The Serial Peripheral Interface Bus –VLSI Lab:   
Top Architecture

Contents

[Table of Changes 2](#_Toc302734542)

[Scope 2](#_Toc302734543)

[Top Architecture 2](#_Toc302734544)

[Master Host 3](#_Toc302734545)

[SPI Master 4](#_Toc302734546)

[Slave Host 5](#_Toc302734547)

[SPI Interface 6](#_Toc302734548)

[Message Structure 7](#_Toc302734549)

List of Tables

[Table 1 - Table of Changes 2](#_Toc299889229)

List of Figures

[Figure 1 - Top Architecture 2](#_Toc302734552)

[Figure 2 - Master Host 3](#_Toc302734553)

[Figure 3 - SPI Master Pinout 4](#_Toc302734554)

[Figure 4 - Slave Host 5](#_Toc302734555)

[Figure 5 - SPI Interface 6](#_Toc302734556)

[Figure 6 - Message Structure 7](#_Toc302734557)

# Table of Changes

|  |  |  |
| --- | --- | --- |
| Version | Date | Reason of change |
| 1.0 | 31.07.2011 | Creation |
| 1.1 | 02.09.2011 | Architecture modifications in SPI Master |
| 1.2 | 25.10.2011 | SPI Master Update |

Table 1 - Table of Changes

# Scope

This document describes the Top Architecture of the SPI Project, which is created in the VLSI Lab.

# Top Architecture

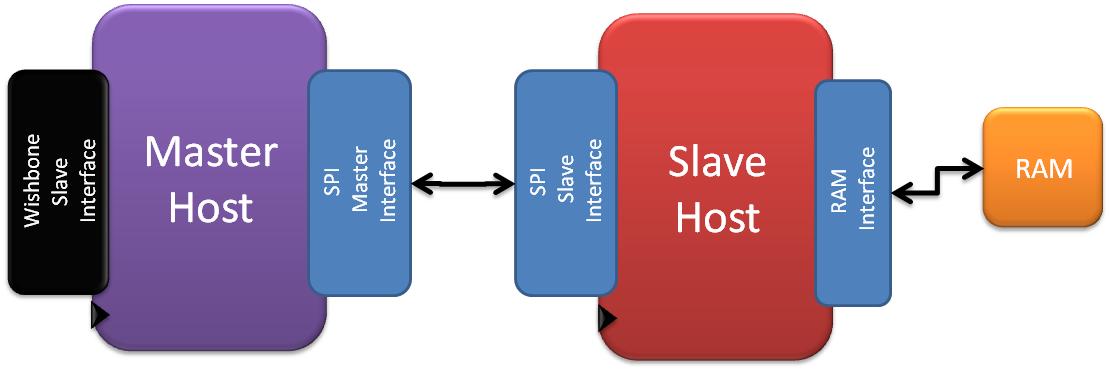


Figure 1 - Top Architecture

1. **Master Host** – Connects between the outside world to the SPI world, through the Wishbone interface.
2. **Slave Host** – Connects between the SPI world to the outside world – here it is a RAM, which data can be read from and written to.

**Note:**

SPI Master and Slave transmit and receive data at the same time. Therefore, in order to read, SPI Master should transmit data, which should be ignore by the SPI Slave.

# Master Host

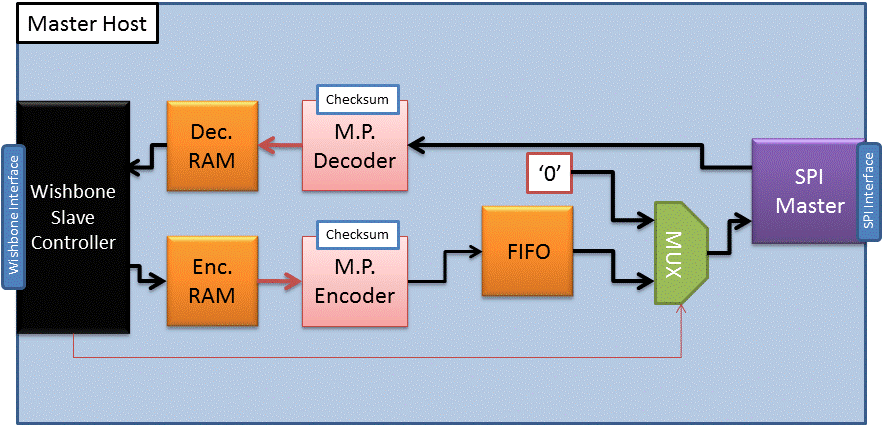


Figure 2 - Master Host

The Master Host receives a Wishbone command from the WBS (Wishbone Slave) Interface. When a new command ("Write" / "Read") is received, the *Wishbone Slave Controller* encodes the data into Message Pack, and then places it into FIFO, and from there, through MUX to the SPI Master, which transmit this command to the SPI Host as a SPI message.

Received data from the SPI Slave is decoded in the Message Pack Decoder, In case of "Read" mode (WBS\_WE\_I = '0', which means that the wishbone interface has received a read command), the data will be decoded, and transmitted through the Wishbone Interface.

When reading – "garbage" data will be inserted to the FIFO, so SPI Master will initialize read/write transaction, by changing the MUX status to transmit '0' (Simulate FIFO\_DIN\_VALID = '1', and FIFO\_EMPTY = '0').

## SPI Master

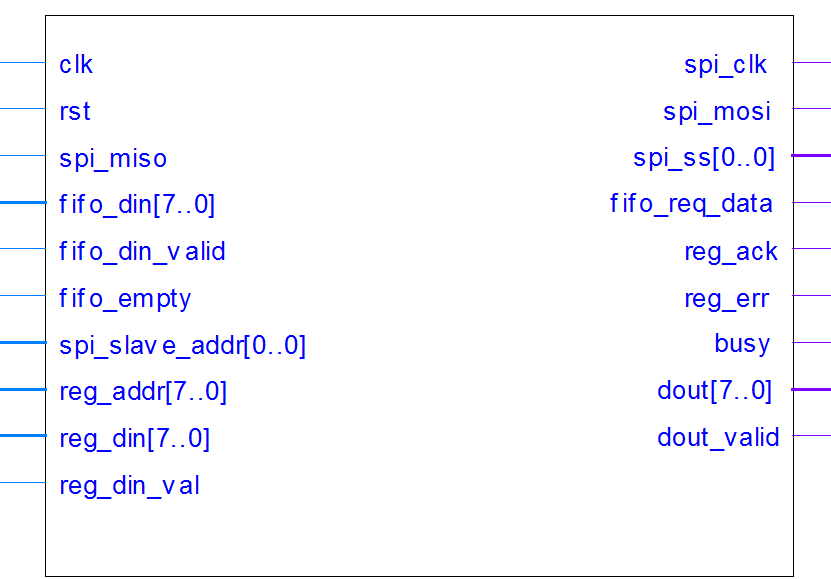


Figure 3 - SPI Master Pinout

SPI Master has interfaces to the following:

1. System clock and reset
2. SPI Slave
3. FIFO – Transaction (Write + Read) will initialize when FIFO is not empty.
4. Registers – Two registers may be written:
   1. Clock Divide Register – which hold a System Clock divide factor, to generate SPI\_CLK. For example: writing 3h to this register, will generate SPI\_CLK, which asserts / negates each 3 System Clock cycles. Minimum value to write is 2h.
   2. Configuration Register – to set CPOL and CPHA.
5. Master Host – Input is SPI Slave Select Address, and output is DOUT + DOUT\_VALID, which is the parallel data that has been received from the SPI Slave.

# Slave Host

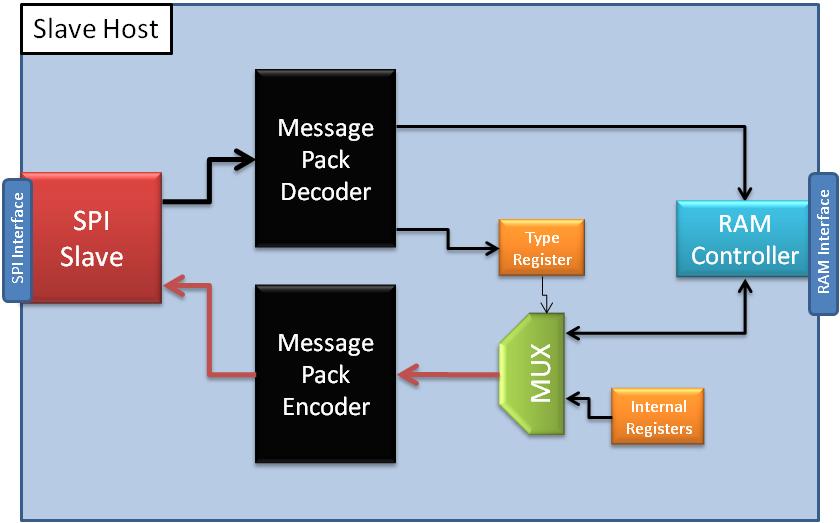


Figure 4 - Slave Host

Data is received from the SPI Master, decoded in the Message Pack Decoder, and in case of write command - is saved in the RAM, through the RAM Controller.

The current command is saved in the Type Register.

When command "read" is received, data from the RAM is encoded in the Message Pack Encoder, and transmitted through the SPI Slave.

Status register will be read upon a start of write / read transaction, or when required (Master Host request).

# SPI Interface

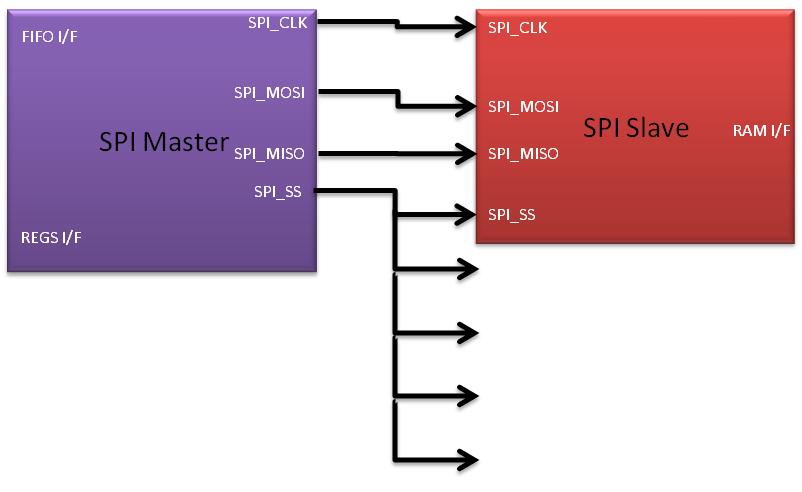


Figure 5 - SPI Interface

SPI Master Interface contains interface to FIFO, to Internal Registers interface, System clock and Reset.

SPI Slave Interface contains interface to RAM.

**SPI\_SS** is a bus, which its size is determined by a generic parameter.

# Message Structure

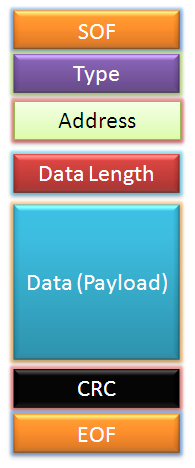


Figure 6 - Message Structure

Message structure is consumed of the following blocks:

1. **SOF** – Start of Frame (Head)
2. **Type** – Message type (command / data…)
3. **Address** – To where the message will sent to (which address in RAM)
4. **Data length** – Data (Payload) length
5. **Data (Payload)** – the data itself
6. **CRC** – Here Checksum will be used, which will be calculated from the Type block to the Data block
7. **EOF** – End of frame (Tail)