Weekly Report - 2&3.

Names and Roll no: Ch. Sai Nived (AP18110020174) A. Rohit Kumar (AP18110020168) P. Jithendra (AP18110020164) A. Kaushik Sai (AP18110020156)
Guide: Siva Sankar Yellampalli
Place of Execution: HDL(Verilog), Xilinx., EDA playground.
Project Title: Implementation of the communication protocols SPI by the HDL-Verilog language.
Week Starting Date: March 15, 2021
Week Ending Date: March 27, 2021

Literature Studied During the Week:

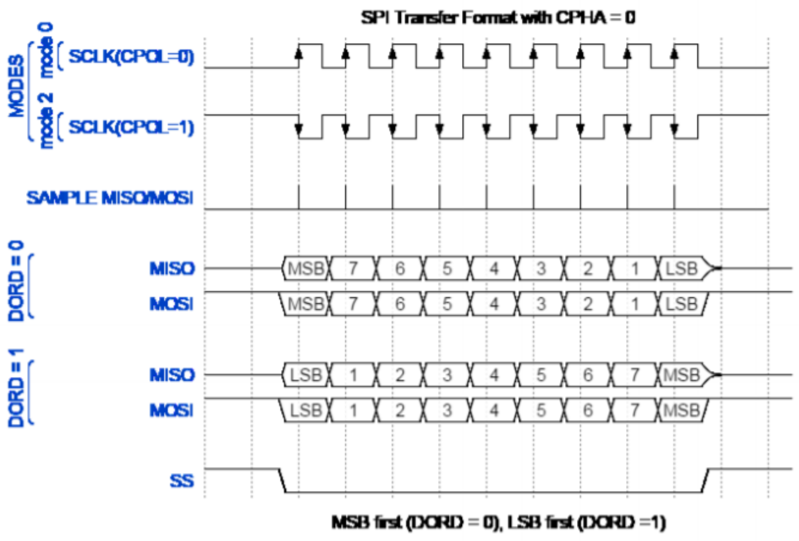
Motorola, Inc. (Accessed 14.7.2016). Motorola's Spi Block Guide V04.01.

Prasad, M. (Accessed 9.6.2016). Serial Peripheral Interface - SPI Basics.

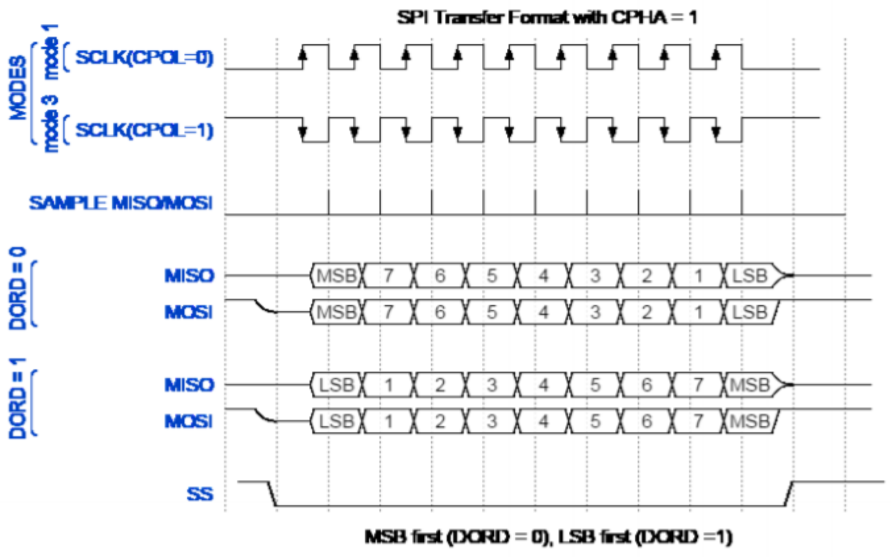
Byte Paradigm sprl. (Accessed 8.11.2016). Introduction to I²C and SPI protocols.

1. Details of Work Carried out:

The SPI-protocol is using four different modes. The modes differ from each other depending on the Clock Phase (CPHA) and Clock Polarity (CPOL)



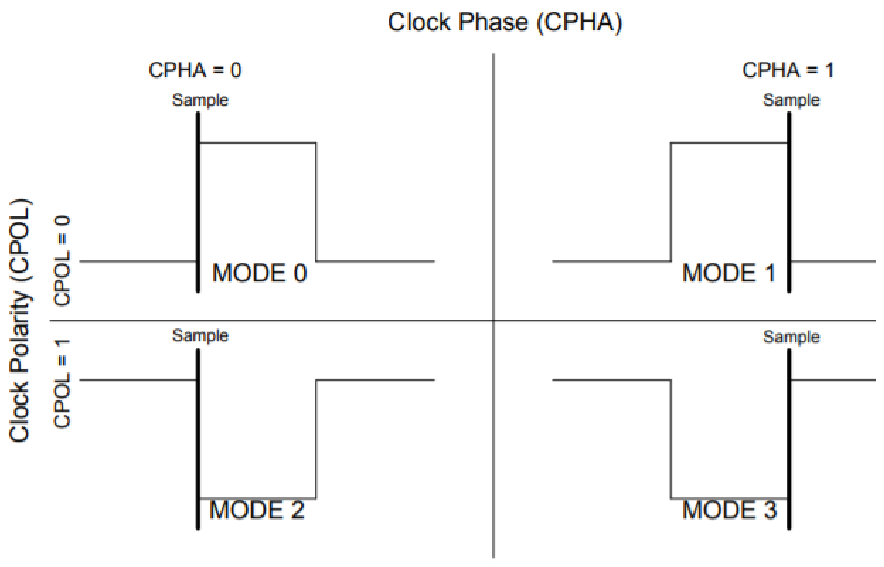
Timing diagram



SPI-bus protocol modes 1 and 3 with CPHA 1. The slave select is zero active. The CPOL defines the idle and active state of the clock. The CPHA defines which edge of the clock phase is used to sample the signal. These parameters set the number of modes to four.

	Leading Edge	**Falling Edge**	**SPI Mode**
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)	0
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)	1
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)	2
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)	3

CPOL and CPHA functionality in different SPI modes



.

The model dictates the basic functionality of the SPI instance. This means that the received data is always literally interpreted by the receiving counterpart. The ambiguity of the data is not taken into account. This subject is addressed in more detail in the following sections. Usually, the modes can be changed for the master when initializing the instance.

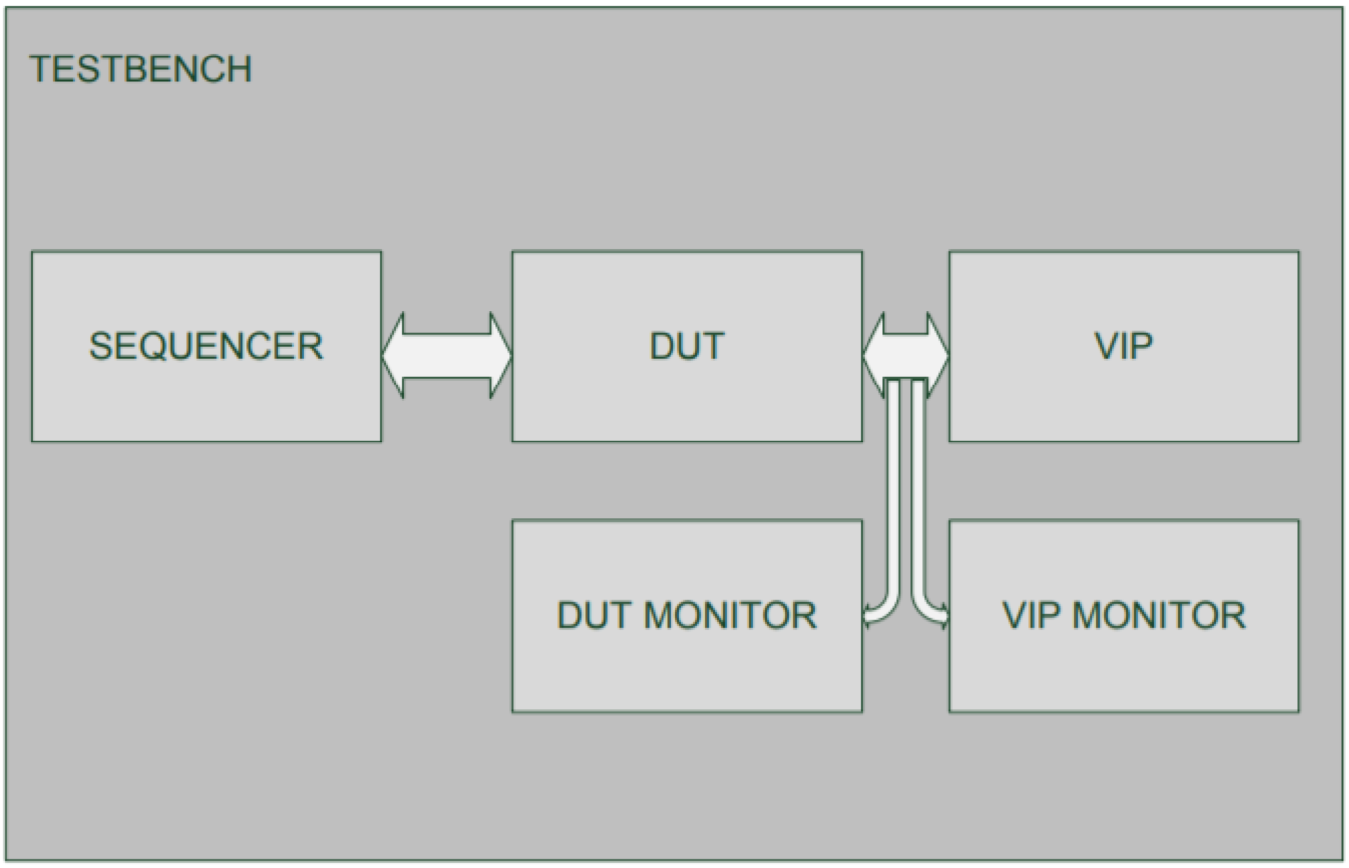
The SPI-slave components are on the other hand more or less static when it comes to the mode selection. This means that the SPI-master component has to adapt to the used protocol between SPI-master and –slave. Because there is no standardized handshaking-protocol for the SPI-bus. As mentioned earlier, the data from SPI-bus is interpreted literally by the receiver, the received data can cause the SPI-instance to react to the incoming data. The response on the other hand might not be as desired.

TESTING:

The good design practice indicates that to pass design criteria a new design has to be tested and verified. The design has the desired properties only on paper if it is not tested in any way. This is a common problem when new designs are created. If a new Intellectual Property (IP) is created but there is no counterpart to test it against, the IP's functionality can be only guessed. This can be solved either by using testbenches with verification IPs, real components, or using logic analyzers, the designer has to evaluate what is the most efficient way to make these tests to see how the design is behaving. The following subsections try to give some points of view on what to expect when using these different techniques. These testbenches can usually be used in simulations and emulators. If these were to be used in Field Programmable Gate Array (FPGA) environment, the HDL has to be fully synthesizable or some actions have to be done to implement the design to the FPGA, i.e. the use of transactors and other instances alike is vital. The testbenches are configurable and can be fed with large data patterns to cover all the corner cases inside the system.

Testbench:

A common way to test the new design is to test it in a testbench that is created specifically to fit the design at hand. This practice can be utilized by using highly complex Hardware Description Language (HDL) structures. A commonly used HDL language is Universal Verification Methodology (UVM).



These testbenches can usually be used in simulations and emulators. If these were to be used in Field Programmable Gate Array (FPGA) environment, the RTL/HDL has to be fully synthesizable or some actions have to be done to implement the design to the FPGA, i.e. the use of transactors and other instances alike is vital. The transactor is a component that actively interacts with the DUT interfaces. The testbenches are configurable and can be fed with large data patterns to cover all the corner cases inside the system. The creation of the testbench can be even more time-consuming than the DUT design. So to use testbenches, the designer has to be sure that all the specifications around the DUT are valid. Usually, testbenches are created separately from the DUT. This is advisable to reduce the error margin. The testbench can be fitted to include automatic evaluating properties, aka assertions, that indicate if the functionality of the DUT is compromised due to inconsistencies in internal or external signals. Assertions however can be only utilized in simulations and emulations; this is due to assertions not compiled in synthesis tools. Using testbenches gives many possibilities to the designer to check the functionality of the system via tools provided by the simulator, emulator, and third-party vendor analyzing tools.

2. **Problems faced during Execution: Timing diagram**

- Timing diagram
- CPOL and CPHA functionality in different SPI modes
- Methods used for testing SPI

3. **Work Carried out to be Next week:**

- Prototyping of SPI.