Serial Peripheral Interface

Description

Refer to the attached document “Introduction to SPI Interface” to learn about the details of the SPI.

In this project, we will only focus on the Multi-Slave Regular Mode (as shown in Fig. 6) with SPI mode 1 (CPOL = 0, CPHA = 1 as shown in Fig. 3).

Attached Files:

- “Introduction to SPI Interface” by Piyu Dhaker.
- Development Integration Testbench “development_tb.v”.

Requirements:

Design and implement the following components of the SPI modules using verilog such that they match the requirements of the development testbench and match the SPI specifications:

- Master
- Slave
- Self-Checking Testbenches for the Master and Slave (see FAQs).

Write a report that discusses the following:

- The design process of the Master, the Slave and their test benches.
- The simulation results (Screenshots from the waveform and the output).

Deliverables

- All the verilog files in the project.
- A text file with the team number and member names. It also must include the work distribution over the team members.
- A report containing all the content mentioned in the requirements.

Grade Distribution

The total grade is calculated from 20 points. It will be distributed as follows:

- Simulation: 8 points
    - Compilation: 2 points (0.5 point for each module).
    - Slave/Master Simulation: 2 (1 point for each test bench).
    - Integration Simulation: 4 points for the hidden delivery test benches.
- Design: 8 points
    - Master: 4 points (2 for Master + 2 for Master TB).
    - Slave: 4 points (2 for Slave + 2 for Slave TB).
- Report: 4 points
    - Module Design Report: 3 points (1 for each module, 0.5 for each test bench).
    - Simulation Results: 1 point (0.5 points for each test bench).

Each team can include up to 4 students. Each member will be evaluated for their contributions and understanding individually. The delivery date is Tuesday June 1st 2021 11:59 PM.

Frequently Asked Questions:

- What data should be communicated between the master and the slave?
    - Both the master and the slave modules must have an interface to receive data and commands from controllers (which will be the testbenches in our case). The only command we have in our case is to tell the master to initiate the transmission with one of the slaves.
    - Refer to the development testbench to find out more about the interface between the master/slave and the test bench.
- What is the size of the data to be transmitted?
    - 8-bits.
- How many slaves will be connected to the master?
    - 3 slaves only.
- What is “Shifting” and what is “Sampling”?
    - Shifting happens when the Master writes data to the MOSI and the Slave writes data to the MISO. (This doesn’t necessarily mean that the master and the slave must be implemented as shift registers).
    - Sampling happens when the Master reads data from the MISO and the Slave reads data from the MOSI.
- What is a self-checking testbench?
    - A self checking test bench is a test-bench that automatically verifies the output of the module. So it should have some test vectors consisting of: test inputs and the corresponding expected outputs.
    - At the end of the simulation, the testbench should automatically display whether the test was a success (all outputs match their corresponding expected outputs) or a failure (any output mismatched its corresponding expected output).

- If the test fails, it should report the failed test cases (test input, expected output and actual output) and the number of failed test cases.
- Self-checking testbenches are more common in hardware design since real applications tend to have lots of modules, so it is impractical to manually check the outputs of each test case in the project.
- The attached development test bench shows an example of a self-checking testbench.