SPI MASTER
Block specification

IMPORTANT NOTICE

Table of contents

Document history

Version	Date	Approved by	Created by	Description
1.0	07/16/08	Mariana Avram	Tudor Durnea	Document created. Changes from initial document: - added figure with internal architecture - presented two additional transfer clocking phases - presented interrupt timing
1.1	02/22/19		Alex Perieteanu	Added WB_ERR behaviour

1 Introduction

1.1 Scope

This document contains the functional specification of SPI Master block.

1.2 Abbreviations

DUT	Device Under Test
TBD	To Be Done/Defined
SPI	Serial Peripheral Interface – serial synchronous data link protocol developed by Motorola. The protocol didn't achieve standard level.
Microwire	Restricted set of SPI – trademark of National Semiconductors. Communication is half duplex and uses only one out of four clock phase sets.
WB	WISHBONE
WB Master	WB Master device, it starts all transfers on WB interface
SPI slave	SPI slave device, needs a Slave Select signal to participate in transfers
SPI Master	This device, will be used interchangeably with the DUT notation

1.3 Documentation

1. Wishbone System-On-Chip Interconnection Architecture for Portable IP Cores, Rev. 1.B.;
2. SPI introduction on http://www.mct.net/faq/spi.html
3. Microwire Serial Interface, National Semiconductors application note AN-452.

2. SPI Master Main Functionality and Features

This document provides specifications for the SPI (Serial Peripheral Interface) Master core. 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)

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

2.1. SPI Master Initialization

Upon reset the core performs the following tasks:

- The Rx0/Tx0, Rx1/Tx1, Rx2/Tx2, Rx3/Tx3 registers for data reception / transmission are cleared.
- The CTRL control register is cleared. This implies manual slave select operations, interrupt disabled, any transfer stopped. The serial word length is set to 128 bits, to be transmitted / received on the rising edge of SCLK, MSB first.
- The DIVIDER register is cleared to 0xFF_FF. The frequency of SCLK will be attained from the value of the WB clock divided by 131072.
- The SS (Slave Select) register is cleared to 0, no SPI slave will transfer data.

For proper operation, perform the following:

- Set the Divider register to the desired value supported by the SPI slaves.
  $f_{sclk}=f_{wb\_clk} / (DIVIDER + 1) *2$
- Set the parameters of the SPI transfers: LSB / MSB convention and output / sample clock phase convention.
- Enable the interrupt.

2.2. SPI Master Operation Flow

The SPI Master is considered configured, serial clock frequency, LSB / MSB and clock phase conventions. The following steps must be performed before starting the SPI transfers:

- WB master writes SS register to select the slaves for SPI transfers. If the CTRL[ASS] bit is 1 the “ss_pad_o” lines remain inactive. If CTRL[ASS] bit is 0 the “ss_pad_o” lines will be loaded with the value of SS register.
- WB master writes the CTRL register: word length and automatic slave select (ASS). If ASS bit is 1 the “ss_pad_o” lines will be asserted when the SPI transfer starts and de-asserted when the transfer ends.

The SPI transfer is started by writing the CTRL register bit GO_BSY with 1. This must be done after all registers (including CTRL) are configured (Ex: CTRL = 0x2A00 and CTRL = 0x2B00). Writing to any register during a SPI transfer will have no effect, the registers will retain initial data.

The SPI transfer includes the following actions:

- The DUT outputs the serial clock on “sclk_pad_o” for the duration of the transfer - “CHAR_LEN” number of serial clock periods;
- The DUT outputs the serial data on “mosi_pad_o”;
- The SPI Slave outputs the serial data on “miso_pad_i” line;
- The slave select line for current SPI slave remains low for the duration of the transfer.
- The DUT signalizes the elapse of “CHAR_LEN” time duration with an interrupt on “wb_int_o”. The serial clock “sclk_pad_o” is kept low and if CTRL[ASS] bit is 1 the slave select line will be de-asserted. The CTRL[GO_BSY] bit is automatically cleared.

For timing waveforms regarding SPI transfers see Chapter 5.2.

The DUT is implemented according to SPI specifications and as so, its transmit / receive registers with the same index are implemented on the same flip-flops. The data bits written by WB master in the common register are shifted in the SPI transfer according to LSB / MSB convention and sent on “mosi_pad_o” line. The positions cleared by shifting for transmission are filled with the bits sampled on the “miso_pad_i” line. The received bits are shifted according to the same LSB / MSB convention.

At SPI transfer end the “CHAR_LEN” data bits written by WB master in Tx0/1/2.. registers are completely replaced with “CHAR_LEN” bits from SPI slave. The WB master can read the common registers (WB read operations at RX0/1/2.. addresses) to load the received data after the interrupt line “wb_int_o” line goes high.

The SPI transfers are full duplex all the times. That is the DUT will always output data on “mosi_pad_o” line and the slave will put data on “miso_pad_i” during all transfers. Single DUT writes or reads on SPI interfaces (one SPI data line inactive) are not possible. The SPI slave that is selected by a high level on its “ss_pad_o” line will always receive and transmit data during a transfer. The SPI Master can not broadcast data to several SPI slave as this will result in a data collision on “miso_pad_i” line.

3. SPI Master Architecture

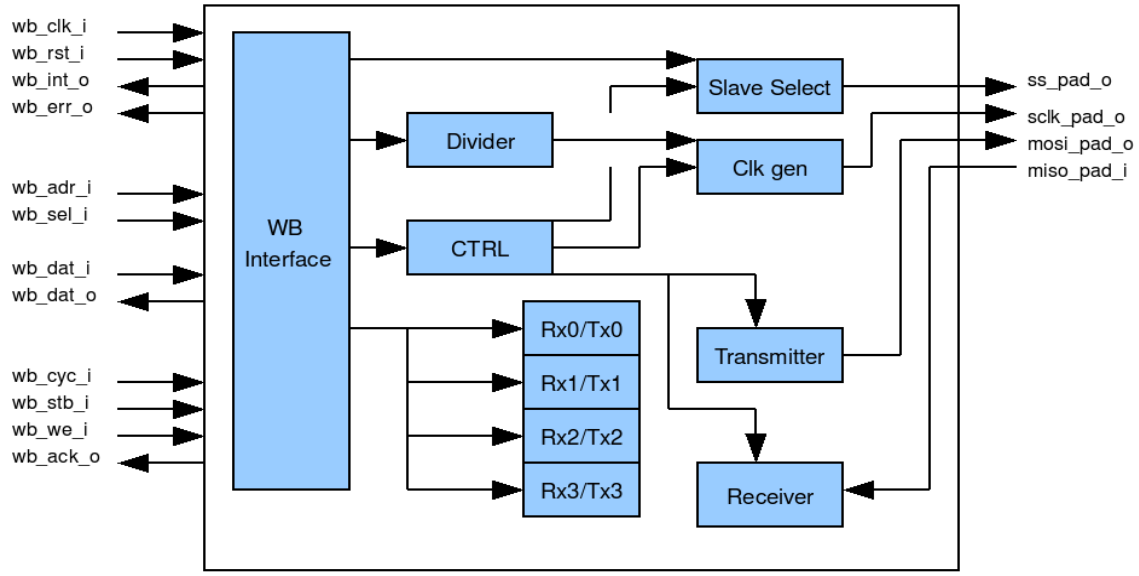

Fig 1. SPI Master architecture

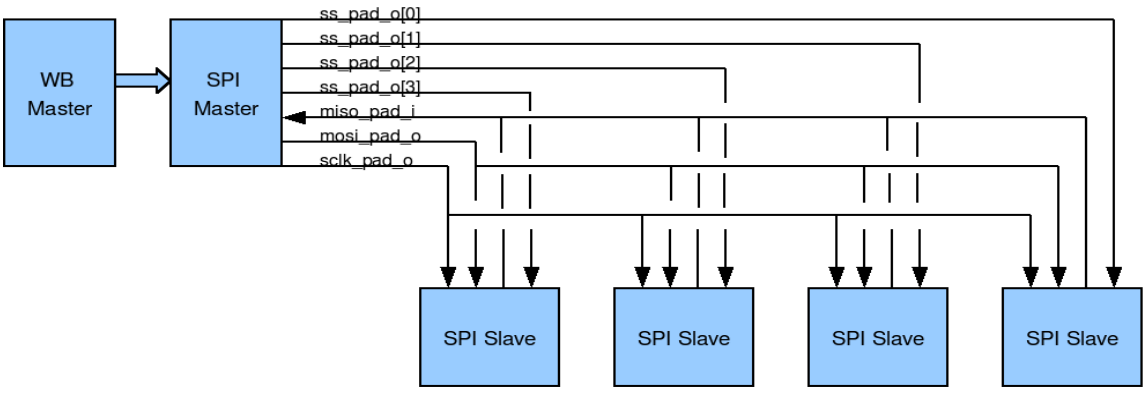

Fig 2. SPI Master Connection Architecture

4. SPI Master Interfaces

4.1. 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, active high

4.2. SPI Interface signals

Port	Width	Direction	Description
ss_pad_o	8	Output	Slave select output signals, active low
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

5. SPI Master Timing Waveforms

5.1. Wishbone Interface Timing Waveforms

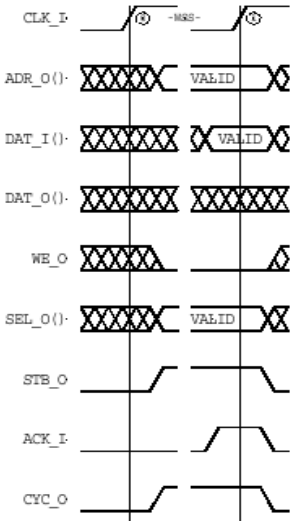

Fig 3. Wishbone Single READ cycle

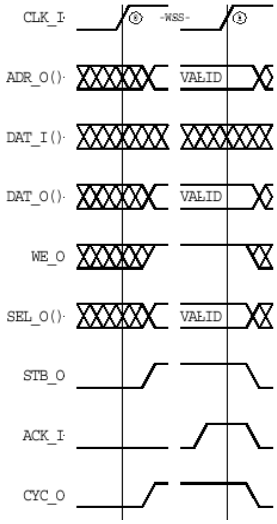

Fig 4. Wishbone Single WRITE cycle

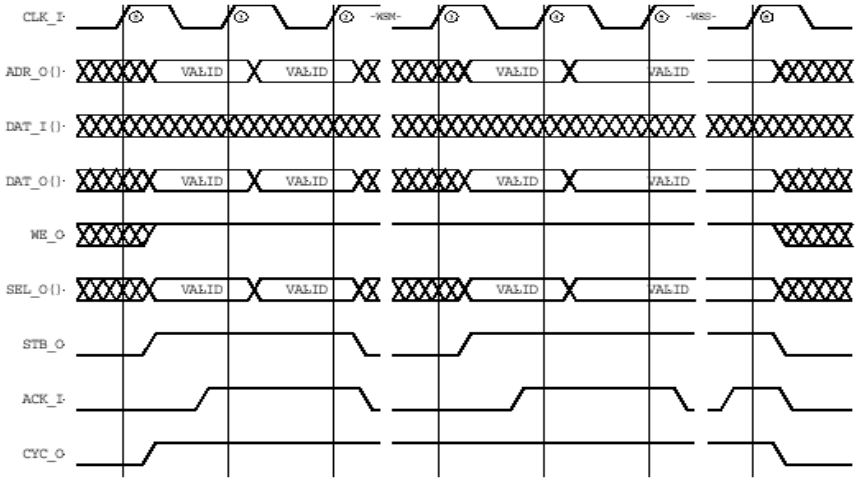

Fig 5. WISHBONE BLOCK WRITE cycle

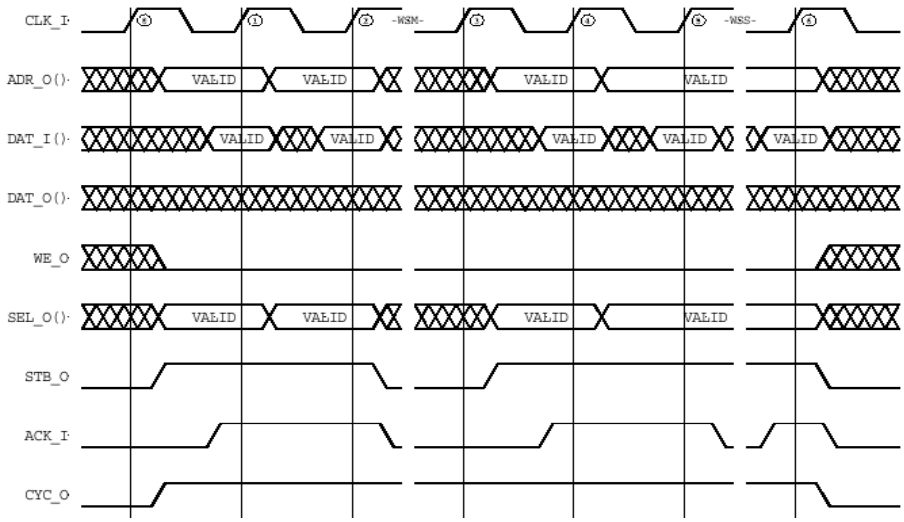

Fig 6. Wishbone BLOCK READ cycle

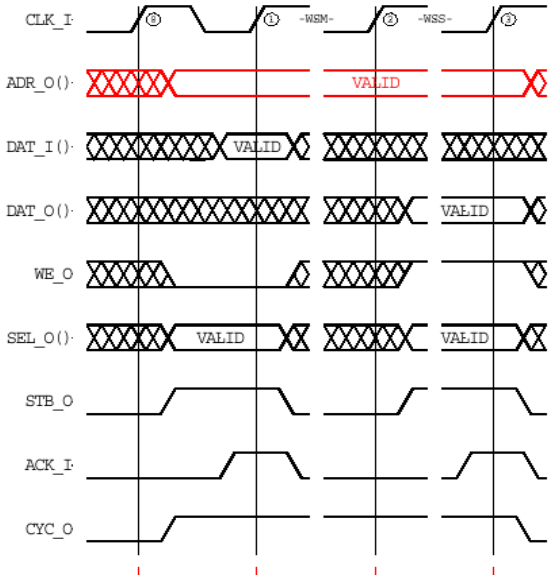

Fig 7. Wishbone RMW cycle

The chapter herein describes the strict usage of WB signals for the SPI Master implementation. For a complete description of WISHBONE protocol see chapter 1.3 . WISHBONE System-on-Chip Interconnection Architecture for Portable IP Cores, Rev. 1.B.

1. WB Master signals a WB transfer with wb_cyc_i high. The signal wb_stb_i is used to mark intervals where the WB Master has placed data on wb_dat_i for WRITE operations or the intervals where the DUT has outputted data on wb_dat_o in READ operation.
2. In WRITE operations the WB Master puts data on wb_adr_i, wb_dat_i and wb_sel_i. The WB Master must then assert wb_stb_i to indicate to SPI Master that it can sample the data. The SPI Master loads the bytes from wb_dat_i which have the corresponding bits in wb_sel_i set to 1. For wb_sel_i equal to 4'b0001 the DUT will load only the LSByte from wb_dat_i. SPI Master responds with wb_ack_o high to signal it has loaded the data. After the falling edge of wb_ack_o, the WB Master can de-assert wb_stb_i and remove data and address from buses.
3. In READ operations the WB Master puts data on wb_adr_i and wb_sel_i. The WB Master must then assert wb_stb_i to indicate to the SPI Master that it can samplee the address. The DUT responds with wb_ack_o high to signal it has outputted the required register content on wb_dat_o. The DUT will place the content of addressed register on wb_dat_o without any constraint from wb_sel_i. That is the DUT places all 32 bits from Rx0 register even though wb_sel_i is 4'b0100. The WB Master can de-assert wb_stb_i and remove the address after the falling edge of wb_ack_o.
4. READ and WRITE operations on WB bus can be performed independently or in groups:
   - An wb_cyc_i pulse with one wb_stb_i pulse inside defines a WB SINGLE cycle.
   - An wb_cyc_i pulse with several wb_stb_i pulses inside and wb_we_i with the same level defines an WB BLOCK cycles. WB BLOCK READ / WRITE cycles allow the user to read / configure several SPI Master registers on a single access .
   - A special WB cycle is the RMW one. It consists of a WB READ followed by a WB WRITE cycle. The DUT can be accessed by WB BLOCK and RMW cycles with different address for each atomic WB operation.
5. A clock cycle with wb_cyc_i high, wb_stb_i high and wb_ack_o low is called SLAVE WAIT state and it has no effect. A clock cycle with wb_cyc_i high and wb_stb_i low is called a MASTER WAIT state and has no effect.
6. When a WB access targets an address which is not mapped, the SPI_MASTER block/WB slave will accept it and raise wb_err_o together with wb_ack_o, for the same duration as wb_ack_o.

5.2. SPI Interface Timing Waveforms

The SPI Master is configurable on the clock phases used to output / sample the data bits on “mosi_pad_o” and “miso_pad_i” lines. The clock phases are controlled by the bits Tx_NEG and Rx_NEG in CTRL register. Both SPI Master and Slave must sample the data bit when stable, at the middle of its time period.

The SPI specifications for clock phases are:

Rx_NEG	***Tx_NEG***	***SPI Master***
0	1	Toggles “mosi_pad_o” on the falling edge of “sclk_pad_o” Samples “miso_pad_i” on the rising edge of “sclk_pad_o”
1	0	Toggles “mosi_pad_o” on the rising edge of “sclk_pad_o” Samples “miso_pad_i” on the falling edge of “sclk_pad_o”
Rx_NEG	***Tx_NEG***	***SPI Slave***
0	1	Toggles “miso_pad_i” on the falling edge of “sclk_pad_o” Samples “mosi_pad_o” on the rising edge of “sclk_pad_o”
1	0	Toggles “miso_pad_i” on the rising edge of “sclk_pad_o” Samples “mosi_pad_o” on the falling edge of “sclk_pad_o”

The SPI Master has 2 additional feature modes for clock phases, that are not SPI mandatory.

Rx_NEG	***Tx_NEG***	***SPI Master***
0	0	Toggles “mosi_pad_o” on the rising edge of “sclk_pad_o” Samples “miso_pad_i” on the rising edge of “sclk_pad_o”
1	1	Toggles “mosi_pad_o” on the falling edge of “sclk_pad_o” Samples “miso_pad_i” on the falling edge of “sclk_pad_o”
Rx_NEG	***Tx_NEG***	***SPI Slave***
0	0	Toggles “miso_pad_i” on the falling edge of “sclk_pad_o” Samples “mosi_pad_o” on the falling edge of “sclk_pad_o”
1	1	Toggles “miso_pad_i” on the rising edge of “sclk_pad_o” Samples “mosi_pad_o” on the rising edge of “sclk_pad_o”

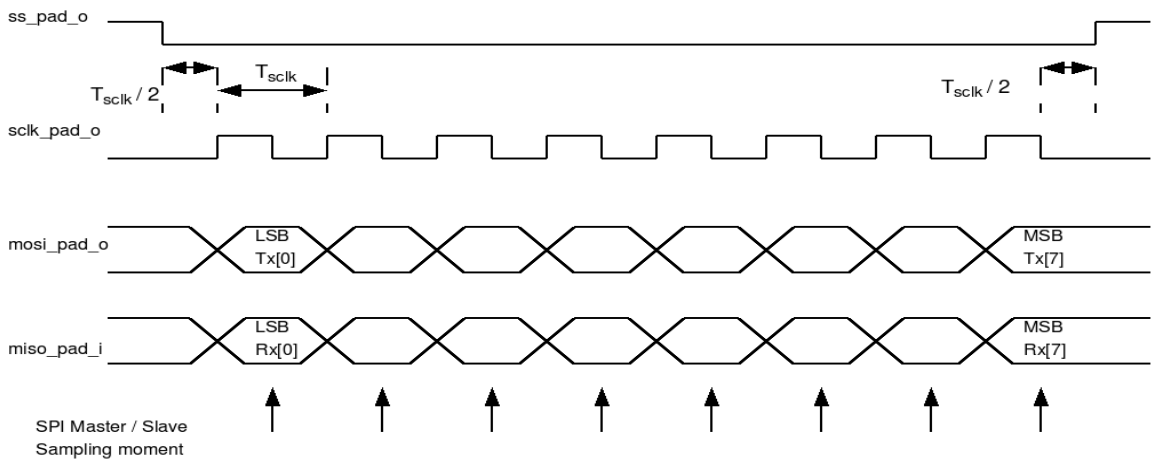

Fig. 8 SPI transfer with CTRL[LSB] = 1, CTRL[CHAR_LEN] = 0x08,
CTRL[TX_NEG] = 0, CTRL[RX_NEG] = 1

The first SPI clock phase mode is selected by writing Tx_NEG = 0 and Rx_NEG = 1 in CTRL register. Both SPI Slave and Master change the data bits on rising edge og sclk_pad_o and sample the data line on the falling edge of sclk_pad_o. Data toggle starts at the first rising edge of sclk_pad_o encountered.

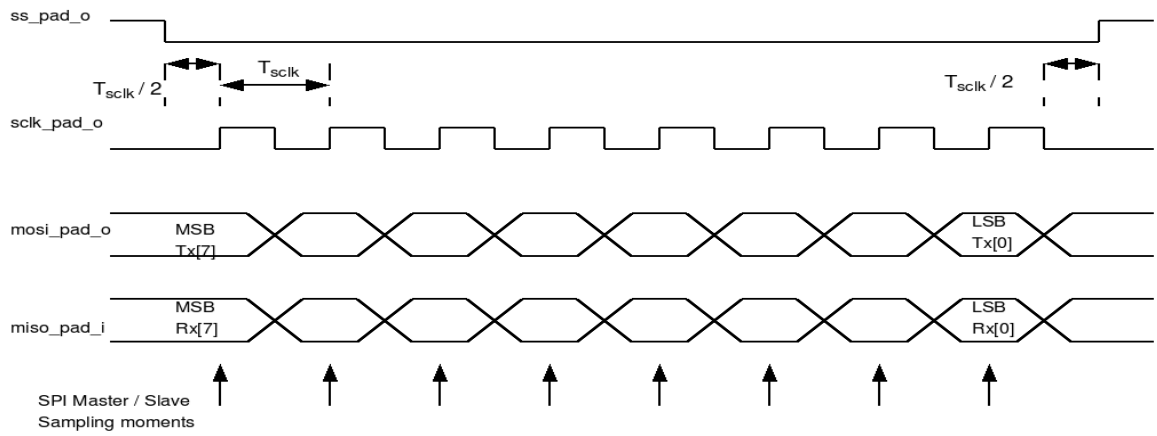

Fig. 9 SPI transfer with CTRL[LSB] = 0, CTRL[CHAR_LEN] = 0x08,
CTRL[TX_NEG] = 1, CTRL[RX_NEG] = 0

The second SPI clock phase mode is selected by writing Tx_NEG = 1 and Rx_NEG = 0 bits in CTRL register. Both SPI Master and Slave change the data bits on the falling edge of sclk_pad_o. The DUT puts its first data bit on “mosi_pad_o” line at the falling edge of “ss_pad_o” line. Both SPI Master and Slave sample the data lines on the rising edge of “sclk_pad_o”.

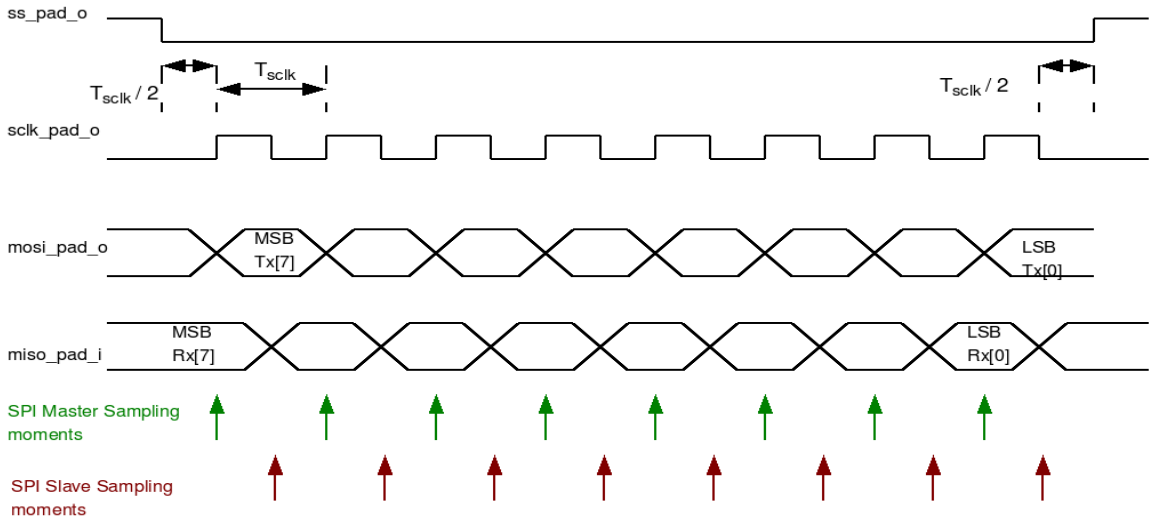

Fig. 9 SPI transfer with CTRL[LSB] = 0, CTRL[CHAR_LEN] = 0x08,

CTRL[TX_NEG] = 0, CTRL[RX_NEG] = 0

The first SPI Master clock mode feature is when both Tx_NEG and Rx_NEG are equal to 0. That is the SPI Master both toggles the “mosi_pad_o” and samples the “miso_pad_i” line on the rising edge of “sclk_pad_o”. Because of sampling requirements, the SPI Slave must put data on “miso_pad_i” at the falling edge of “sclk_pad_o” and sample “mosi_pad_o” at the same moment. The SPI Slave must put the first data bit on “miso_pad_i” line at the falling edge of “ss_pad_o”. SPI Master toggles “mosi_pad_o” at the first rising edge of “sclk_pad_o”. The last data bit is placed on the lines at the last clock rising edge for SPI Master and at the last clock falling edge for SPI Slave.

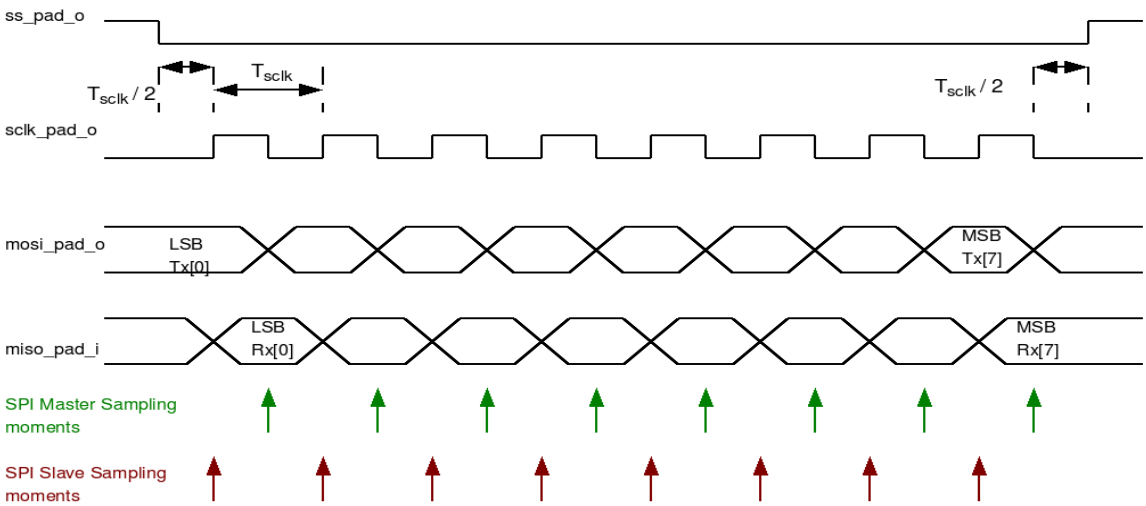

Fig. 10 SPI transfer with CTRL[LSB] = 1, CTRL[CHAR_LEN] = 0x08,
CTRL[TX_NEG] = 1, CTRL[RX_NEG] = 1

The second SPI Master clock mode feature is when both Tx_NEG and Rx_NEG are equal to 1. That is the SPI Master both toggles the “mosi_pad_o” and samples the “miso_pad_i” line on the falling edge of “sclk_pad_o”. Because of sampling requirements, the SPI Slave must put data on “miso_pad_i” at the rising edge of “sclk_pad_o” and sample “mosi_pad_o” at the same moment. The SPI Master must put the first data bit on “mosi_pad_o” line at the falling edge of “ss_pad_o”. SPI Slave toggles “miso_pad_i” at the first rising edge of “sclk_pad_o”. The last dat bit is placed on the lines at the last clock falling edge for SPI Master and at the last clock rising edge for SPI Slave.

5.3. WB_INT_O Timing Waveforms

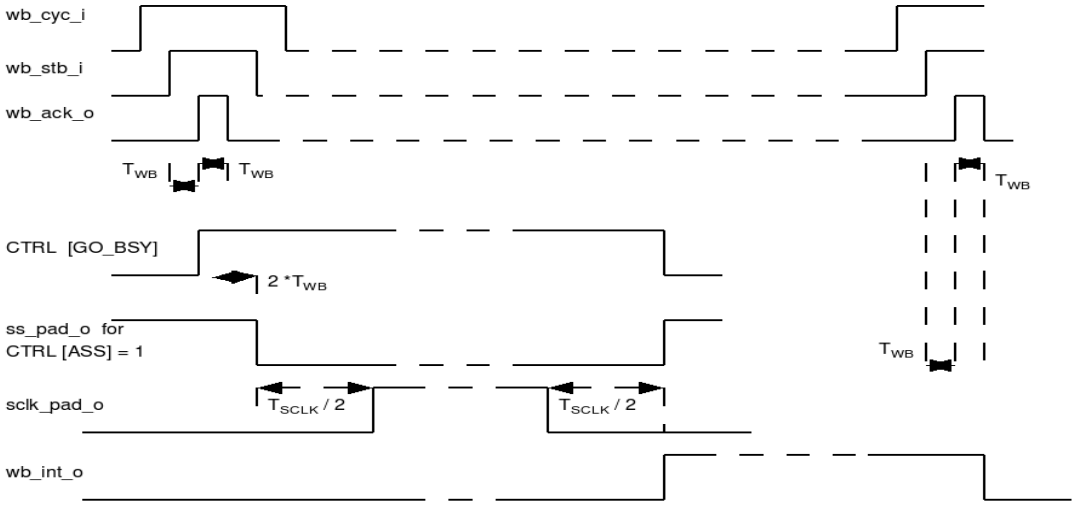

Fig. 11 Interrupt timing from transfer start to interrupt de-assertion

After the WB Master has configured the SPI Master, it must write the CTRL [GO_BSY] bit with 1 to start the SPI transfer. We consider that the SPI Master has the interrupt enabled and it selects the SPI slaves automatically – CTRL [IE] = 1 and CTRL [ASS] = 1.

The ss_pad_o line will go low after 2 WB clock periods from the rising edge of wb_ack_o for the CTRL write operations. After half of SPI clock period, the sclk_pad_o will rise and start to toggle. The sclk_pad_o will generate CHAR_LEN pulses. After half of SPI clock period from the last falling edge of sclk_pad_o the ss_pad_o will go high de-selecting the slave. The bit CTRL [GO_BSY] is cleared in same moment and the line “wb_int_o” is asserted, signalizing to the WB Master that the SPI transfer has ended. The “wb_int_o” line remains high until the first WB read / write operation.

6. SPI Master Registers

Register 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

All registers are 32-bit wide. In order to write the entire location all wb_sel_i bits should be active.1. Each register byte is considered a specific address, that Rx[15:8] address is 0x01 and DIVIDER [15:8] address is 0x15.

This is because each register byte can be written / read independently by WB interface. The WB Master can transfer from 8 bits to 32 bits on a single WB operation, which implies a single address for all register's bytes. So all bytes of a register have a unique address, but they can be accessed independently using "wb_sel" signals. A high value on a "wb_sel" line means that the corresponding register byte will be written / read.

As so, the 2 LSBits of the address bus are don't care for SPI_Master, only the 3 MSBits are taken into account. All address values in [0x00..0x1B] are valid addresses.

6.1. Data Receive Registers [RxX]

This register allows enabling and disabling interrupt generation by the UART.

Bit #	Access	Description
31:0	R	Receiver register used as a shift register loaded by miso_pad_i line

Reset Value: 32'h0000_0000

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[6:0] is set to 0x08, bits Rx0[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.

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.

6.2. Data Transmit Registers [TxX]

This register allows enabling and disabling interrupt generation by the UART.

Bit #	Access	Description
31:0	R/W	Transmitter register used as a shift register sourcing to mosi_pad_o line

Reset Value: 32'h0000_0000

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[6:0] is set to 0x08, the bits 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 length is less than 64 bits, Tx2 and Tx3 are not used and so on.

6.3. Control and Status Register [CTRL]

Bit #	Access	Description
31:14	R	Reserved, writing to them has no effect
13	R/W	ASS – Automatic Slave Select
12	R/W	IE – Interrupt Enable
11	R/W	LSB – Least Significant Bit first to transmit
10	R/W	Tx_NEG – SPI transmission on negative front of clock
9	R/W	Rx_NEG – SPI sample on negative front of clock
8	R/W	GO_BSY – Start SPI transfer / SPI transfer in progress
7	R	Reserved, writing to it has no effect
6:0	R/W	CHAR_LEN – number of bits to be sent / received during a SPI transfer

Reset Value: 32'h0000_0000

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 de-asserted 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.

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 128 bits can be transmitted.
CHAR_LEN = 0x01 … 1 bit
CHAR_LEN = 0x02 … 2 bits
…
CHAR_LEN = 0x7f … 127 bits
CHAR_LEN = 0x00 … 128 bits

6.4. Divider Register [DIVIDER]

This register allows enabling and disabling interrupt generation by the UART.

Bit #	*Access*	*Description*
31:16	R	Reserved, writing to them has no effect
15:0	R/W	DIVIDER

Reset Value: 32'h0000_FFFF

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:

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

6.5. Slave Select Register [SS]

This register allows enabling and disabling interrupt generation by the UART.

Bit #	Access	Description
31:8	R	Reserved, writing to them has no effect
7:0	R/W	SS

Reset Value: 32'h0000_0000

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.