Advanced VLSI Design
Exercise 04
Assignments

Wolfgang Mueller, Heinz Nixdorf Institut/Universität Paderborn

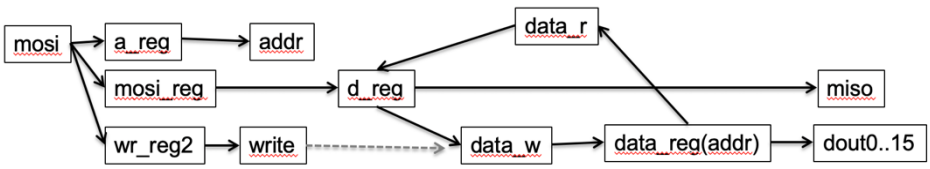
For this exercise you should be familiar with the principles of the SPI communication protocol. You can consult en.wikipedia.org/wiki/Serial_Peripheral_Interface for a very basic overview.
We apply an SPI slave in this exercise for simulation and in one of the next exercises for synthesis. SPI (Serial Peripheral Interface) defines a full duplex communication between a master and several slaves based on a 4-wire bus. The SPI slave has four input and output ports:

- SCLK: Serial Clock Input
- MOSI: Master Output Slave Input (data input from master)
- MISO: Master In Slave Out (data output to master)
- CS: active low Chip Select (selects slave when CS=0)

A selected SPI slave receives serial bits via MOSI input and shifts its output to MISO output at the same time. Our SPI slave is configured to read/write 8-bit data to 4-bit address memory (via 16 additional output ports: dout0…dout15). At a falling SCLK edge, MOSI serially reads 8 data bits followed by 4 address bits followed by a read/write bit. You can consult en.wikipedia.org for additional basic SPI information.

Assignments

1. Create a new directory and copy the two VHDL files of Exercise 04 (in the zip file) into that directory. The VHDL files implement a SPI slave for 16 x 8 bit memory and its testbench.
   The two VHDL files implement a model of a SPI and its testbench.
2. Try to understand the testbench configuration with the SPI slave dut and complete it (marked by comments). Start qhsim, create a new project, compile and simulate the VHDL configuration of the SPI slave with the testbench and the given test vector. Recall that the setup for qhsim is given in the description of Exercise 02.
3. Inspect and try to understand the VHDL implementation of the SPI slave and the testbench by simulation. Try to follow the shift in and shift out behavior of the data, the address and the write/read bit and try to follow the following signal dependencies:



4. Extend test vectors by use of the transfer_data procedure to address different variations of signals and data read & write transfers.