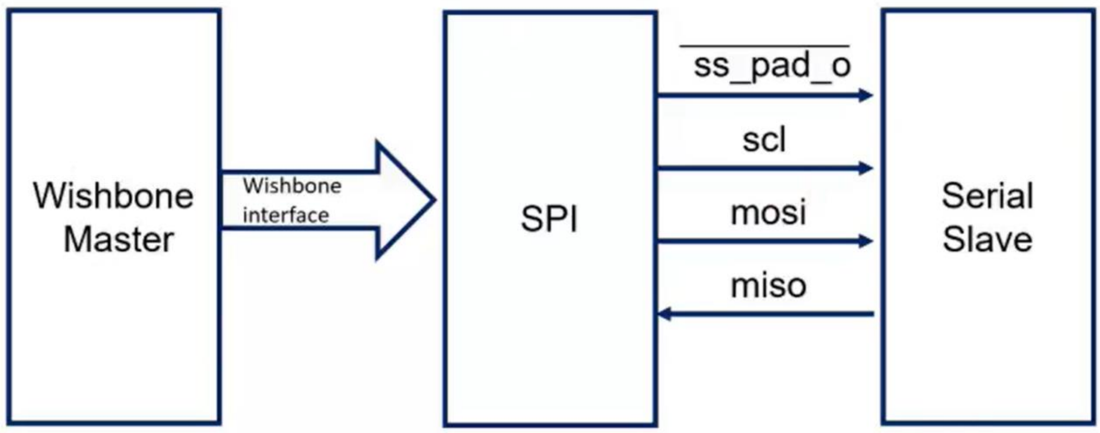
SPI Master Core Specification

SPI Protocol

- SPI stand for serial peripheral interface.
- SPI is the most widely used interfaces between microcontroller and peripheral IC's as sensors, ADC, DAC, shift registers, SRAM and others.
- SPI is a synchronous protocol that allows a master device to initiate communication within a slave.
- Data is exchanged simultaneously within the devices.

SPI Master Core Architecture



SPI Transfer Formats

- During SPI transfer, data is simultaneously transferred (shifted out serially) and received (shifted in serially)
- A serial clock line synchronizes shifting and sampling of the information on the two serial data lines.
- A slave select line allows individual selection of a slave SPI device; slave devices that are not selected so not interface with the SPI bus activities.

SPI Protocol

- ss_pad_o :- chip select/slave select (active low), which bit is 0 that slave is selected.
- scl :- serial clock
- mosi :- Master out slave in
- miso :- Master in slave out

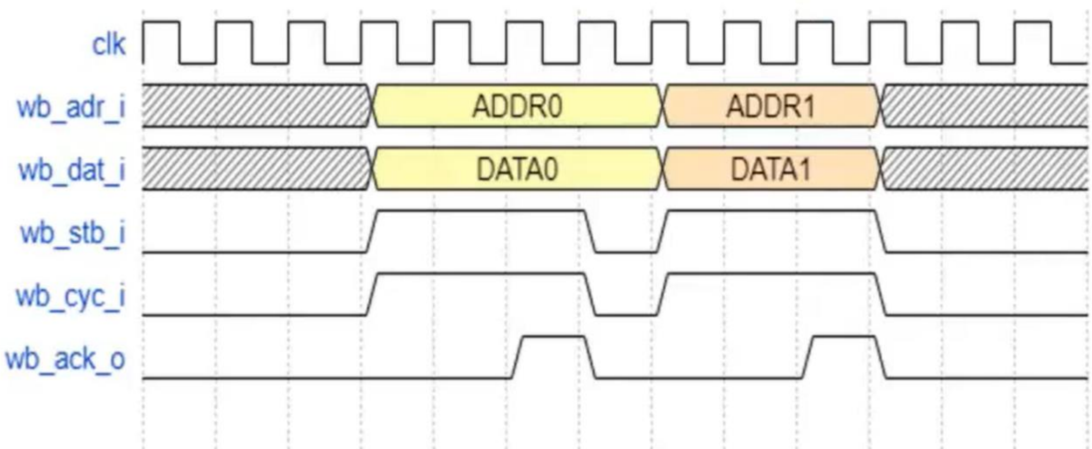
SPI Master Core Features

- Full duplex synchronous serial data.
- Variable length of transfer word up to 128 bits.
- MSB or LSB first data transfer.
- Rx and Tx on both rising and falling edge of serial clock independently.
- Up to 32 slave select lines.
- Auto slave select.
- Interrupt enable.
- Clock divider.

WISHBONE Interface Signals (Master Side)

Port	Width	Direction	Description
wb_clk_i	1	Input	Master clock
wb_rst_i	1	Input	Synchronus 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 signal
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 acknowledgement output
wb_int_o	1	Output	Interrupt signal output

WISHBONE Protocol



SPI External Connection Slave Side

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
miso_pad_i	1	Input	Master in slave out data signal

Core Registers List

Name	Address	Width	Access	Description
Rx0	0X00	32	R	Data Receive Register 0
Rx1	0X04	32	R	Data Receive Register 1
Rx2	0X08	32	R	Data Receive Register 2
Rx3	0X0c	32	R	Data Receive Register 3
Tx0	0X00	32	R/W	Data Transmit Register 0
Tx1	0X04	32	R/W	Data Transmit Register 1
Tx2	0X08	32	R/W	Data Transmit Register 2
Tx3	0X0c	32	R/W	Data Transmit Register 3
CTRL	0X10	32	R/W	Control and status register
DIVIDER	0X14	32	R/W	Clock divider circuit
SS	0X18	32	R/W	Slave Select Register

- Transmit and Receive Register

Bit#	31:0
Access	R
Name	Rx

Bit#	31:0
Access	R/W
Name	Tx

- There are 4 receive and 4 transmit registers, each register is of 32 bit size so total of 128 bits.
- Recive and transmit register share the same memory, if reception is happening at the posedge of clock transmit register will be allocated at the negedge of the same clock.

- Control and Status Register

Bit#	31:14	13	12	11	10	9	8	7	6:0
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Name	Reserved	ASS	IE	LSB	TX_NEG	RX_NEG	GO_BUSY	Reserved	CHAR_LEN

- CHAR_LEN
    - 7 bits of control register.
    - It represents the number of bits which is transmitted or received.
    - “0000000” represents 128 bit size of characters in transmission.
    - Receiving or transmission of data in register happens from 0^{th} register and 0^{th} bit and goes to next register according to the size needed.

- GO_BUSY
    - It is an active high bit.
    - It represents whether a transmission is happening or not, if the bit it 0 ot represents that the transmission is completed.

- RX_NEG and TX_NEG
    - Each is 1 bit signal.
    - Active high signal.
    - Both are complements, ie. if RX_NEG is high TX_NEG is low.
    - When RX_NEG is 1 miso will happen at the negedge of serial clock and mosi will happen at the posedge of the serial clock. Vice versa if TX_NEG is 1.

- LSB
    - Active high signal.
    - If LSB is 1 Transmission of data will happen from 0^{th} bit and recieption of data will be in 0^{th} bit of receive register.

- IE
    - Interrupt enable
    - Active high bit.
    - Interrupt signal is generated by SPI if IE bit is high, otherwise interrupt signal will be masked.
    - When GO_BUSY goes low the interrupt signal will be generated.

- ASS
    - Auto Slave Select.
    - Active high bit.
    - It ASS is high after transmission of all the data the slave will be released automatically, otherwise master want to manually do it.

- Divider Register

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

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

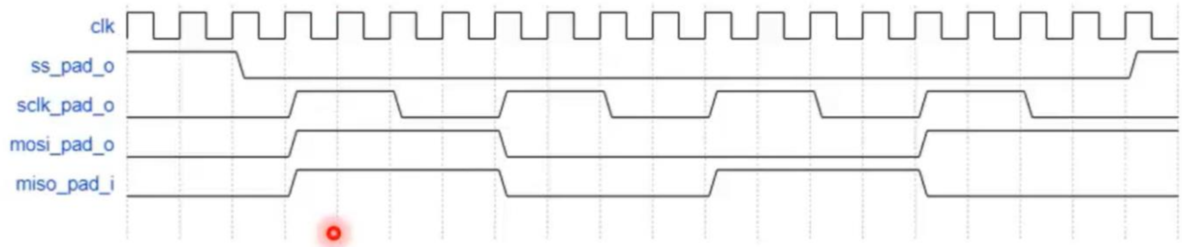
  - First 16 bits determine the clock frequency.
  - According to the divider register value and wishbone clock frequency the serial clock frequency is generated.

- Slave Select Register

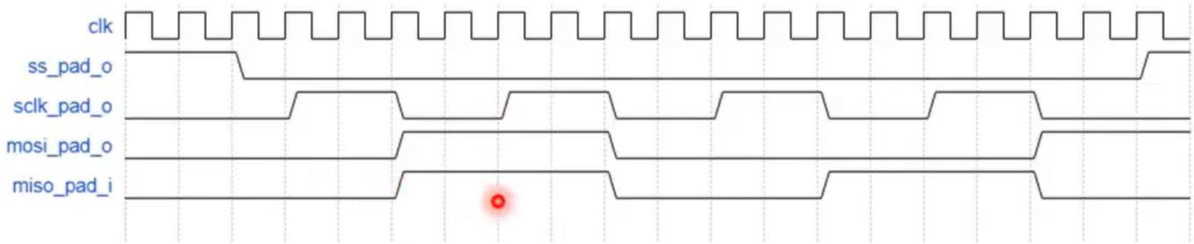
Bit#	31:8	7:0
Access	R	R/W
Name	Reserved	SS

  - It is used to select which slave is used in the transmission.
  - If 3rd bit is high means 3rd slave is selected.
  - In our design we are only using 8 slaves, so we are using only 8 bits as selection bits and remaining as reserved bits.

Data Transfer Waveform



TX_NEG =0, RX_NEG=1,CHAR_LEN =4



TX_NEG =1, RX_NEG=0,CHAR_LEN =4

Applications of SPI

- Sensors: temperature, pressure, ADC, touchscreens, video game controllers.
- Control devices: audio codecs, digital potentiometers, DAC.
- Camera lenses: Canon EF lens mount.
- Communications: Ethernet, USB, USART, CAN, IEEE 802.15.4, IEEE 0 802.11 handheld video games.
- Memory: flash and EEPRO.
- Real-time clocks.
- LCD, sometimes even for managing image data .
- SD card.

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 128 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
Rx0	0x00	32	R	Data receive register 0
Rx1	0x04	32	R	Data receive register 1
Rx2	0x08	32	R	Data receive register 2
Rx3	0x0c	32	R	Data receive register 3
Tx0	0x00	32	R/W	Data transmit register 0
Tx1	0x04	32	R/W	Data transmit register 1
Tx2	0x08	32	R/W	Data transmit register 2
Tx3	0x0c	32	R/W	Data transmit register 3
CTRL	0x10	32	R/W	Control and status register
DIVIDER	0x14	32	R/W	Clock divider register
SS	0x18	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 registers[RxX]

Bit #	31:0
Access	R
Name	Rx

Table 4: Data Receive register

Reset Value: 0x00000000

RxX

The Data Receive registers hold 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[9:3] is set to 0x08, bit RxL[7:0] holds the received data). If character length is less or equal to 32 bits, Rx1,Rx2 and Rx3 are not used, if character length is less than 64 bits, Rx2 and Rx3 are not used and so on.

NOTE: The Data Received registers are read-only registers. A Write to these registers will actually modify the Transmit registers because those registers share the same FFs.

3.3 Data transmit register [TxX]

Bit #	31:0
Access	R/W
Name	Tx

Table 5: Data Transmit register

Reset Value: 0x00000000

TxX

The Data Receive registers hold 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[9:3] is set to 0x08, the bit Tx0[7:0] will be transmitted in next transfer). If character length is less or equal to 32 bits, Tx1, Tx2 and Tx3 are not used, if character len is less than 64 bits, Tx2 and Tx3 are not used and so on.

3.4 Control and status register [CTRL]

Bit #	31:14	13	12	11	10	9	8	7	6:0
Access	R	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Name	Reserved	ASS	IE	LSB	Tx_NEG	Rx_NEG	GO_BSY	Reserved	CHAR_LEN

Table 6: Control and Status register

Reset Value: 0x00000000

ASS

If this bit is set, ss_pad_o signals are generated automatically. This means that slave select signal, which is selected in SS register is asserted by the SPI controller, when transfer is started by setting CTRL[GO_BSY] and is de-asserted after transfer is finished. If this bit is cleared, slave select signals are asserted and de-aserted by writing and clearing bits in SS register.

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

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.

CHAR_LEN
This field specifies how many bits are transmitted in one transfer. Up to 64 bits can be transmitted.
CHAR_LEN = 0x01 … 1 bit
CHAR_LEN = 0x02 … 2 bits
…
CHAR_LEN = 0x7f … 127 bits
CHAR_LEN = 0x00 … 128 bits

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

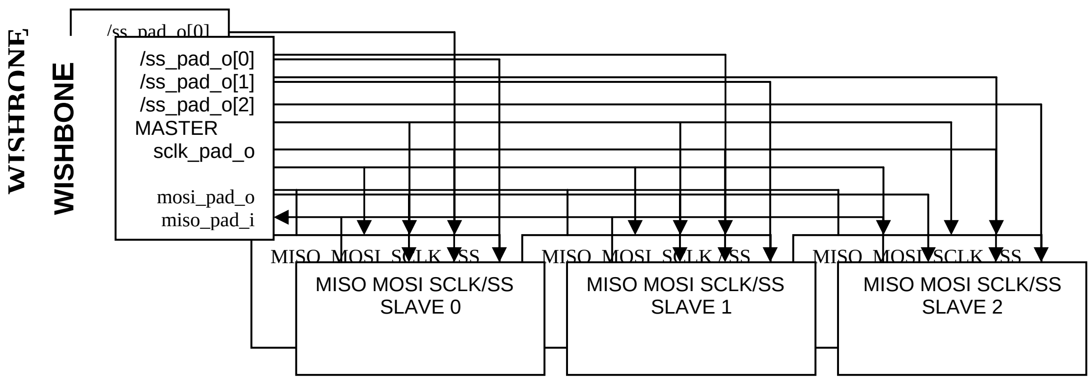
SS

If CTRL[ASS] bit is cleared, writing 1 to any bit location of this field sets the proper ss_pad_o line to an active state and writing 0 sets the line back to inactive state. If CTRL[ASS] bit is set, writing 1 to any bit location of this field will select appropriate ss_pad_o line to be automatically driven to active state for the duration of the transfer, and will be driven to inactive state for the rest of the time.

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.

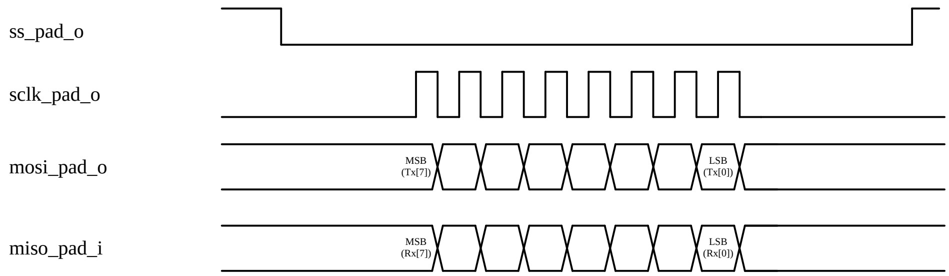


4.1 WISHBONE interface

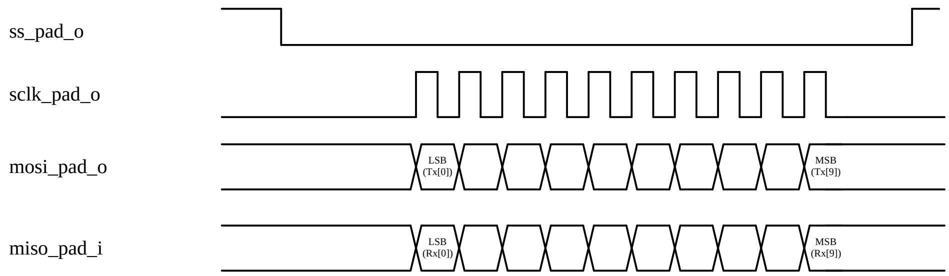
The SPI core has five 32-bit registers through the WISHBONE rev. B1 compatible interface. All accesses to SPI registers must be 32-bit (wb_sel[3:0] = 0xf). Please refer to the WISHBONE specification at
http://www.opencores.org/wishbone/specs/wbspec_b1.pdf

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 64 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 RxX and TxX 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 TxX register is executed between the transfers.



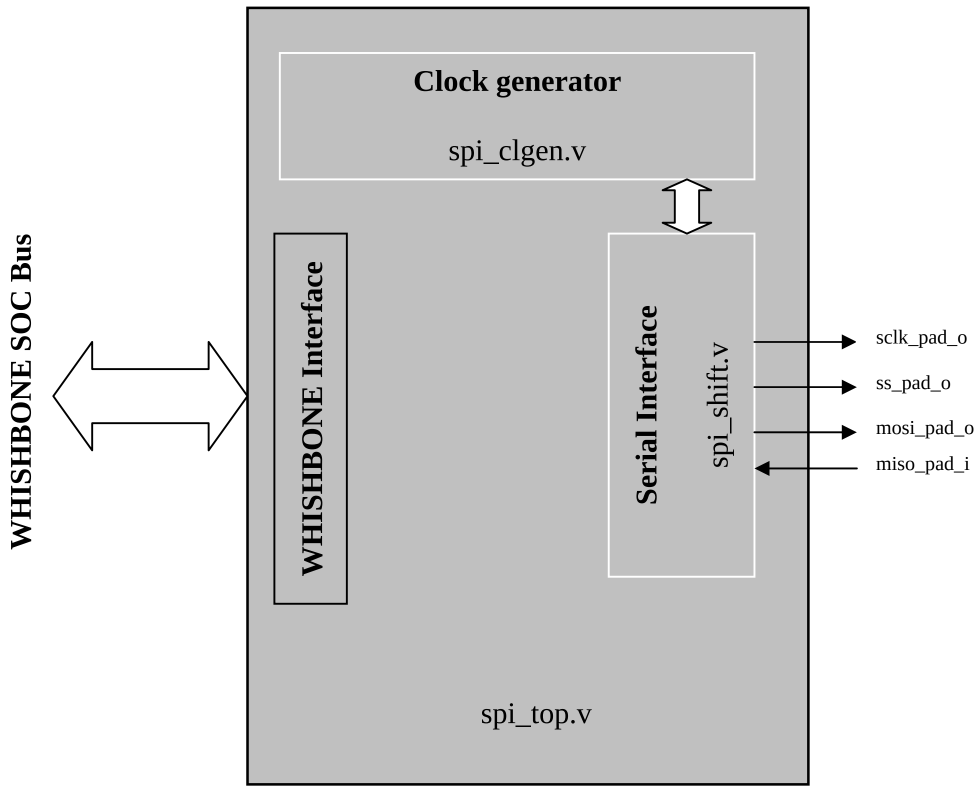
CTRL[LSB] = 0, CTRL[CHAR_LEN] = 0x08, CTRL[TX_NEG] = 1, CTRL[RX_NEG] = 0



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 configuredby 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:

$$\mathrm{SPI\_DIVIDER\_BIT\_NB} = \log_2\left[\frac{f_{sys\max}}{f_{sclk\min} * 2} - 1\right]$$

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 64.

SPI_SS_NB

This parameter defines the number of slave select lines.

The default value is 8.