*Technion*

*Electrical Engineering Department*

VLSI Laboratory

SPI Master and Slave

Documentation

**Version**: 1.0

**Last update**: 24.10.2011

**Written By**: Beeri Schreiber and Omer Shaked

Table of content

[Table of Changes 2](#_Toc307305429)

[1 Scope 2](#_Toc307305430)

[2 Abbreviations 2](#_Toc307305431)

[3 SPI Master 2](#_Toc307305432)

[3.1 SPI Master Pinout 3](#_Toc307305433)

[3.2 SPI Master Generic Parameters 4](#_Toc307305434)

[3.3 Resources 4](#_Toc307305435)

[3.4 SPI Master Waveforms 5](#_Toc307305436)

[3.4.1 Burst 5](#_Toc307305437)

[3.4.2 Start and End of Burst 6](#_Toc307305438)

[3.5 Performed Simulations for SPI Master 7](#_Toc307305439)

[4 SPI Slave 8](#_Toc307305440)

[5 Appendix A – SPI Protocol 9](#_Toc307305441)

[5.1 Interface 9](#_Toc307305442)

[5.2 Data Transmission 9](#_Toc307305443)

[5.3 Clock Polarity and Phase 9](#_Toc307305444)

Table of Tables

[Table 1 – Table of Changes 2](#_Toc307301168)

[Table 2 – Global Nets Pinout 4](#_Toc307301169)

[Table 3 – SPI Master Generic Parameters 4](#_Toc307301170)

Tables of Figures

[Figure 1 – SPI Master Pinout 3](#_Toc307305423)

[Figure 2 – Burst Waveform 5](#_Toc307305424)

[Figure 3 – Start and End of Burst 6](#_Toc307305425)

[Figure 4 – SPI Protocol Diagram 9](#_Toc307305426)

# Table of Changes

|  |  |  |
| --- | --- | --- |
| Version | Date of Change | Description |
| 1.0 | 01.10.2011 | Creation of documentation |

Table 1 – Table of Changes

# Scope

This document aims to describe the working method of the SPI Master and Slave Cores.

# Abbreviations

1. SPI – Serial Peripheral Interface
2. FIFO – First In First Out
3. MOSI – Master Out Slave In
4. MISO – Master In Slave Out

# SPI Master

SPI Master is consumed of the following interfaces:

* Clock and Reset interfaces
* FIFO Interface. Data from FIFO will be transmitted through the SPI\_MOSI line
* Registers Interface, to control internal SPI Core Configurations
* SPI Interface, according to SPI Protocol
* Miscellaneous – SPI Address Slave ; Busy signal

As long as the input FIFO is not empty, data will be transmitted to the relevant SPI Slave, which is determined by SPI slave address.

When transaction is active, 'BUSY' signal will be asserted ('1').

The following registers may be modified during normal operation, when there is no active transmission:

* Configuration Register - to configure CPOL and CPHA:
  + Bit 0 - CPHA
  + Bit 1 - CPOL
* Clock Divide Register - to configure system clock divide factor to the SPI clock. i.e: Suppose divide factor is 2, then each two system clock cycles (4 system clocks event), SPI\_CLK will change its polarity.

**Requirements**:

* Reset deactivation MUST be synchronized to the system clock's rising edge!
* Reset activation may be asynchronous to the clock.
* FIFO should assert *FIFO\_DIN\_VALID* within one clock from *FIFO\_REQ\_DATA*.

## SPI Master Pinout

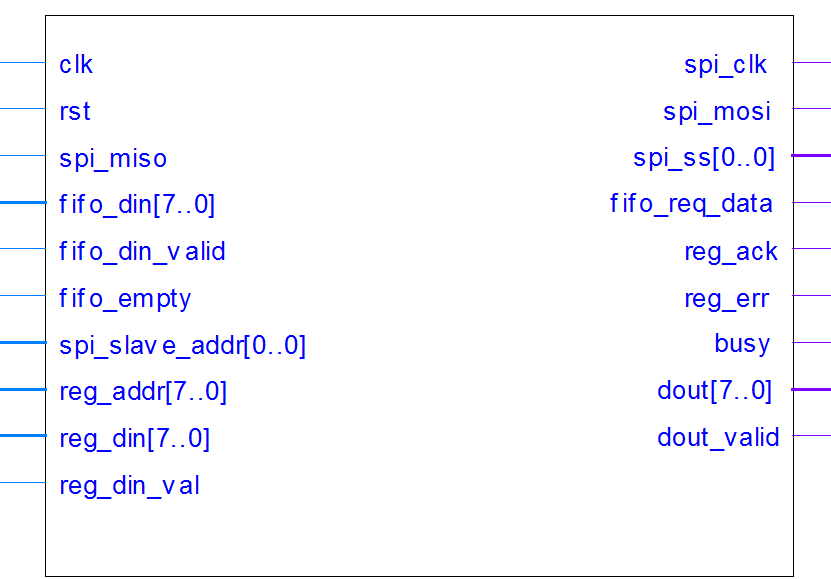


Figure 1 – SPI Master Pinout

| Pin Name | Direction | Description |
| --- | --- | --- |
| Clk | In | System Clock |
| Rst | In | Reset (active according to Reset Polarity Generic) |
| Spi\_clk | Out | SPI Output Clock |
| Spi\_mosi | Out | SPI Master Out Slave In |
| Spi\_miso | In | SPI Master In Slave Out |
| Spi\_ss [] | Out | SPI Slave Select |
| Fifo\_req\_data | Out | Request for data from FIFO |
| Fifo\_din [] | In | Input data from FIFO |
| Fifo\_din\_valid | In | '1' when input data from FIFO is valid (wil be '1' for one clock cycle, with delay of one clock cycle from *fifo\_req\_data*) |
| Fifo\_empty | In | '1' when FIFO is empty |
| Spi\_slave\_addr [] | In | SPI Addressed Slave. Feed with the NUMBER of the slave (i.e, for slave number 3, set value of "11") |
| Reg\_addr [] | In | Address to Internal Configurations Registers |
| Reg\_din [] | In | Input data to Registers |
| Reg\_din\_val | In | Data to registers is valid (should be asserted for one clock cycle) |
| Reg\_ack | Out | Register Data has been acknowledged |
| Reg\_err | Out | Error while writing data to registers |
| Busy | Out | SPI Master is in the middle of a transaction |
| Dout [] | Out | Output data, which has been received from SPI Slave |
| Dout\_valid | Out | Asserted when *dout* is valid |

Table 2 – Global Nets Pinout

## SPI Master Generic Parameters

| Generic Parameter | Default Value | Description |
| --- | --- | --- |
| reset\_polartiy\_g | '0' | Reset active in this polarity |
| ss\_polarity\_g | '0' | Slave select polarity is active at this value |
| data\_width\_g | 8 | Input / Output parallel data |
| bits\_of\_slaves\_g | 1 | Number of bits for SPI\_SS |
| reg\_width\_g | 8 | Number of bits in SPI Configurations Register |
| dval\_conf\_reg\_g | 0 | Default (initial) value of Configuration Register |
| dval\_clk\_reg\_g | 2 | Default (initial) value of Clock Divide Register (Divide system clock by 2 is the minimum) |
| reg\_addr\_width\_g | 8 | Registers Configuration Address Width |
| reg\_din\_width\_g | 8 | Registers Configuration Input Data Width |
| first\_dat\_lsb | true | TRUE: Transmit and Receive LSB first. FALSE - MSB first |

Table 3 – SPI Master Generic Parameters

## Resources

Required resources, when synthesizing, using Synplify, for Altera's Cyclon II FPGA:

* Total combinational functions:  102
* Logic element usage by number of inputs:
  + 4 input functions: 46
  + 3 input functions: 29
  + [=2 input functions: 27
* Logic elements by mode:
  + Normal mode:             85
  + Arithmetic mode: 17
* Total registers: 81
* I/O pins: 47

**Maximum Working Frequency**: 269MHz

## SPI Master Waveforms

### Burst

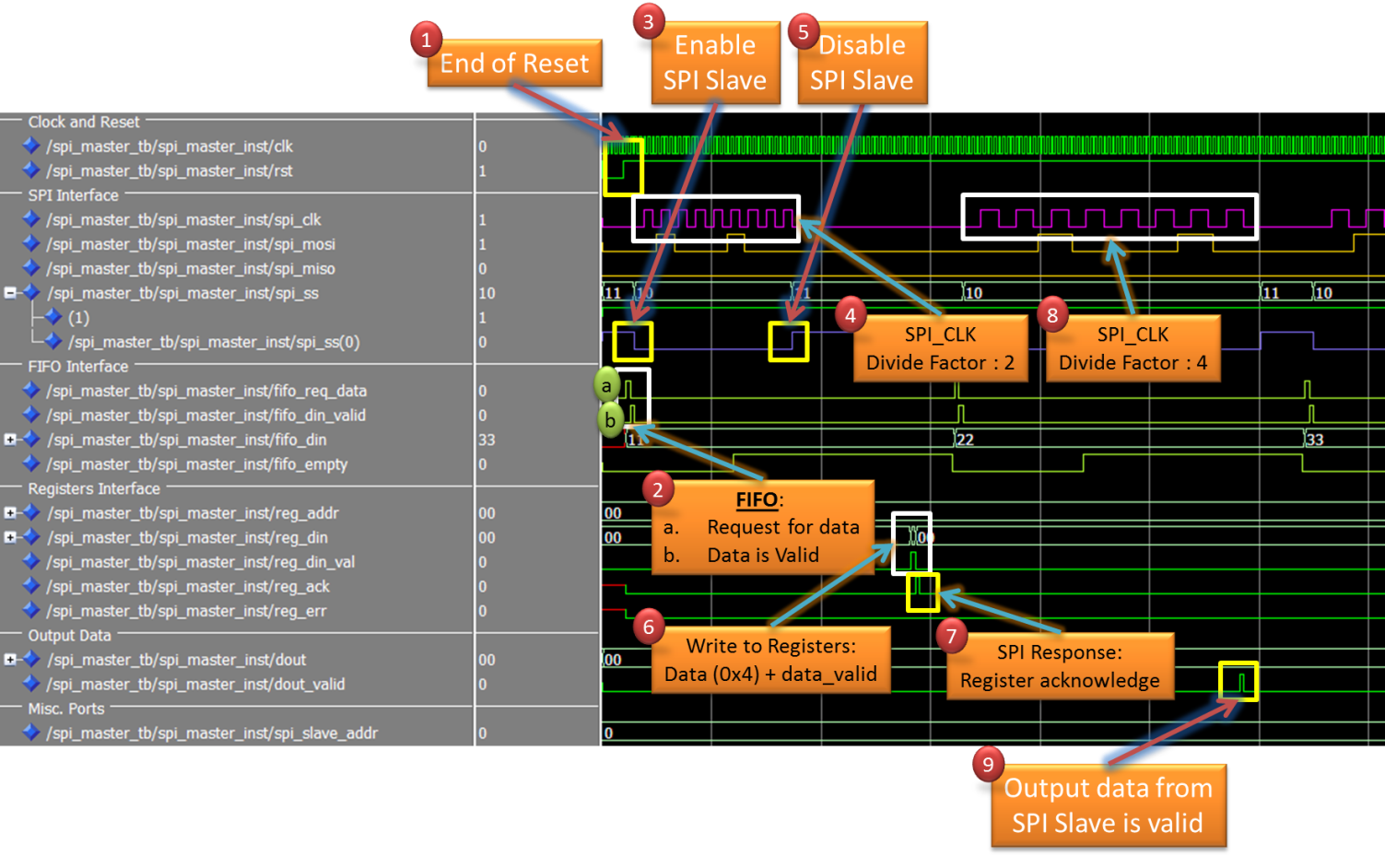


Figure 2 – Burst Waveform

1. End of System Reset
2. FIFO Interface:
   1. SPI Master requests for data from FIFO
   2. FIFO confirms that output data is valid
3. SPI Slave Select for Slave [0] is asserted ('0')
4. SPI Clock is generated. In this waveform, the divide factor, from whole clock cycle, is 2
5. End of first burst – negate ('1') SPI\_SS
6. Configure SPI Master: Change clock divide factor to 4
7. SPI Master response – reg\_ack is asserted
8. SPI Clock is generated, this time with divide factor of 4
9. Data from SPI Slave has been received, and is now valid as parallel data from SPI Master

### Start and End of Burst

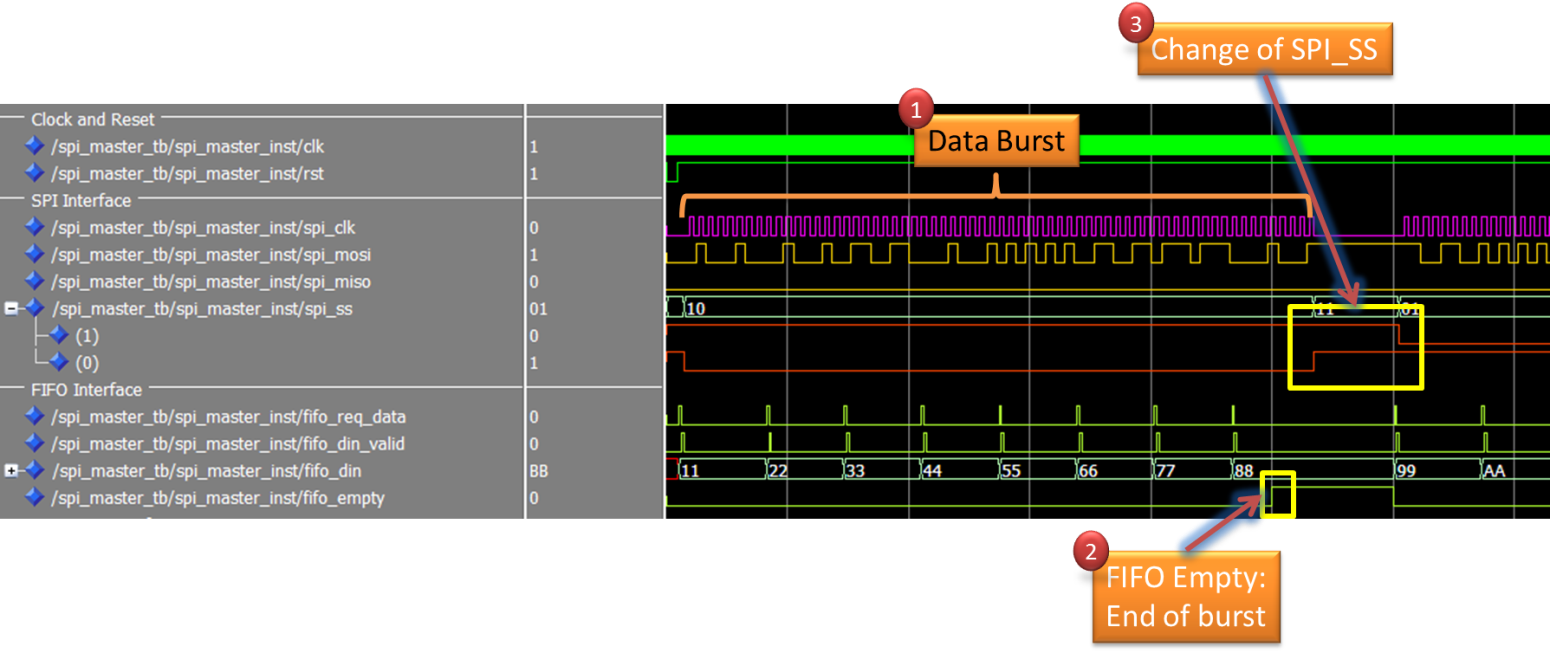


Figure 3 – Start and End of Burst

1. Data Burst
2. FIFO is empty causes end of burst
3. New data is available. This time, the second slave is addresses

## Performed Simulations for SPI Master

The following test where performed on the SPI Master

1. Different burst length (1 to 20)
2. Writing to register during active transaction
3. Using all CPOL, CPHA modes
4. Using all clock divide factors, according to the clock divide register's width
5. Setting different input values at reset, and validating that output data is at its default state
6. FIFO\_data\_valid is not asserted when it should
7. Writing not supported Clock Divide Factors (0, 1)
8. Writing special words:
   1. Only 0x00
   2. Only 0xFF

# SPI Slave

Will be added later

# Appendix A – SPI Protocol

The Serial Peripheral Interface Bus or SPI bus is a[synchronous](http://en.wikipedia.org/wiki/Synchronization_(computer_science)) [serial data link](http://en.wikipedia.org/wiki/Serial_communications) standard named by [Motorola](http://en.wikipedia.org/wiki/Motorola) that operates in [full duplex](http://en.wikipedia.org/wiki/Full_duplex) mode. Devices communicate in [master/slave](http://en.wikipedia.org/wiki/Master-slave_(technology)) mode where the master device initiates the [data frame](http://en.wikipedia.org/wiki/Data_frame). Multiple slave devices are allowed with individual [slave select](http://en.wikipedia.org/wiki/Slave_select) ([chip select](http://en.wikipedia.org/wiki/Chip_select)) lines.

## Interface

The SPI bus specifies four logic signals:

* **SPI\_CLK**: Serial Clock (output from master);
* **SPI\_MOSI; SPI\_SIMO**: Master Output, Slave Input (output from master);
* **SPI\_MISO; SPI\_SOMI**: Master Input, Slave Output (output from slave);
* **SPI\_SS**: [Slave Select](http://en.wikipedia.org/wiki/Slave_Select) ([active low](http://en.wikipedia.org/wiki/Logic_level), output from master).

## Data Transmission

To begin a communication, the master first configures the clock, using a frequency less than or equal to the maximum frequency the slave device supports. Such frequencies are commonly in the range of 1–70 MHz.

The master then pulls the chip select (SPI\_SS) low for the desired chip.

During each SPI clock cycle, a [full duplex](http://en.wikipedia.org/wiki/Full_duplex) data transmission occurs:

* The master sends a bit on the MOSI line; the slave reads it from that same line.
* The slave sends a bit on the MISO line; the master reads it from that same line.

## Clock Polarity and Phase

In addition to setting the clock frequency, the master must also configure the clock polarity (CPOL) and phase (CPHA) with respect to the data.

* At CPOL=0 the base value of the clock is zero.
  + For CPHA=0, data are captured on the clock's rising edge (low→high transition) and data are propagated on a falling edge (high→low clock transition).
  + For CPHA=1, data are captured on the clock's falling edge and data are propagated on a rising edge.
* At CPOL=1 the base value of the clock is one (inversion of CPOL=0).
  + For CPHA=0, data are captured on clock's falling edge and data are propagated on a rising edge.
  + For CPHA=1, data are captured on clock's rising edge and data are propagated on a falling edge.

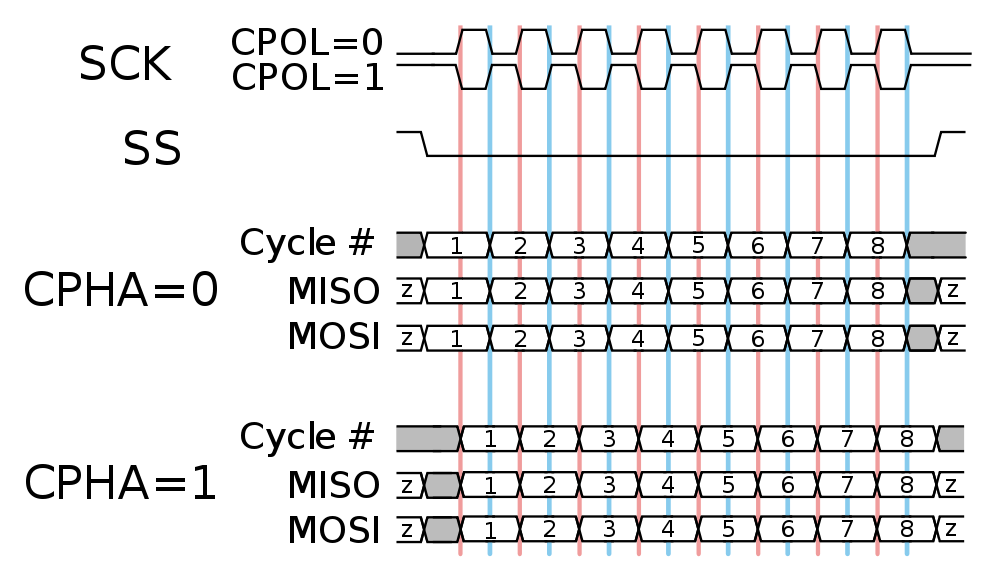


Figure 4 – SPI Protocol Diagram

That is, CPHA=0 means sample on the leading (first) clock edge, while CPHA=1 means sample on the trailing (second) clock edge, regardless of whether that clock edge is rising or falling. Note that with CPHA=0, the data must be stable for a half cycle before the first clock cycle. For all CPOL and CPHA modes, the initial clock value must be stable before the chip select line goes active.

The MOSI and MISO signals are usually stable (at their reception points) for the half cycle until the next clock transition. SPI master and slave devices may well sample data at different points in that half cycle.

This adds more flexibility to the communication channel between the master and slave.