SPI WISHBONE Controller

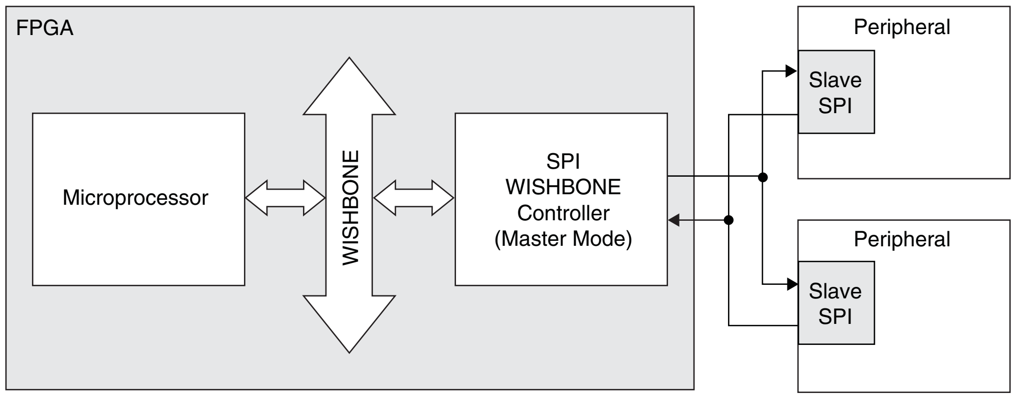
January 2015 **Reference Design RD1044**

Introduction

The Serial Peripheral Interface (SPI) bus provides an industry standard interface between microprocessors and other devices as shown in Figure 1. This reference design documents a SPI WISHBONE controller designed to provide an interface between a microprocessor with a WISHBONE bus and external SPI devices. In master mode, the SPI controller can be configured for communication with multiple off-chip SPI ports. In slave mode, the SPI supports communications with an off-chip SPI master.

As a simple serial port, the SPI uses few FPGA resources (see Table 6) and little board space for wires. This SPI reference design uses only three pins (clock, data in, and data out) plus one select for each slave device. A SPI is a good choice for communicating with low-speed devices that are accessed intermittently and transfer data streams rather than reading and writing to specific addresses. A SPI is an especially good choice if you can take advantage of its full-duplex nature, which sends and receives data at the same time.

Figure 1. Using the SPI WISHBONE Controller to Connect to Peripherals



Both Verilog and VHDL versions of the reference design are available. Lattice design tools are used for synthesis, place and route and simulation. The design can be targeted to multiple Lattice device families. Its small size makes it portable across different FPGA or CPLD architectures.

This design assumes the user has experience with WISHBONE controllers. Information available in the documents listed in the References section is not repeated in this document.

Theory of Operation

Overview

This SPI WISHBONE controller provides an interface between a microprocessor with a WISHBONE bus and a SPI device. The controller can either act as the SPI Master or SPI Slave device. The selection of the Master or Slave mode is done using parameters in the HDL code. The design uses a single module.

The SPI WISHBONE reference design provides standard, fully-configurable SPI ports including:

- WISHBONE B.3 interface
- Slave and master modes. Master mode can control up to eight slaves. More can be added if desired.
- Interrupt request to the processor, configurable for a variety of status conditions.
- Configurable serial clock (SCLK) frequency.

- Configurable timing relationships between data and clock signals, and between data and slave-select signals.
- Double-buffered transmission, allowing new data to be written at the same time that previous data is being shifted out.
- Receive and transmit registers configurable from 1 to 32 bits wide. Longer transfers can be done with software support.
- Option for least-significant bit or most-significant bit first.

Figure 2 shows the Lattice SPI WISHBONE controller configured as a master port, and Figure 3 shows it configured as a slave port.

Figure 2. Lattice SPI WISHBONE Controller Master Implementation

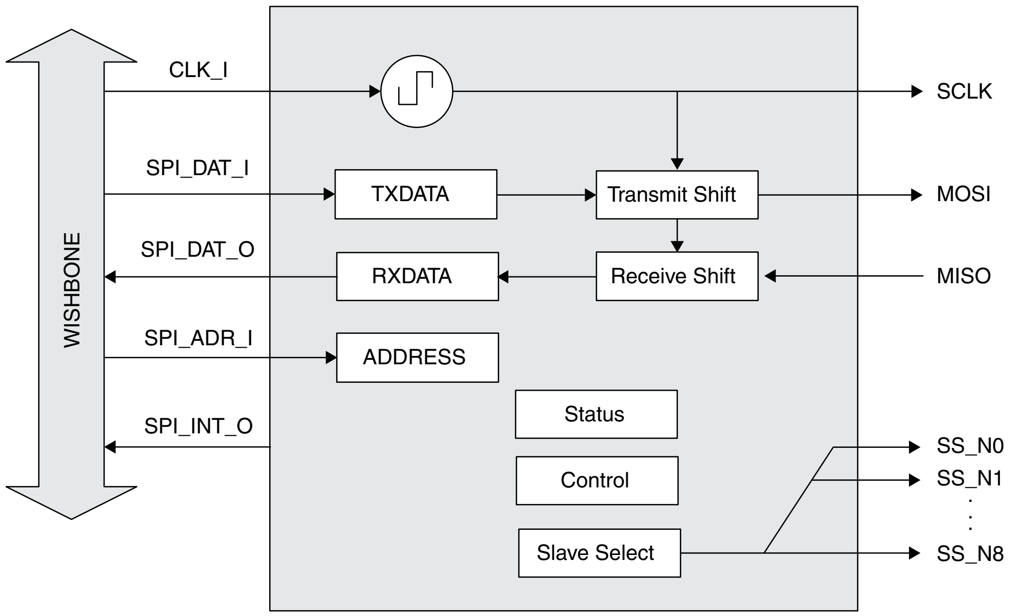
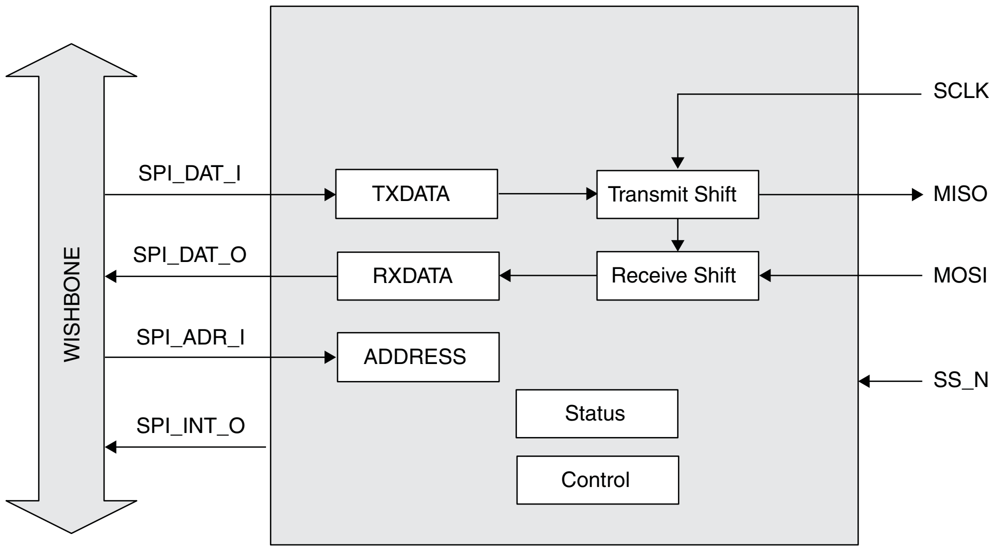


Figure 3. Lattice SPI WISHBONE Controller Slave Implementation



On the internal side (the left in these diagrams), the SPI has a standard WISHBONE slave bus, which connects the SPI with a microprocessor and other on-chip components. From the WISHBONE bus, the SPI appears as a set of addressable registers that can be read or written. Through these registers, the processor can transmit and receive data and control the operation of the SPI. Please note that only the major WISHBONE bus signals are shown in the figures above for clarity. A complete listing of the WISHBONE bus signals is shown in the Top-Level Port Description section below.

On the external side (the right in these diagrams), the SPI has a standard master or slave SPI interface:

- **SCLK** (serial clock) generated by the master SPI to synchronize the data transfers.
- **MISO** (master in, slave out), which transfers data going to the master SPI from a slave.
- **MOSI** (master out, slave in), which transfers data going from the master SPI to a slave.
- **SS_N** (slave select), which is asserted by the master SPI to start a data transfer. In master mode, the SPI has a slave select signal (SS_N0, SS_N1, and so on) for each slave SPI. In slave mode, the SPI has a single SS_N input.

HDL Parameter Descriptions

The SPI WISHBONE controller has a number of parameters that are used to control the configuration of the controller. This allows the user to modify the configuration to meet their needs without making changes to the Verilog code itself. Table 1 provides descriptions of the parameters used in the SPI WISHBONE controller.

Table 1. SPI WISHBONE Parameter Descriptions

Parameter	Description	Value
MASTER	Specifies whether the SPI controller is a master or slave device. 1 or True = master.	0 | 1
SLAVE_NUMBER	Specifies the number of slave devices supported.	1-8 (32)[1]
DATA_LENGTH	Specifies the number of serial data bits.	1-8 (32)[2]
SHIFT_DIRECTION	Specifies whether the most significant bit or least significant bit is first. 0 = msb first	0 | 1
CLOCK_PHASE	Specifies the clock phase of the SPI instance. If 0, the data is latched on the leading edge of SCLK and data changes on the trailing edge of SCLK. If 1, the data is latched on the trailing edge of SCLK and data changes on the leading edge of SCLK.	0 | 1
CLOCK_POLARITY	Specifies the Polarity of the SPI instance. If 0, the idle state for SCLK is low. If 1, the idle state for SCLK is high.	0 | 1
CLKCNT_WIDTH	Specifies the range limit for the clock counter. The width must be enough to meet the number of bits required for the slave clock (SCLK).	1-32
CLOCK_SEL[3]	Specifies the factor for deriving SCLK from the processor clock CLK_I. SCLK is derived using the following equation: SCLK = CLK_I / (2*(CLOCK_SEL+1))	$0\text{-}2^{\text{clkcnt\_width}}\text{-}1$
DELAY_TIME	Specifies the time delay factor before shifting the first bit of data after the SS_N signal is asserted. The start delay time is derived from the following equation: Delay = DELAY_TIME *(SCLK period/ 2)	0-63
INTERVAL_LENGTH	Specifies the number of SCLK cycles for which the SS_N signal is held inactive between SPI transmit requests.	0-63

1. The number of slave devices can be increased if a wider SPI_DAT_I bus is used.
2. The data length can support up to 32 bits by changing the width of the SPI_DAT_I bus. Please see appendix A for more details.
3. The SPI WISHBONE Controller source code file has a CLOCK_SEL parameter value of 1 in order to speed up SCLK for use in simulations. Please check the maximum SPI clock rate for the devices you are using in determining the correct value for the CLOCK_SEL parameter.

Top-Level Port Descriptions

Table 2 provides descriptions of the input/output ports of the SPI WISHBONE controller. Port names ending with "_N" indicate an active low signal. This convention is used throughout the design.

Table 2. SPI WISHBONE Controller Port Descriptions

Signal	Type	Active	Description
WISHBONE Interface			
CLK_I	Input	High	Input Clock signal
RST_I	Input	High	System Reset signal
SPI_ADR_I[7:0]	Input	N/A	Slave address bus
SPI_DAT_I[7:0]	Input	N/A	Slave data input bus
SPI_DAT_O[7:0]	Output	N/A	Slave data output bus
SPI_WE_I	Input	High	Slave write enable signal
SPI_CYC_I	Input	High	Slave cycle signal
SPI_STB_I	Input	High	Slave strobe signal
SPI_SEL_I[3:0]	Input	High	Slave select signal
SPI_CTI_I[2:0]	Input	High	Slave cycle-type indicator signal
SPI_BTE_I[1:0]	Input	High	Slave burst type signal
SPI_LOCK_I	Input	High	Slave bus locked signal
SPI_ACK_O	Output	High	Slave acknowledge signal
SPI_ERR_O	Output	High	Slave error signal
SPI_RTY_O	Output	High	Slave retry signal
SPI_INT_O	Output	High	Slave interrupt request signal
SPI Interface			
MISO_MASTER	Input	High	Master input/slave output
MOSI_MASTER	Output	High	Master output/slave input
SCLK_MASTER	Output	Selectable	Serial clock - user-selectable
SS_N_MASTER	Output	Low	SPI slave select (active low)
MISO_SLAVE	Output	High	Master input/slave output
MOSI_ SLAVE	Input	High	Master output/slave input
SCLK_ SLAVE	Input	Selectable	Serial clock - user-selectable
SS_N_ SLAVE	Input	Low	SPI slave select (active low)

Register Descriptions

The SPI WISHBONE controller has five registers which are used to interact with the WISHBONE bus. A description of each of these registers is shown in Table 3. Further definition of the status and control registers can be found in Tables 4 and 5.

Table 3. SPI WISHBONE Register Descriptions

Register	Address	Width	WISHBONE Access	Description
REG_RXDATA	0x00	8[1]	Read only	Data from SPI port
REG_TXDATA	0x04	8[1]	Read/write	Data to SPI port, SSMASK or control register
REG_STATUS	0x08	8	Read only	Status register
REG_CONTROL	0x0C	8	Read/write	Control register
REG_SSMASK	0x10	8[1]	Read/write	Slave select register

1. The width can be increased to support up to 32 bits with modifications. Please see appendix A for details.

Table 4. SPI WISHBONE Status Register Bit Definitions

Bit Name	Bit	Description
Reserved	1:0	Not used
ROE	2	Receive overrun error: 1 = error. Error indicates that the RXDATA register received new data before the previous data was read. The previous data was lost if this occurs.
TOE	3	Transmit overrun error: 1 = error. Error indicates that the TXDATA register received new data before the previous data was moved to the shift register. The new data is discarded if this occurs.
TMT	4	Transmit shift register is empty: 1 = empty. When the transmit shift register is empty the next data will be transferred from the TXDATA register unless it is empty.
TRDY	5	Transmit ready status: 1 = empty. Indicates that the TXDATA register is empty and can accept new data from the WISHBONE bus. New data will be blocked until TRDY = 1.
RRDY	6	Receive ready status: 1 = data available. Indicates that the RXDATA register has data and is ready to be read. Reading the RXDATA register clears RRDY.
E	7	Error bit: 1 = error. This bit indicates that an overrun error has occurred. The E bit is the logical OR of the ROE and TOE status bits.

Table 5. SPI WISHBONE Control Register Bit Definitions

Bit Name	Bit	Description
IROE	0	Set to 1 to enable interrupt requests for receive overrun errors.
ITOE	1	Set to 1 to enable interrupt requests for transmit overrun errors.
Reserved	2	Not used
ITRDY	3	Set to 1 to enable interrupt requests for transmitter ready conditions.
IRRDY	4	Set to 1 to enable interrupt requests for receiver ready conditions.
IE	5	Set to 1 to enable interrupt requests for transmit or receive overrun errors.
Reserved	6	Not used
SSO	7	In a master SPI, set to 1 to assert the SS_N outputs according to the mask in the slave select register. SSO holds the slave select signal after it would normally be de-asserted. The SPI will continue to exchange data frames until SSO is cleared.

SPI WISHBONE Controller Operation

The Controller's WISHBONE interface supports only classic cycle transfers, which means that the System master CTI_O is fixed at 000. The interface does not support cache line wrap; the System master BTE_O is fixed at 00. The slave port does not have RTY_O and ERR_O signals. The RTY_O and ERR_O signals are terminated low. The slave adds an interrupt port (SPI_INT_O) to the master (processor). If the SPI WISHBONE Controller is configured for 32-bit operation, only a master processor with 32-bit operation is supported; S_SEL_I is not supported.

Data Transmit and Receive

Both of the data transmit and receive paths within the Lattice SPI WISHBONE controller use two registers: a holding register and a shift register. This double buffering allows the paths to hold one data frame while another is being shifted in or out. The holding registers, TXDATA and RXDATA, are addressable and can be written to or read through the WISHBONE bus.

In the transmit path, TXDATA is written to (or read) through the WISHBONE bus. Writing to TXDATA clears the transmitter ready status bit (TRDY) to 0, blocking any new data until the previous data has moved to the shift register. If no serial transfer is in process, TXDATA immediately moves its data to the shift register and sets TRDY to 1. If a serial transfer is in process, TXDATA holds the new data until the previous data has shifted out. If new data comes in while TRDY is 0, the new data is blocked and the transmit overrun error status bit (TOE) is set.

In the receive path, the shift register, when full, immediately moves its data to the holding register, RXDATA, and sets the receiver ready status bit (RRDY) to 1. RXDATA can be read through the WISHBONE bus. Reading

RXDATA clears the RRDY status bit. If new data comes in while RRDY is set, RXDATA is overwritten with the new data and the receive overrun error status bit (ROE) is set.

SPI operations are always full duplex, so every data transfer operation transmits and receives at the same time. If you just want to receive, your software must load TXDATA with appropriate dummy data to transmit. Your software must also use or ignore the received data as appropriate.

The registers in this SPI WISHBONE controller are eight bits wide but can be configured up to 32 bits wide. See Appendix A for more details. The transmit and receive logic can be configured to assume the data is either least-significant bit (LSB) first or most-significant bit (MSB) first.

Status and Control

The SPI WISHBONE controller includes status and control registers. These are mainly used to trigger interrupt requests. The master SPI also has a slave-select register that is used to select a slave SPI and start a data transfer.

To check the status of the SPI WISHBONE controller, read the status register. It reports conditions such as receive and transmit overrun, transmit shift register empty, and transmitter and receiver ready. For details, see Table 4.

To set up an interrupt request on the SPI_INT_O output, set one or more of the interrupt enable bits in the control register. These bits enable interrupt requests for most of the conditions reported in the status register. For details, see Table 5.

To clear an interrupt request, clear the associated bit in the control register. Writing any value to the control register clears the overrun error status bits (ROE, TOE, and E). Writing the TXDATA register clears the transmit ready status bit (TRDY). Reading the RXDATA register clears the receiver ready status bit (RRDY).

To start a data transfer, load the slave select register of the master SPI with a slave mask and then load the TXDATA register. Loading TXDATA triggers the next data transfer. The slave select register has one bit for each SS_N output. Setting the bit to 1 asserts the active-low SS_N. For example, a binary slave mask of 00010000 asserts SS_N5, selecting that slave SPI. It is possible to assert more than one slave select signal, but you must take care to prevent contention on the MISO bus.

The SPI WISHBONE controller uses a SPI_DAT_I bus which is has an 8-bit width. If the application requires more than eight SPI Slave devices, then the designer must modify the design to use an alternate encoding scheme and also add some logic to decode the slave selected from the SPI_DAT_I bus using the selected encoding scheme. If the design is modified to accept a wider SPI_DAT_I bus then it is not necessary to change the existing encoding scheme. See Appendix A for more details.

To have a data transfer longer than the transmit and receive registers, set the SSO bit in the control register to 1. SSO holds the slave select signal after it would normally be de-asserted. The SPIs will continue to exchange data frames until SSO is cleared. SCLK stops toggling between frames. Before clearing SSO to end the data transfer, make sure the transmit shift register is empty by checking the TMT status bit.

A slave SPI cannot start a data transfer. When the SPI’s SS_N input goes low, the Slave SPI Controller immediately begins the data transfer.

Clocking Sources

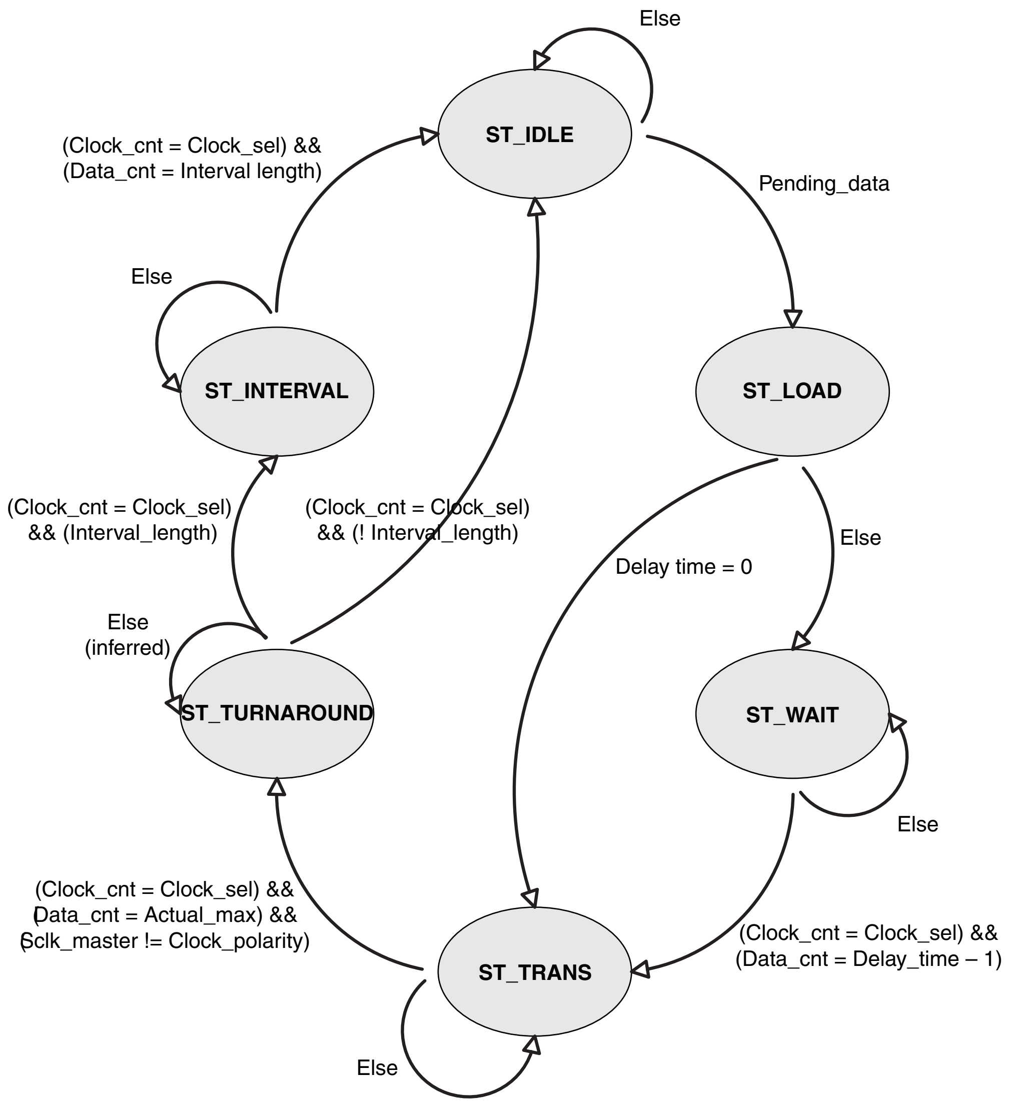
The master SPI generates SCLK to synchronize the data transfers. SCLK is only available during the data transfer, while SS_N is asserted.

SCLK is derived from the system clock, CLK_I, by dividing the frequency. The divisor and other aspects of SCLK can be selected when the SPI is configured, as shown in Table 1.

Module Description

The state machine used in the Master mode of this design is shown in Figure 4. The Slave mode does not use a state machine. This state machine starts in the ST_IDLE state (at top) and waits until there is pending data indicating that a write operation has been initiated. The pending data status is indicated by the microprocessor writing a value of 0x04 to the address register across the WISHBONE bus and then asserting the SPI_STB_I, SPI_WE_I, and SPI_CYC_I signals. The state machine then transitions through the various states until the write has been completed when it return to the ST_IDLE state.

Figure 4. State Machine



Test Bench Description

The test bench for this design consists of a single module called spi_tf (the filename is spi_wb_tb1.v). This module instantiates the SPI WISHBONE Controller and provides the stimulus inputs for the WISHBONE ports as well as the CLK_I and RST_I signals. This test bench is used for a Master SPI implementation of the SPI WISHBONE Controller. The CLK_I is generated at 50 MHz in the test bench.

The SPI WISHBONE Controller source code file has a CLOCK_SEL parameter value of 1 in order to speed up SCLK_MASTER for use in the simulation. A hardware implementation of this design may require this parameter to be changed for successful SPI communications to occur. Please check the maximum SPI clock rate for the devices you are using in determining the value for the CLOCK_SEL parameter.

Design Flow

Lattice design tools are used for synthesis, simulation and place and route. In addition to the place and route/fitter engine, the Lattice ispLEVER® design tool includes Synplify®/Synplify Pro® from Synplicity® for synthesis, and Active-HDL® from Aldec® for simulation. The details of the design flow can be found in the rd1044_readme.txt file that comes with the reference design.

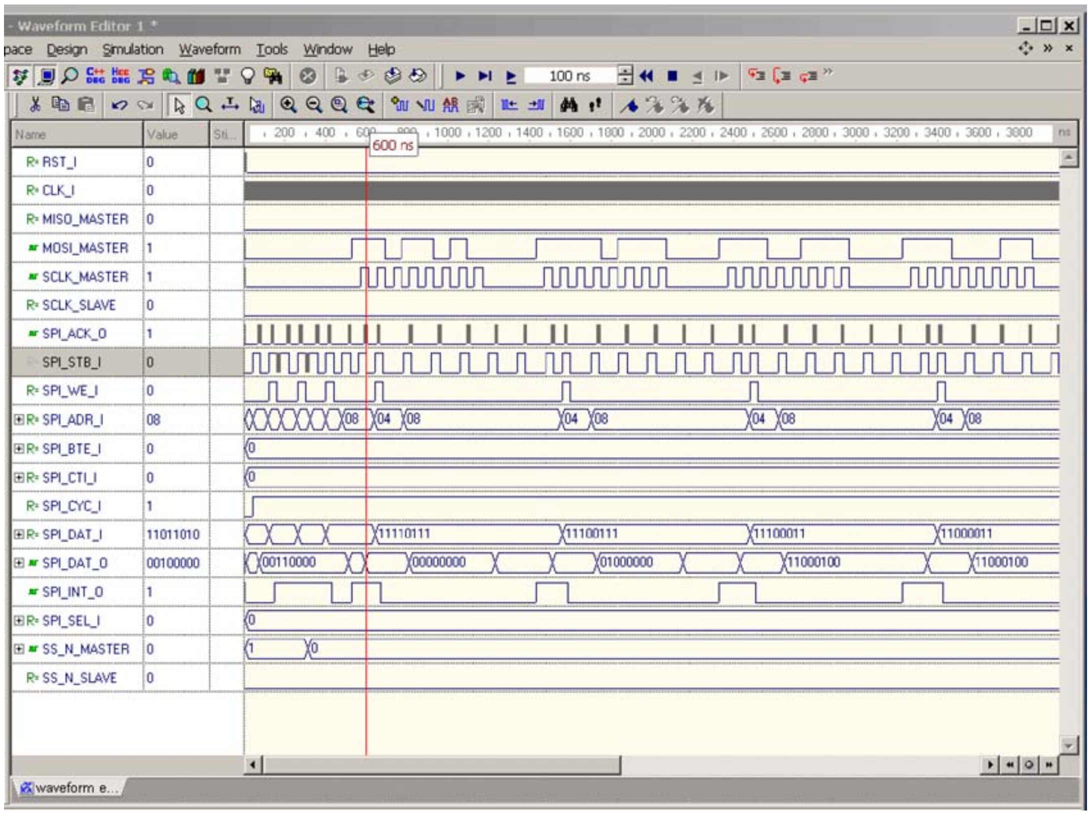
Timing Diagrams

The RTL Simulation timing diagram (Figure 5) shows a series of write commands being executed by the microprocessor across the WISHBONE bus with the SPI Controller in a Master configuration. The SPI WISHBONE Controller source code file has a CLOCK_SEL parameter value of 1 in order to speed up SCLK for use in simulations. Please check the maximum SPI clock rate for the devices you are using in determining the correct value for the CLOCK_SEL parameter.

Before writing data to the SPI WISHBONE Controller, the microprocessor reads the status register to insure that the controller is ready to accept data as indicated by the TRDY status bit (bit 5 of the Status register). Then the microprocessor sends the data and address across the SPI_DAT_I and SPI_ADR_I lines while asserting the write enable signal, SPI_WE_I, and the strobe, SPI_STB_I. Note that the SPI_CYC_I signal must also be asserted. For this example, the SPI_CYC_I signal is asserted at the beginning of the simulation and left asserted for the duration of the transaction.

In Figure 5, the cursor at 600 ns shows that the SCLK_MASTER signal is just beginning to clock the data out from the transfer register to the SPI device. The value of the data being sent, 11011010, is also shown in the SPI_DAT_I register (in the value column) since no data has overwritten it yet. The SPI_ADR_I register value is 0x08 which specifies a read status register command was executed at the last SPI_STB_I transition. The result of the status read is shown in the SPI_DAT_O register, 00100000 (in the value column), and this indicates that the SPI Controller is ready to accept new data (TRDY = 1). Therefore the next transaction will be a transmit and the SPI_WE_I is asserted, the SPI_ADR_I value is changed to 0x04 (transmit data), and the value of 11110111 is written to the SPI_DAT_I register. This value is the next value that will be sent out the SPI port. The process then repeats by reading the status register repeatedly to determine when the controller is ready to accept new data. For this simulation, the data reads occur every 140 ns until the TRDY bit has changed state and the write process can begin again. This simulates a microprocessor transaction which does not use the interrupt from the SPI Controller, SPI_INT_O.

Figure 5. RTL Simulation Timing Diagram



There is a total of four bytes of data sent from the SPI Controller in the timing diagram of Figure 5, and the fifth byte is loaded into the controller to be sent out in the next cycle. The data cycle shown is: 11011010, 11110111, 11100111, 111000011, 11000011. Please note that the SS_N_MASTER signal was driven low earlier by writing a value of 00000001 to the SPI_DAT_I signal, and a value of 0x10 to the SPI_ADR_I signal.

Implementation

This design is implemented in Verilog and VHDL. When using this design in a different device, density, speed, or grade, performance and utilization may vary. Default settings are used during the fitting of the design.

Table 6. Performance and Resource Utilization[1]

Device Family	Language	Speed Grade	Utilization[2]	f_{MAX} (MHz)	I/Os[3]	Architecture Resources
MachXO3L[9]	Verilog-LSE	–6	126 LUTs	>60	38	N/A
	Verilog-Syn	–6	121 LUTs	>60	38	N/A
	VHDL-LSE	–6	123 LUTs	>60	38	N/A
	VHDL-Syn	–6	121 LUTs	>60	38	N/A
MachXO2™ [4]	Verilog-LSE	–6	126 LUTs	>60	38	N/A
	Verilog-Syn	–6	121 LUTs	>45	38	N/A
	VHDL-LSE	–6	125 LUTs	>60	38	N/A
	VHDL-Syn	–6	121 LUTs	>45	38	N/A
MachXO™ [5]	Verilog-LSE	–5	123 LUTs	>60	38	N/A
	Verilog-Syn	–5	116 LUTs	>45	38	N/A
	VHDL-LSE	–5	122 LUTs	>60	38	N/A
	VHDL-Syn	–5	122 LUTs	>45	38	N/A
LatticeECP3™ [6]	Verilog	–6	119 LUTs	>45	38	N/A
	VHDL	–6	120 LUTs	>45	38	N/A
LatticeXP2™ [7]	Verilog	–5	165 LUTs	>60	38	N/A
	VHDL	–5	164 LUTs	>60	38	N/A
ispMACH® 4000ZE [8]	Verilog	–5 (ns)	94 Macrocells	>45	38	N/A
	VHDL	–5 (ns)	94 Macrocells	>45	38	N/A

1. This utilization is for a WISHBONE interface with 8-bit address and data buses.
2. For LUT counts using Lattice Diamond® design software, the strategy option under Synplify Pro was set to Area=True.
3. The number of I/Os is for the SPI WISHBONE Controller reference design utilized as a master device with a single slave SPI device. A single external SPI port uses only five I/Os plus the clock and reset inputs. The remainder are WISHBONE I/Os which will normally be connected internally within the device.
4. Performance and utilization characteristics are generated using LCMXO2-1200HC-6TG144C with Lattice Diamond 3.3 design software.
5. Performance and utilization characteristics are generated using LCMXO2280C-5FT256C with Lattice Diamond 3.3 design software.
6. Performance and utilization characteristics are generated using LFE3-70EA-6FN484C with Lattice Diamond 1.2 design software.
7. Performance and utilization characteristics are generated using LFXP2-5E-5FT256C with Lattice Diamond 3.3 design software.
8. Performance and utilization characteristics are generated using LC4256ZE-5TN144C with Lattice ispLEVER® Classic 1.4 software.
9. Performance and utilization characteristics are generated using LCMXO3L-4300C-6BG256C, with Lattice Diamond 3.3 using Synplify Pro® and LSE (Lattice Synthesis Engine).

References

- WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores Revision: B.3, Released: September 7, 2002
- Freescale Semiconductors, M68HC11 Microcontroller Reference Manual - M68HC11RM/D Rev. 6.1, Section 8, Synchronous Serial Peripheral Interface.

Technical Support Assistance

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
January 2015	1.8	Updated Implementation section. Updated Table 6, Performance and Resource Utilization. — Updated values. — Added support for LSE and Synplify Pro. — Added support for Lattice Diamond 3.3 design software.
March 2014	01.7	Updated Implementation section. Updated Table 6, Performance and Resource Utilization. — Added support for MachXO3L device family. — Added support for Lattice Diamond 3.1 design software.
		Updated corporate logo.
		Updated Technical Support Assistance information.
April 2011	01.6	Added support for the LatticeECP3 device family.
		Added support for Lattice Diamond 1.2 design software.
January 2011	01.5	Added Address column to SPI WISHBONE Register Descriptions table.
November 2010	01.4	Added support for MachXO2 device family.
		Added support for Lattice Diamond and ispLEVER 8.1 SP1 software.
December 2009	01.3	Added support for LatticeXP2 device family.
		Updated for ispLEVER 8.0.
August 2009	01.2	Added support for VHDL language.
July 2009	01.1	Added support for ispMACH 4000ZE CPLD family.
February 2009	01.0	Initial release. This design was adapted from the LatticeMico32™ SPI core. The Address bus, Data_in, and Data_out buses were reduced from 32 bits to 8 bits to interface with the LatticeMico8™ Microcontroller reference design.

Appendix A. Using the SPI WISHBONE Controller with a 32-Bit Data Bus

If the designer wishes to use the SPI WISHBONE Controller reference design in a processor application that supports 32-bit data accesses, the SPI_DAT_I and SPI_DAT_O buses must be modified in the Verilog code in addition to changing the DATA_LENGTH parameter value. There are also two registers that need to be modified as shown below.

The SPI_ADR_I bus does not need to be changed. However, please note that this reference design uses a different SPI_ADR_I bus arrangement than is used by the LatticeMico32 SPI controller design.

Table 7. Ports and Registers that Must be Changed to Support a WISHBONE 32-Bit Data Bus Processor

```
input [31:0]     SPI_DAT_I;       //change from 8-bit width to 32-bit
output [31:0]    SPI_DAT_O;       //change from 8-bit width to 32-bit
reg [31:0]       SPI_DAT_O;       //change from 8-bit width to 32-bit
reg [31:0]       latch_s_data;    //change from 8-bit width to 32-bit
```

The SS_Nx signal is decoded from the SPI_DAT_I bus using a one-hot encoding scheme in the SPI WISHBONE Controller reference design. Each bit position represents the SS_Nx signal for an individual Slave SPI device. Changing the SPI_DAT_I bus to a 32-bit width also allows the designer to use up to 32 SPI slave devices without requiring additional changes to the design.