supported (automatic generation of additional master slave select signals).
- Supports maximum SPI clock rates up to one-half the OPB clock rate in both master and slave modes when both SPI devices are in the same FPGA part (routing constraints of the SPI bus signals must be incorporated in the MAP/PAR process). In anticipation of remote master operation, slaves operation supports one-fourth the OPB clock rate (artifact of asynchronous SCK clock relative to the OPB clock which requires clock synchronization).
- Parameterizable baud rate generator.
- External ports (selected via a parameter) for off-chip slave interconnects (off-chip masters not supported).
- In the M68HC11 implementation, the transmit register is transparent to the shift register and the receive register is double buffered with the shift register. In this implementation without FIFOs, both the transmit register (SPIDTR) and receive register (SPIDRR) are double buffered.
- The WCOL flag is not supported as a write collision error as described in the M68HC11 reference manual. This can not occur in the Xilinx OPB SPI implementation because the hardware is designed to prevent the data transfer from the transmit buffer to the shift register when an SPI transfer is in progress.

SPI Protocol with Automatic Slave Select Assertion

The SPI protocol is designed to have automatic slaves select assertion and manual slave select assertion which are described in the following sections.

This section describes the SPI protocol where slave select ($\overline{SS}(n)$) is asserted automatically by the SPI master device (i.e SPICR bit(24) = '0'). 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.

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 will

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

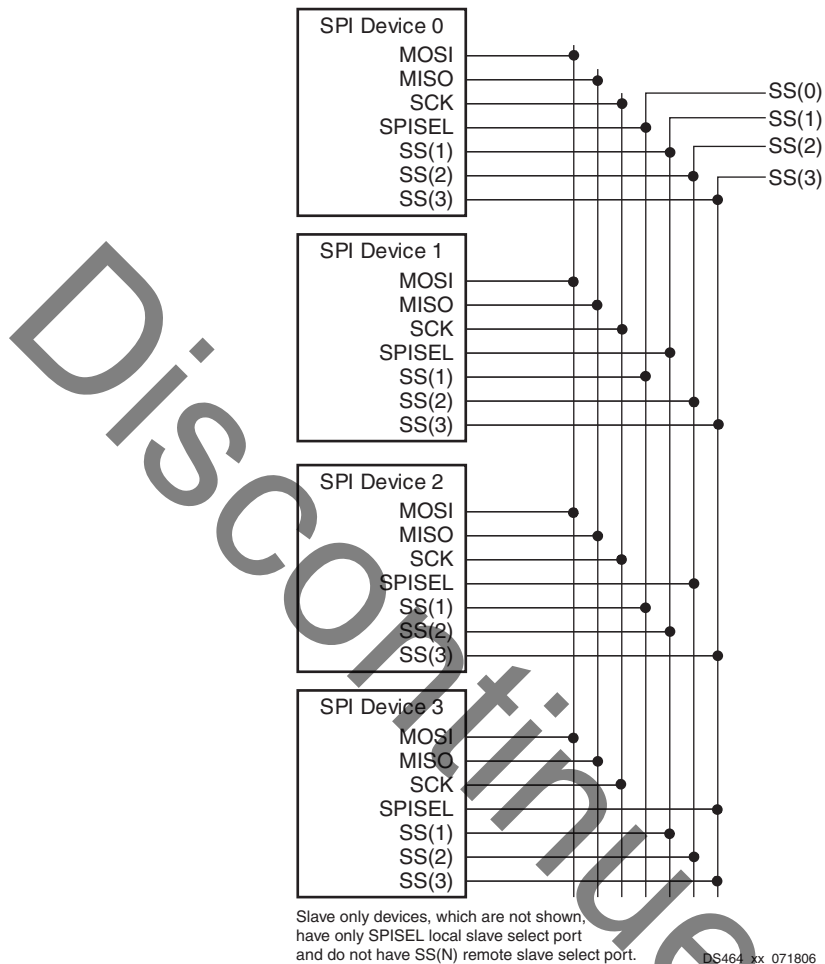


Figure 13: Multi-Master Configuration Block Diagram

The SCK signal is driven by the SPI master controlling the bus and regulates the flow of data. The master must be configured at the time of system configuring to transmit data at 2, 4, 16, 32, ..., 1280, 2048 baud rates.

The M68HC11 SPI specification recommends baud rate selection via bits in the control register; however, in this FPGA implementation, the baud rate is selected via a parameter that fixes the baud rate at the time of system configuration. The FPGA permits reconfiguration by resetting the parameters and rebuilding the system. This approach was adopted to reduce FPGA resource requirements.

Figure 14 shows the timing diagram for an SPI data write cycle and **Figure 15** shows the timing diagram for an SPI data read cycle when CPHA = '0'. The waveforms are shown for CPOL = '0' and the value of generic C_OPB_SCK_RATIO = 4. 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. Since assertion of the $\overline{SS}(n)$ line denotes the start of a transfer, it must be de-asserted and re-asserted for sequential byte transfers to the same slave device.

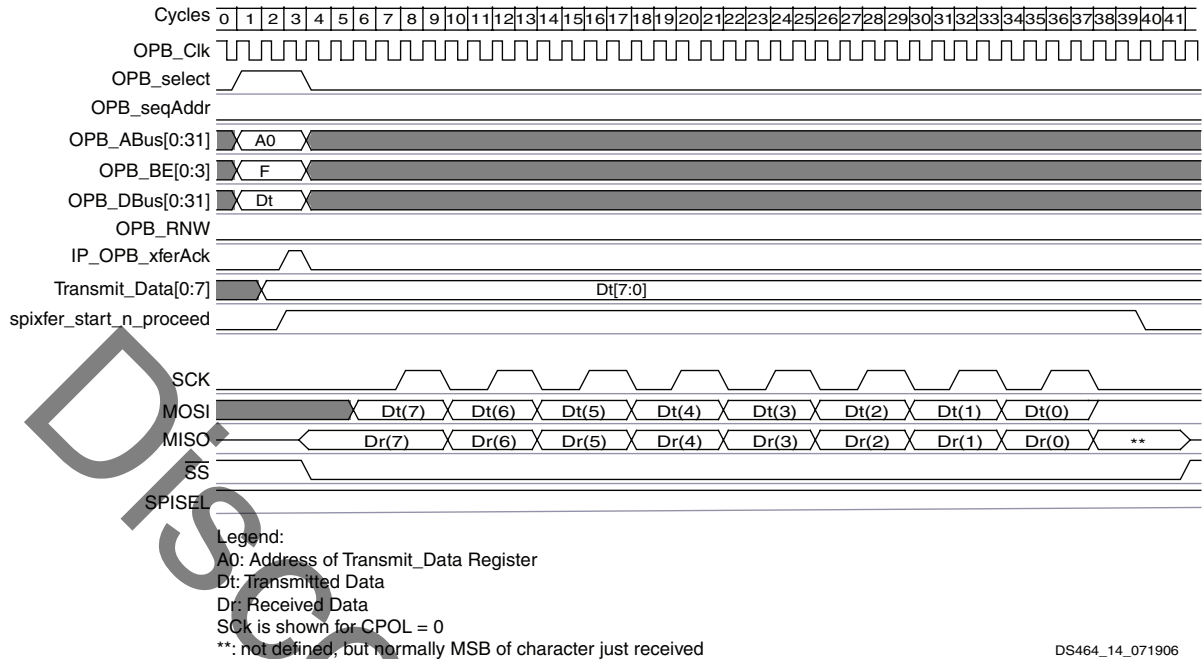


Figure 14: Data Write Cycle on SPI Bus with CPHA = 0 and CR(24) = 0 for 8-bit Character

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 oriented data transfer is possible via bits in the SPICR.

The clock phase and polarity (or idle state) can be modified for SPI data transfers with programmable bits in the SPICR. The clock polarity (CPOL) bit selects an active high (i.e. the clock's idle state = low) or active low clock (i.e. the clock's idle state = high).

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.

Determination of whether the edge of interest is rising or falling edge depends on the idle state of the clock (i.e. CPOL setting). For successful transfers the clock phase and polarity must be identical for the master SPI device and the selected slave device.

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 may be only one master and one slave transmitting data at any given time. The bus architecture provides limited contention error detection (i.e. 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.

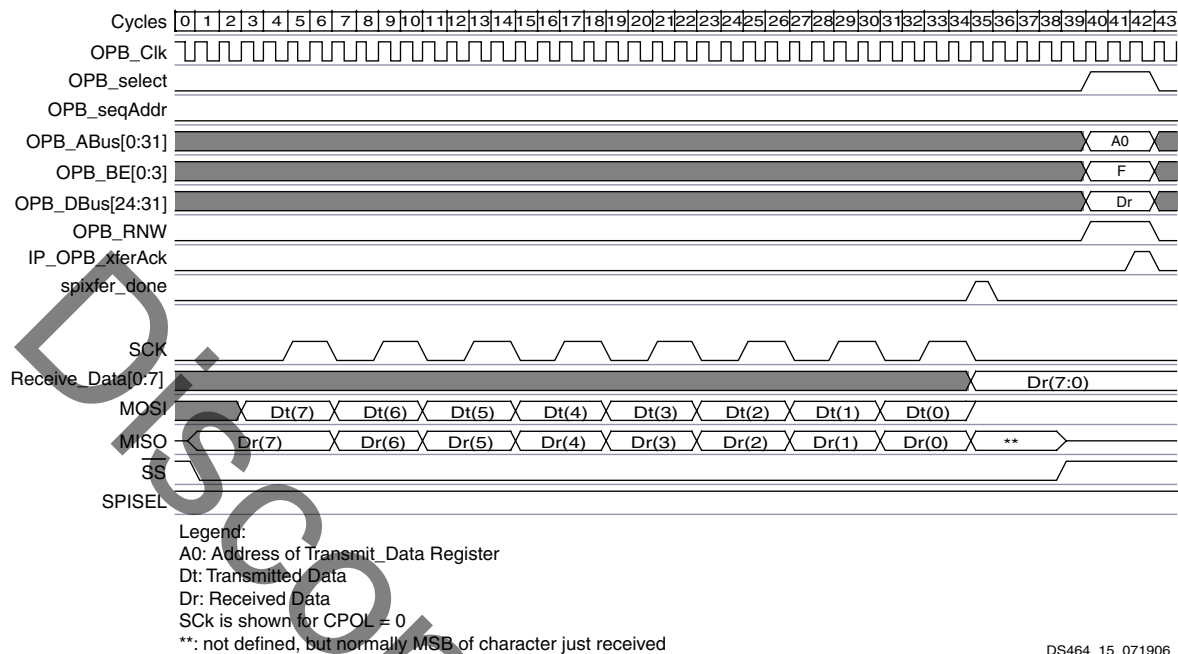


Figure 15: Data Read Cycle on SPI Bus with CPHA = 0 and CR(24) = 0 for 8-bit Character

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 (i.e. \overline{SS} pin high) drive their MISO pin to tri-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 insure that 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.

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 16. The timing diagram for an SPI data write cycle is shown in Figure 16. The timing diagram for an SPI data read cycle when CPHA = '1' is shown in Figure 17. The waveforms are shown for CPOL = '0' and the value of generic C_OPB_SCK_RATIO = 4.

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 $\overline{SS}(n)$ line may 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 insure that the data is properly loaded in the SPISSR register prior to the first SCK edge.

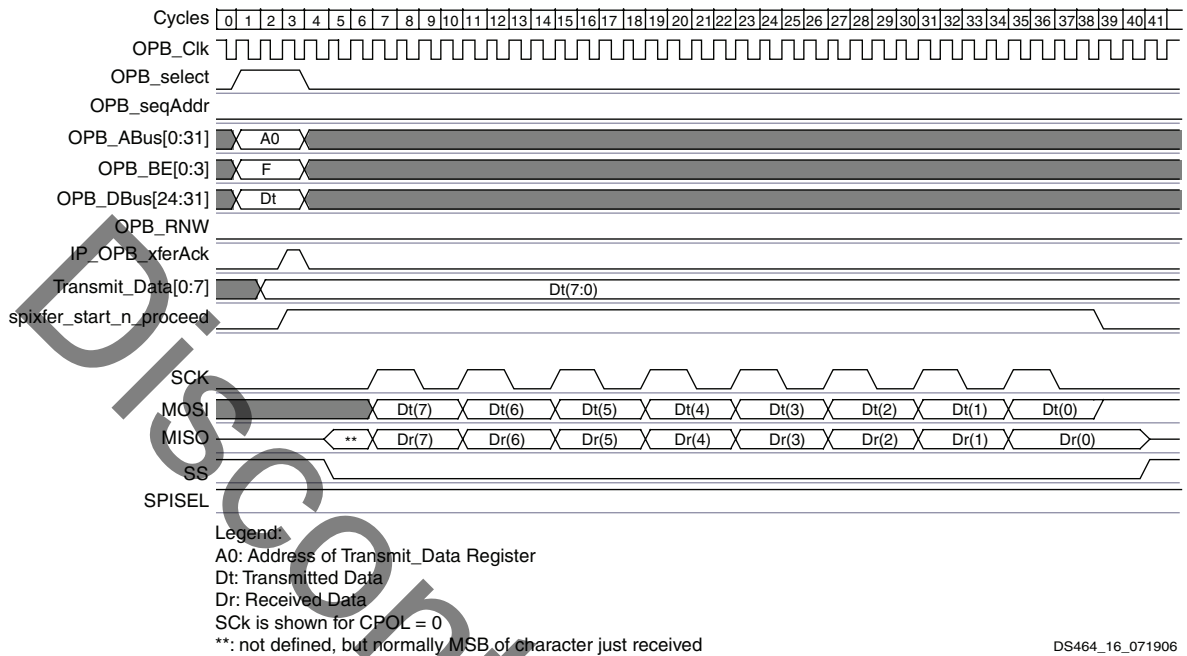


Figure 16: Data Write Cycle on SPI Bus with CPHA = 1 and CR(24) = 0 for 8-bit Character

The OPB SPI implementation double buffers the SPIDTR register with the shift register. Therefore additional logic would be required to monitor the SPIDTR register and asynchronously load the shift register. This is required prior to the first shift whenever a write to the SPIDTR register is performed. In this implementation, 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.

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.

The SPI specification details the timing and waveforms for byte transfers where the MSB is shifted out first on the SPI bus. All data written to the SPIDTR register will be transmitted on the SPI bus, and all data received on the SPI bus will be stored in a SPIDRR register for the user logic to interpret.

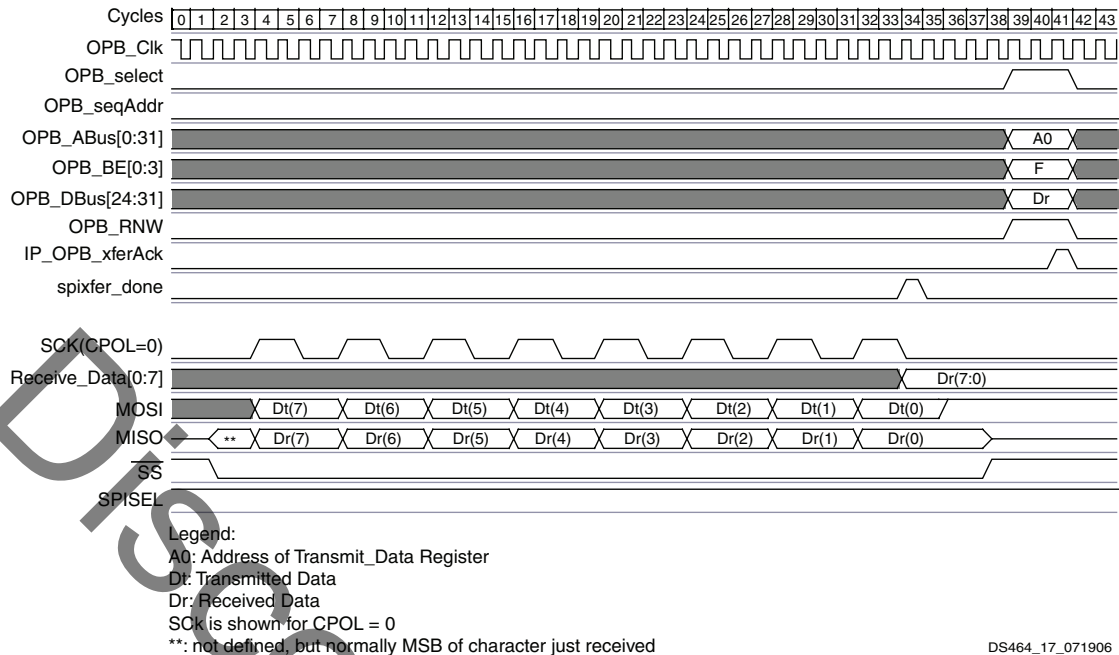


Figure 17: Data Read Cycle on SPI Bus with CPHA = 1 and CR(24) = 0 for 8-bit Character

Transfer Beginning Period

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

As a slave, the processor considers a transfer to begin with the first SCK edge or the falling edge of \overline{SS} , depending on the CPHA format selected. When CPHA equals zero, the falling edge of \overline{SS} indicates the beginning of a transfer. When CPHA equals one, the first edge on the SCK indicates the start of the transfer. In either CPHA format, a transfer can be aborted by de-asserting the $\overline{SS}(n)$ signal. This causes the SPI slave logic and bit counters to be reset. In this implementation, the software driver can deselect all slaves (i.e. $\overline{SS}(n)$ 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 OPB 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 under-run interrupt is asserted.

Transfer Ending Period

The definition of the transfer ending period for the OPB SPI 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 may be additional tasks to be performed before the system can consider the transfer complete.

When configured without FIFOs, the Rx_Full bit, bit(30) in the SPISR, is set to denote the end of transfer. When data is available in the SPIDRR register, bit(27) 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, Rx_Empty bit, bit(31) and Tx_Full bit, bit(28) in the SPISR are cleared, Tx_Empty bit, bit(29) and Rx_Full bit, bit(30) in SPISR are set, and DRR Full bit, bit(27) and Slave MODF bit, bit(30) in the IPISR are set on the first rising OPB clock edge after the end of the eighth SCK cycle.

Note that the end of the eighth 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 16](#). 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 permits the simultaneous setting of SPISR and IPISR bits for both master and slave SPI devices. Note that external SPI slave devices may 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 OPB SPI cannot be used with external slave devices that do not use the OPB clock.

When the OPB SPI is configured with FIFOs and a series of consecutive SPI 8-bit character 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 less than half full.

When the SPI device is configured as a slave, the setting/clearing of the bits discussed above 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' (i.e. no SCK edge denoting the end of the eighth clock period. See [Figure 16](#). The slave has no way of knowing when the end of the eighth SCK period occurs unless an OPB clock period counter was included in the SPI slave device.

Optional FIFOs

The user has the option to include FIFOs in the OPB SPI as shown in [Figure 1](#). Since 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. Both transmit and receive FIFOs are 16 bytes deep and are accessed via single OPB transactions since 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 has the data available for a retry of the transmission. The occupancy number is a read-only register.

If a write is attempted when the FIFO is full, then no acknowledgement is given and a bus time-out will occur. Interrupts associated with the transmit FIFO include data transmit FIFO empty, transmit FIFO half empty and transmit FIFO under-run. See the section on [OPB SPI Interrupt Descriptions](#) 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 no acknowledgement is given and a bus time-out will occur.

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.

The other interrupt associated with the receive FIFO is the receive FIFO full interrupt. SPI transfers, when the OPB SPI 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 bytes are written to the transmit FIFO, an interrupt will be asserted. When the OPB SPI SCK frequency ratio is sufficiently small, this scenario is highly probable.

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

Local Master Loopback Mode

Local master loopback mode, although not included in the M68HC11 reference manual, has been implemented to expedite testing. When this mode is selected via setting the loop 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 remote slave) is ignored. This mode 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 but 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 (i.e. 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 may 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 (i.e. its \overline{SS} bit is asserted) by another SPI device. When this condition is detected, IPISR bit(30) is set by a strobe to the IPISR register if the interrupt module is included in the OPB SPI design. This error detection does not exist if the interrupt module is not included in the OPB SPI.

Under-run and over-run conditions error detection is provided as well. Under-run 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 under-run interrupt is asserted and the slave shift register is loaded with all zeros for transmission. Over-run can happen to both master and slave devices where a transfer occurs when the receive register or FIFO is full. During an over-run condition, the data received in that transfer is not registered (i.e. it is lost) and the IPISR over-run interrupt bit(26) is asserted.

Software Freeze Command Operation

The software freeze command impacts only master operation and not slave operation. When a freeze command is asserted (i.e. freeze signal goes high), the master completes any transfer in progress, but does not initiate a subsequent SPI transfer until the freeze signal is de-asserted. Operation is identical to as if the SPI master SPIDTR register/FIFO is empty when the freeze signal is asserted.

SPI Protocol with Manual Slave Select Assertion

This section briefly describes the SPI protocol where slave select ($\overline{SS}(n)$) is manually asserted by the user (i.e. SPICR bit(24) = 1).

This is the configuration mode provided to permit transfers of an arbitrary number of bytes without toggling slave select until all the bytes are transferred. In this mode, the data in the SPISSR register appears directly on the $\overline{SS}(n)$ output.

As described earlier, SCK must be stable before the assertion of slave select. Therefore, when manual slave select mode is utilized, the SPI master must be enabled first (SPICR bit(24) = 1) to assert SCK to the idle state prior to asserting slave select.

Note that the master transfer inhibit (SPICR bit(23)) can be utilized to inhibit master transactions until the slave select is asserted manually and all data registers are initialized as desired.

When the above rules are followed, the timing is the same as presented for the automatic slave select assertion mode with the exception that assertion of slave select signal and the number of bytes transferred is controlled by the user.

Note that the master transfer inhibit can be asserted to permit loading of registers or FIFOs as needed. This can be utilized before the first transaction and after any transaction that is allowed to complete.

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 permits the transfer of N-bytes in a single toggling of the slave select vector (default mode). Follow these steps to successfully complete an SPI transaction:

1. Start from proper state including SPI bus arbitration.
2. Configure DGIE and IPIER registers as desired.
3. Configure target slave SPI device as required.
4. Write initial data to master SPIDTR register/FIFO. This assumes that the SPI master is disabled.
5. Insure SPISSR register has all ones.
6. Write configuration data to master SPI device SPICR as desired including setting bit(24) 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 above configuration data to master SPI device SPICR, but clear inhibit bit which starts transfer.
9. Wait for interrupt (typically IPISR bit(30)) or poll status for completion. Wait time depends on OPB to SPI clock ratio.
10. Set master transaction inhibit bit to service interrupt request. Write new data to master register/FIFOs and slave device then clear master transaction inhibit bit to continue N 8-bit character transfer. Note that an overrun of the SPIDRR register/FIFO can occur if the SPIDRR register/FIFOs are not read properly. Also note that SCK will have "stretched" idle levels between byte transfers (or groups of byte transfers if utilizing FIFOs) and that MOSI can transition at end of a byte 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 transfer (optional mode)

Follow these steps to successfully complete an SPI transaction:

1. Start from proper state including SPI bus arbitration.
2. Configure master DGIE and IPIER. Also configure slave DGIE 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(30)) 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)

Follow these steps to successfully complete an SPI transaction:

1. Start from proper state including SPI bus arbitration.
2. Configure master DGIE and IPIER. Also configure slave DGIE 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(30)) 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 transferred (optional mode)

Follow these steps to successfully complete an SPI transaction:

1. Start from proper state including SPI bus arbitration.

2. Configure master DGIE and IPIER. Also configure slave DGIE 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(30)) 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.

Design Constraints

Note: An example UCF for this core is available and must be modified for use in the system. Please refer to the *EDK Getting Started Guide* for the location of this file.

Timing Constraints

For complete timing coverage, it is recommended that Flip-Flop edge to edge constraint is specified along with OPB_Clk input constraint. An example is shown in **Figure 18**.

```
NET "OPB_Clk" TNM_NET = "opb_clk";
TIMESPEC "TS_opb_clk" = PERIOD "opb_clk" 10 ns HIGH 50%;

TIMEGRP "rise_group" = RISING ffs;
TIMEGRP "fall_group" = FALLING ffs;
TIMESPEC "TS_rise" = FROM "rise_group" TO "fall_group" "TS_opb_clk"/2;
TIMESPEC "TS_fall" = FROM "fall_group" TO "rise_group" "TS_opb_clk"/2;
```

DS464_18_071906

Figure 18: Timing Constraints

Design Implementation

Target Technology

The intended target technology is Spartan and Virtex family FPGAs.

EEPROM Test Matrix for Virtex-II Pro (XC2VP7-5-FF672)

Table 15 lists the EEPROM test carried out on Xilinx ML300 board with microchip 25LC160 device. This data can be used as a reference for hardware testing.

Table 15: EEPROM 25LC160 Test Matrix

C_OPB_SCK_RATIO	RESULT	OPB Frequency in MHz
2	Exceeds EEPROM Fmax	25
4	Exceeds EEPROM Fmax	25
16	Passed	25

Table 15: EEPROM 25LC160 Test Matrix

C_OPB_SCK_RATIO	RESULT	OPB Frequency in MHz
64	Passed	100
512	Passed	100
640	Passed	100
720	Passed	100
1024	Passed	100
1280	Passed	100
1440	Below EEPROM minimum Clock Rate	100
2048	Below EEPROM minimum Clock Rate	100

Device Utilization and Performance Benchmarks

The OPB SPI is a module that will be used with other design modules in the FPGA. The utilization and timing numbers reported in this section are just estimates. As the SPI device is combined with other FPGA designs, the utilization of FPGA resources and timing of the OPB SPI may vary from the results reported here.

The OPB SPI benchmarks are shown in Table 16 for a Virtex-4 FPGA device and in Table 17 for a Virtex-5 FPGA device.

Table 16: OPB SPI FPGA Performance and Resource Utilization Benchmarks on Virtex-4 (xc4vix25-11-ff676)

Parameter Values (other parameters at default values)				Device Resources			f _{MAX} (MHz)
C_FIFO_EXIST	C_INTERRUPT_PRESENT	C_OPB_SCK_RATIO	C_NUM_SS_BITS	Slice Flip-Flops	Slices	LUTs	
0	0	2	2	81	147	216	134.264
0	0	4	2	94	163	222	136.463
0	0	16	2	97	166	233	135.318
0	0	2048	2	103	169	242	126.486
0	0	2	12	89	170	254	138.102
0	1	2	2	85	156	222	130.378
0	1	4	2	98	163	230	131.787
0	1	16	2	101	172	240	129.702

Table 16: OPB SPI FPGA Performance and Resource Utilization Benchmarks on Virtex-4 (xc4vlx25-11-ff676)

Parameter Values (other parameters at default values)				Device Resources			f _{MAX} (MHz)
C_FIFO_EXIST	C_INTERRUPT_PRESENT	C_OPB_SCK_RATIO	C_NUM_SS_BITS	Slice Flip-Flops	Slices	LUTs	
0	1	2048	2	108	177	249	134.953
0	1	2	12	93	177	260	126.295
1	0	2	2	77	170	272	130.174
1	0	4	2	90	184	290	121.507
1	0	16	2	92	187	294	118.287
1	0	2048	2	99	193	303	122.444
1	0	2	12	84	193	312	129.333
1	1	2	2	84	179	281	121.966
1	1	4	2	97	192	299	128.041
1	1	16	2	99	196	303	125.818
1	1	2048	2	106	201	312	137.703
1	1	2	12	91	203	321	113.404

Table 17: OPB SPI FPGA Performance and Resource Utilization Benchmarks on Virtex-5 (xc5vlx30-2-ff676)

Parameter Values (other parameters at default values)				Device Resources		f _{MAX} (MHz)
C_FIFO_EXIST	C_INTERRUPT_PRESENT	C_OPB_SCK_RATIO	C_NUM_SS_BITS	Slice Flip-Flops	LUTs	
0	0	2	2	81	188	132.802
0	0	4	2	94	205	174.216
0	0	16	2	96	203	141.004

Table 17: OPB SPI FPGA Performance and Resource Utilization Benchmarks on Virtex-5 (xc5v1x30-2-ff676)

Parameter Values (other parameters at default values)				Device Resources		f _{MAX} (MHz)
C_FIFO_EXIST	C_INTERRUPT_PRESENT	C_OPB_SCK_RATIO	C_NUM_SS_BITS	Slice Flip-Flops	LUTs	
0	0	2048	2	103	209	137.741
0	0	2	12	88	221	164.88
0	1	2	2	85	198	143.184
0	1	4	2	98	208	139.899
0	1	16	2	100	215	125.913
0	1	2048	2	107	223	144.844
0	1	2	12	92	235	162.575
1	0	2	2	77	244	109.721
1	0	4	2	90	255	134.012
1	0	16	2	92	258	136.054
1	0	2048	2	99	267	131.165
1	0	2	12	84	277	126.486
1	1	2	2	84	250	138.427
1	1	4	2	97	260	150.512
1	1	16	2	99	265	152.3
1	1	2048	2	106	274	140.154
1	1	2	12	91	283	132.767

Specification Exceptions

Exceptions to the Motorola's 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(24) in the SPICR register. When the device is configured as a master and is enabled and bit(24) of SPICR register is set, the vector in the

SPISSR register is asserted. When this mode is enabled, multiple bytes 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 has main utility in manual control of the \overline{SS} signals.
6. In this implementation without FIFOs, both the SPIDTR and SPIDRR registers are double buffered. Hardware prevents data transfer from the transmit buffer to the shift register while an SPI transfer is in progress, consequently, the write collision error (WCOL) described in the M68HC11 reference manual can not occur.
7. 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 required or implemented in this design.
8. 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.
9. An option is implemented in this FPGA design to implement FIFOs on both transmit and receive (Full Duplex only) mode.
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. Thus, in this implementation run time configuration of baud rate is not possible. Furthermore, in addition to the prescribed ratios of 2, 4, 16 and 32, all integer multiples of 16 up to 2048 are allowed.

Reference Documents

The following documents contain reference information important to understanding the OPB SPI design:

- *Motorola M68HC11-Rev. 4.0 Reference Manual*
- *Motorola MPC8260 PowerQUICC II™ Users Manual 4/1999 Rev. 0*

Revision History

Date	Version	Revision
7/25/06	1.0	Initial Xilinx release.