



SPI Master Core Specification

*Author: Simon Srot
simons@opencores.org*

**Rev. 0.2
July 12, 2002**

Revision History

Rev.	Date	Author	Description
0.1	June 13, 2002	Simon Srot	First Draft
0.2	July 12, 2002	Simon Srot	Document is lectured.

Contents

CONTENTS	II
INTRODUCTION	1
IO PORTS	2
2.1 WISHBONE INTERFACE SIGNALS	2
2.2 SPI EXTERNAL CONNECTIONS	2
REGISTERS	3
3.1 CORE REGISTERS LIST	3
3.2 DATA RECEIVE REGISTER [RX]	3
3.3 DATA TRANSMIT REGISTER [TX]	3
3.4 CONTROL AND STATUS REGISTER [CTRL]	4
3.5 DIVIDER REGISTER [DIVIDER]	5
3.6 SLAVE SELECT REGISTER [SS]	5
OPERATION	6
4.1 WISHBONE INTERFACE	6
4.2 SERIAL INTERFACE	6
ARCHITECTURE	8
CORE CONFIGURATION	9

1

Introduction

This document provides specifications for the SPI (Serial Peripheral Interface) Master core. Synchronous serial interfaces are widely used to provide economical board-level interfaces between different devices such as microcontrollers, DACs, ADCs and other. Although there is no single standard for a synchronous serial bus, there are industry-wide accepted guidelines based on two most popular implementations:

- SPI (a trademark of Motorola Semiconductor)
- Microwire/Plus (a trademark of National Semiconductor)

Many IC manufacturers produce components that are compatible with SPI and Microwire/Plus.

The SPI Master core is compatible with both above-mentioned protocols as master with some additional functionality. At the hosts side, the core acts like a WISHBONE compliant slave device.

Features:

- **Full duplex synchronous serial data transfer**
- **Variable length of transfer word up to 32 bits**
- **MSB or LSB first data transfer**
- **Rx and Tx on both rising or falling edge of serial clock independently**
- **8 slave select lines**
- **Fully static synchronous design with one clock domain**
- **Technology independent Verilog**
- **Fully synthesizable**

2

IO ports

2.1 WISHBONE interface signals

Port	Width	Direction	Description
wb_clk_i	1	Input	Master clock
wb_rst_i	1	Input	Synchronous reset, active high
wb_adr_i	5	Input	Lower address bits
wb_dat_i	32	Input	Data towards the core
wb_dat_o	32	Output	Data from the core
wb_sel_i	4	Input	Byte select signals
wb_we_i	1	Input	Write enable input
wb_stb_i	1	Input	Strobe signal/Core select input
wb_cyc_i	1	Input	Valid bus cycle input
wb_ack_o	1	Output	Bus cycle acknowledge output
wb_err_o	1	Output	Bus cycle error output
wb_int_o	1	Output	Interrupt signal output

Table 1: Wishbone interface signals

All output WISHBONE signals are registered and driven on the rising edge of wb_clk_i. All input WISHBONE signals are latched on the rising edge of wb_clk_i.

2.2 SPI external connections

Port	Width	Direction	Description
/ss_pad_o	8	Output	Slave select output signals
sclk_pad_o	1	Output	Serial clock output
mosi_pad_o	1	Output	Master out slave in data signal output
miso_pad_i	1	Input	Master in slave out data signal input

Table 2: SPI external connections

3

Registers

3.1 Core Registers list

Name	Address	Width	Access	Description
Rx	0x00	32	R	Data receive register
Tx	0x00	32	R/W	Data transmit register
CTRL	0x04	32	R/W	Control and status register
DIVIDER	0x08	32	R/W	Clock divider register
SS	0x0c	32	R/W	Slave select register

Table 3: List of core registers

All registers are 32-bit wide and accessible only with 32 bits (all `wb_sel_i` signals must be active).

3.2 Data receive register [Rx]

Bit #	31:0
Access	R
Name	Rx

Table 4: Data Receive register

Reset Value: 0x00000000

Rx

The Data Receive register holds the value of received data of the last executed transfer. Valid bits depend on the character length field in the CTRL register (i.e. if CTRL[7:3] is set to 0x08, bit Rx[7:0] holds the received data).

NOTE: The Data Received register is a read-only register. A Write to this register will actually modify the Transmit register because these two registers share the same FFs.

3.3 Data transmit register [Tx]

Bit #	31:0
Access	R/W

Name	Tx
-------------	----

Table 5: Data Transmit register

Reset Value: 0x00000000

Tx

The Data Receive register holds the data to be transmitted in the next transfer. Valid bits depend on the character length field in the CTRL register (i.e. if CTRL[7:3] is set to 0x08, the bit Rx[7:0] holds the received data).

3.4 Control and status register [CTRL]

Bit #	31:10	9	8	7:3	2	1	0
Access	R	R/W	R/W	R/W	R/W	R/W	R/W
Name	Reserved	IE	LSB	CHAR_LEN	Tx_NEG	Rx_NEG	GO_BSY

Table 6: Control and Status register

Reset Value: 0x00000000

IE

If this bit is set, the interrupt output is set active after a transfer is finished. The Interrupt signal is deasserted after a Read or Write to any register.

LSB

If this bit is set, the LSB is sent first on the line (bit Tx[0]), and the first bit received from the line will be put in the LSB position in the Rx register (bit Rx[0]). If this bit is cleared, the MSB is transmitted/received first (which bit in Tx/Rx register that is depends on the CHAR_LEN field in the CTRL register).

CHAR_LEN

This field specifies how many bits are transmitted in one transfer. Up to 32 bits can be transmitted.

CHAR_LEN = 0x01 ... 1 bit

CHAR_LEN = 0x02 ... 2 bits

...

CHAR_LEN = 0x1f ... 31 bits

CHAR_LEN = 0x00 ... 32 bits

Tx_NEG

If this bit is set, the mosi_pad_o signal is changed on the falling edge of a sclk_pad_o clock signal, or otherwise the mosi_pad_o signal is changed on the rising edge of sclk_pad_o.

Rx_NEG

If this bit is set, the miso_pad_i signal is latched on the falling edge of a sclk_pad_o clock signal, or otherwise the miso_pad_i signal is latched on the rising edge of sclk_pad_o.

GO_BSY

Writing 1 to this bit starts the transfer. This bit remains set during the transfer and is automatically cleared after the transfer finished. Writing 0 to this bit has no effect.

NOTE: All registers, including the CTRL register, should be set before writing 1 to the GO_BSY bit in the CTRL register. The configuration in the CTRL register must be changed with the GO_BSY bit cleared, i.e. two Writes to the CTRL register must be executed when changing the configuration and performing the next transfer, firstly with the GO_BSY bit cleared and secondly with GO_BSY bit set to start the transfer. When a transfer is in progress, writing to any register of the SPI Master core has no effect.

3.5 Divider register [DIVIDER]

Bit #	31:16	15:0
Access	R	R/W
Name	Reserved	DIVIDER

Table 7: Divider register

Reset Value: 0x0000ffff

DIVIDER

The value in this field is the frequency divider of the system clock `wb_clk_i` to generate the serial clock on the output `sclk_pad_o`. The desired frequency is obtained according to the following equation:

$$f_{sclk} = \frac{f_{wb_clk}}{(DIVIDER + 1) * 2}$$

3.6 Slave select register [SS]

Bit #	31:8	7:0
Access	R	R/W
Name	Reserved	SS

Table 8: Slave Select register

Reset Value: 0x00000000

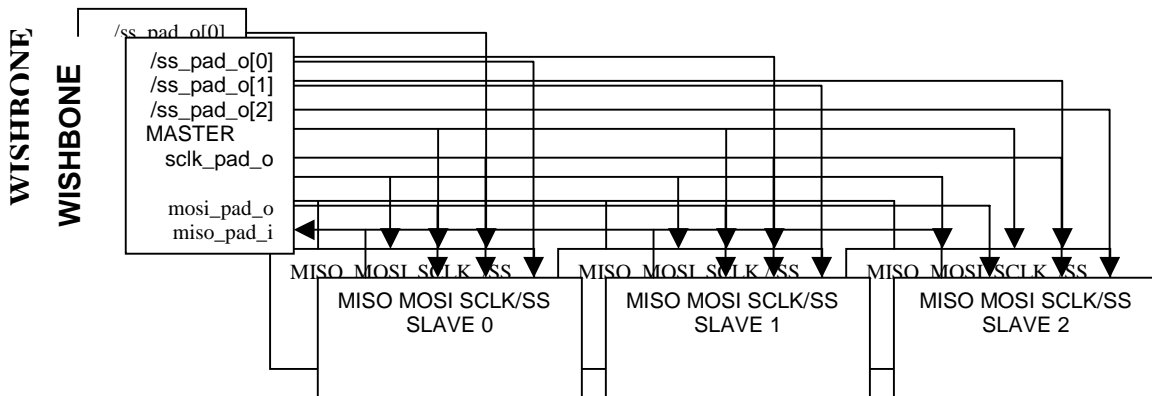
SS

Writing 1 to any bit location of this field sets the proper `ss_pad_o` line to an active state.

4

Operation

This core is an SPI and Microwire/Plus compliant synchronous serial controller. At the host side, it is controlled via registers accessible through a WISHBONE rev. B1 interface.



4.1 WISHBONE interface

The SPI core has four 32-bit registers through the WISHBONE rev. B1 compatible interface. All accesses to SPI registers must be 32-bit ($wb_sel[3:0] = 0xf$). Please refer to the WISHBONE specification at

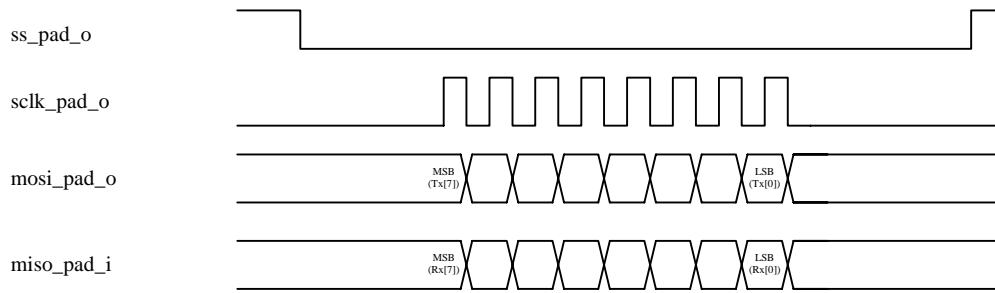
http://www.opencores.org/wishbone/specs/wbspec_b1.pdf

4.2 Serial interface

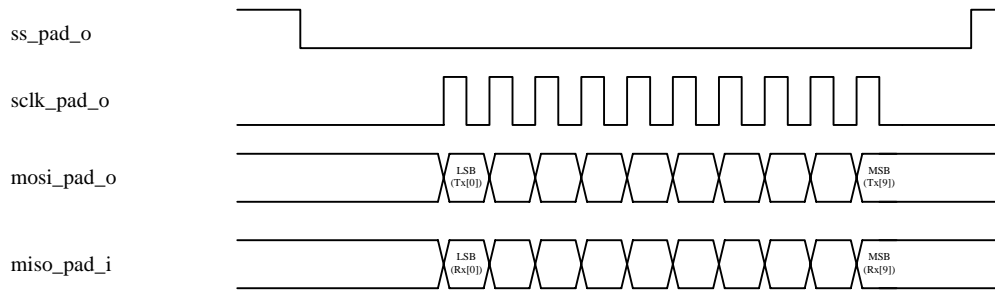
The serial interface consists of slave select lines, serial clock lines, as well as input and output data lines. All transfers are full duplex transfers of a programmable number of bits per transfer (up to 32 bits).

Compared to the SPI/Microwire protocol, this core has some additional functionality. It can drive data to the output data line in respect to the falling (SPI/Microwire compliant) or rising edge of the serial clock, and it can latch data on an input data line on the rising (SPI/Microwire compliant) or falling edge of a serial clock line. It also can transmit (receive) the MSB first (SPI/Microwire compliant) or the LSB first.

It is important to know that the Rx and Tx registers share the same flip-flops, which means that what is received from the input data line in one transfer will be transmitted on the output data line in the next transfer if no write access to the Tx register is executed between the transfers.



CTRL[LSB] = 0, CTRL[CHAR_LEN] = 0x08, CTRL[TX_NEG] = 1, CTRL[RX_NEG] = 0

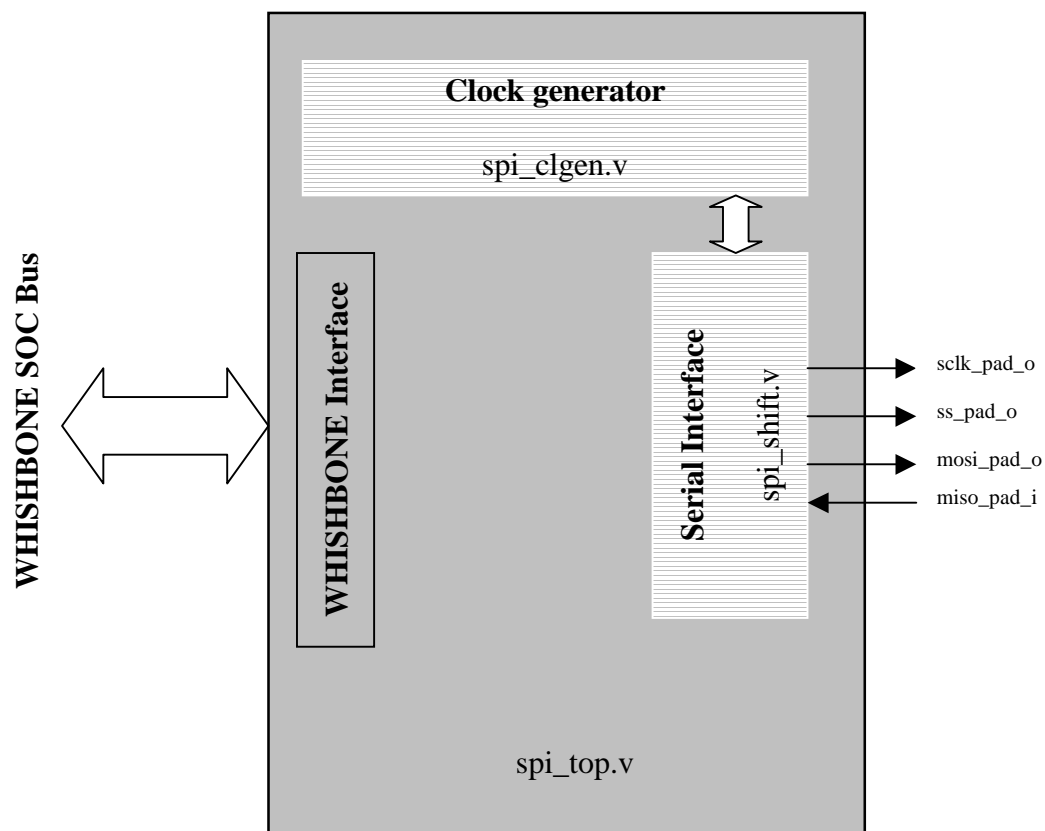


CTRL[LSB] = 1, CTRL[CHAR_LEN] = 0x0a, CTRL[TX_NEG] = 0, CTRL[RX_NEG] = 1

5

Architecture

The SPI Master core consists of three parts shown in the following figure:



Appendix A

Core configuration

To meet specific system requirements and size constraints on behalf of the core functionality, the SPI Master core can be configured by setting the appropriate define directives in the `spi_defines.v` source file. The directives are as follows:

SPI_DIVIDER_BIT_NB

This parameter defines the maximum number of bits needed for the divider. Set this parameter accordingly to the maximum system frequency and lowest serial clock frequency:

$$\text{SPI_DIVIDER_BIT_NB} = \log_2 \left[\frac{f_{sys\ max}}{f_{sclk\ min} * 2} - 1 \right]$$

Default value is 16.

SPI_MAX_CHAR

This parameter defines the maximum number of bits that can be received/transmitted in one transfer.

The default value is 32.

SPI_SS_NB

This parameter defines the number of slave select lines.

The default value is 8.