Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Centre of Excellence in VLSI

SERIAL PERIPHERAL INTERFACE(SPI) PROJECT REPORT

COURSE OFFERED BY MAVEN SILICON

DATE OF SUBMISSION: 29TH JUNE 2023

NAME: VINYAS A SHETTY

REGISTRATION NUMBER: 20BEC0780

E-MAIL ID: vinyasshetty.a2020@vitstudent.ac.in

Personal E-mail ID: shettyvinyas1@gmail.com

CONTENTS TO BE INCLUDED:

The project report should contain Protocol information and Outputs (Waveforms).

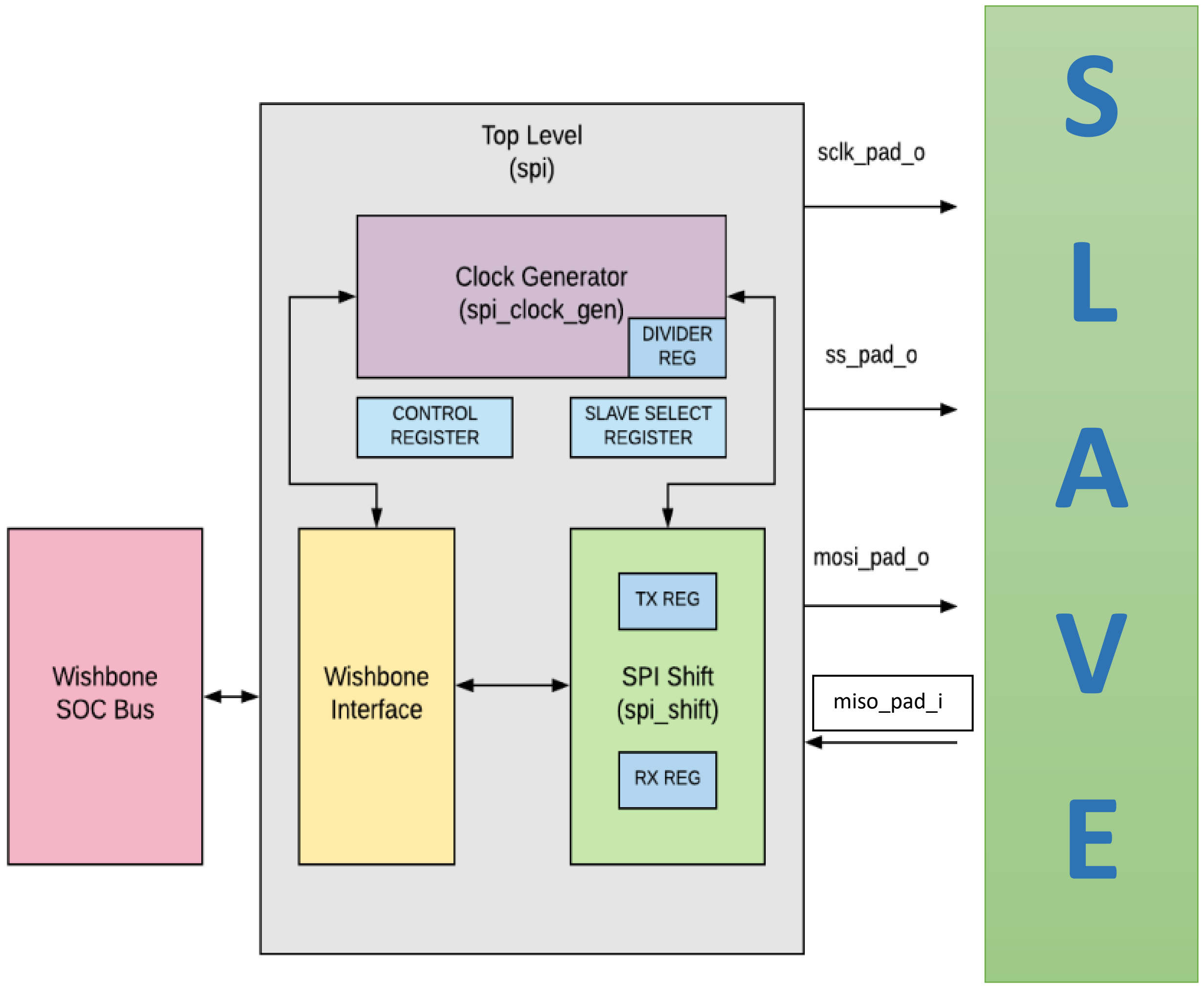
Don't include and paste any codes

1. Top module block diagram - Overall functionality

2.Sub-blocks block diagram - Functionality

3. Output waveforms

Output waveforms should contain Top module output and sub-blocks output waveforms



WISHBONE MASTER

SPI MASTER CORE

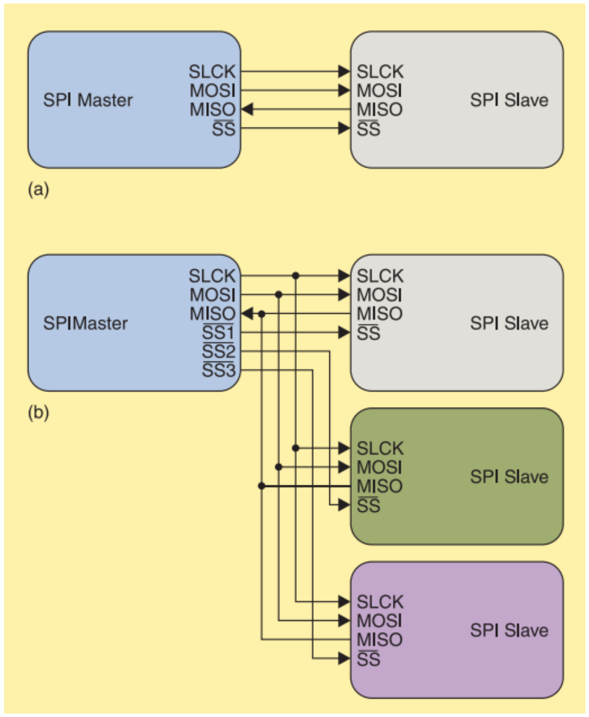
SPI SERIAL SLAVE

SPI FULL BLOCK ARCHITECTURE

Serial peripheral interface (SPI) is a synchronous interface that willing to allow several SPI microcontrollers or SPI-type peripherals to be connected with each other. Serial peripheral interfaces (SPI) are widely used to provide economical board level interfaces between different devices such as microcontrollers, Digital to Analog Converter's, Analog to Digital Converter's and other.

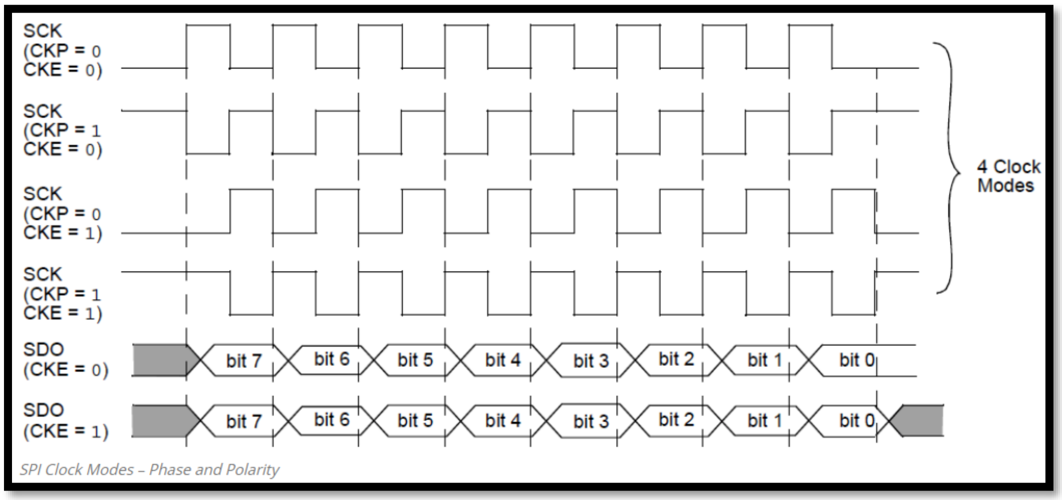
SPI is a protocol on four signal lines

1. A clock signal (SCLK) sent from the bus master to all slaves; all the SPI signals are synchronous to this clock signal
2. A slave select signal (SSn) for each slave, used to select the slave the master communicates with
3. A data line from the master to the slaves, named Master Out-Slave In (MOSI)
4. A data line from the slaves to the master, named Master In-Slave Out (MISO).



(a) shows an SPI master connected to a single slave (point-to-point topology).

(b) shows an SPI master connected to multiple slaves



Four communication modes are available (MODE 0, 1, 2, 3) that define

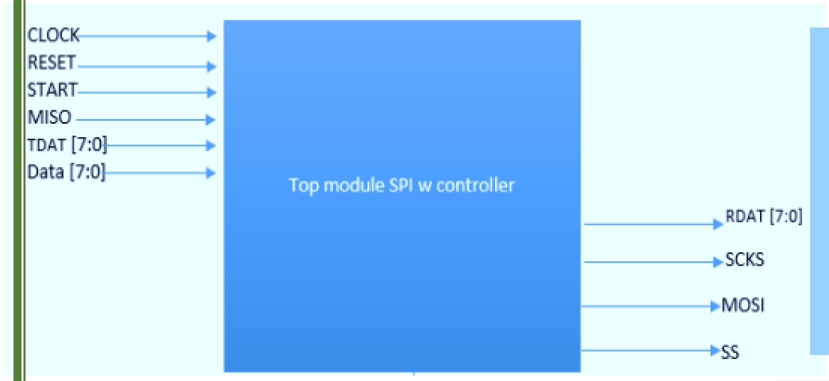
◗the SCLK edge on which the MOSI line toggles

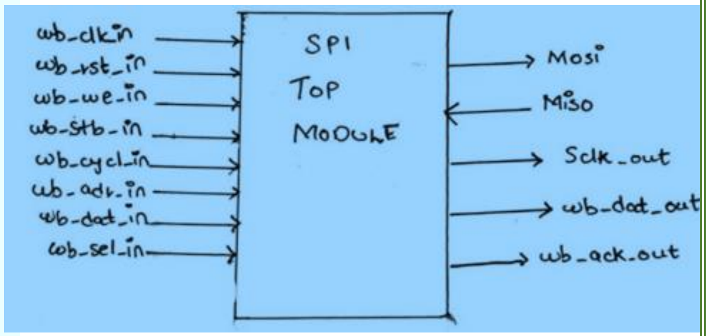
◗ the SCLK edge on which the master samples the MISO line

◗ the SCLK signal steady level (that is, the clock level, high or low, when the clock is not active)

TOP MODULE BLOCK :

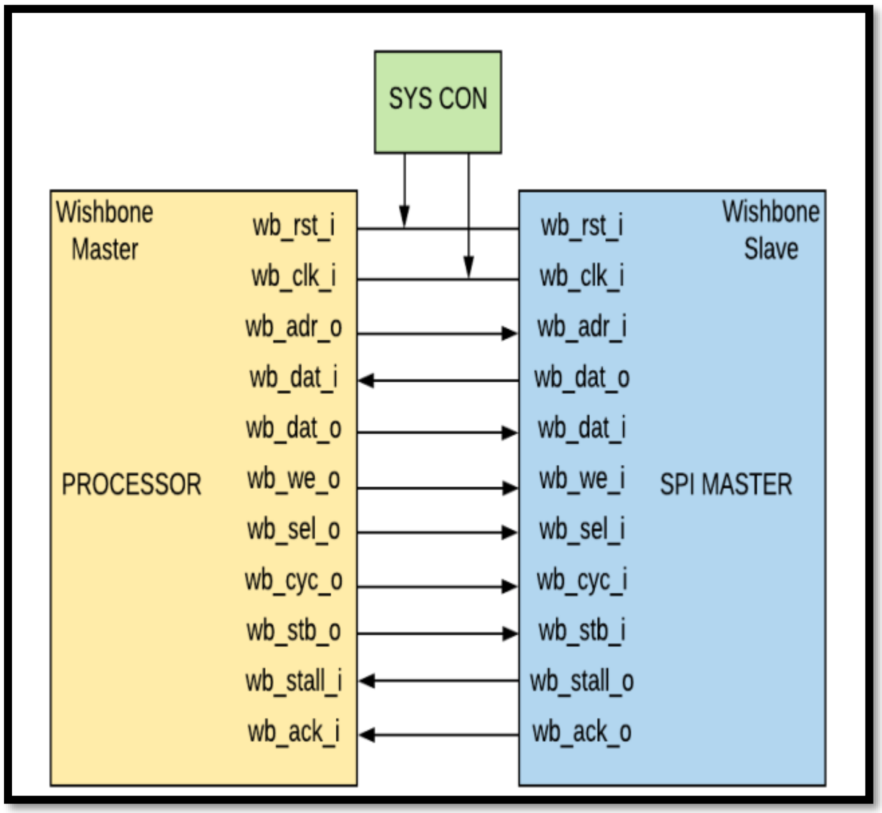
Consists of Wishbone master , SPI Core and SPI Serial Slave……





Pins (Output Top SPI)	Function
MOSI	This pin is used to transmit data out of the SPI module when it is configured as a Slave and receive data when it is configured as Master
SCKS	This pin is used to output the clock with respect to which the SPI transfer data or receive clock in case of Slave
RDAT [7:0]	Start data transmission (e.g.: start =1)
SS	This pin is used to output the select signal from the SPI module to another peripheral with which a data transfer is to take place when its configured as a Master and its used as an input to receive signal when the SPI is configured as Slave

Pins (Input Top SPI)	Function
CLOCK	The clock input defines the bit-rate of the serial communication (16MHz max frequency)
RESET	Resets the SPI state machine to the idle state
START	Start data transmission (e.g.: start =1)
MISO	This pin is used to transmit data out of the SPI module when it is configured as a Master and receive data when it is configured as Slave
Tdat [7:0]	Data transmitted by the APB
Data [7:0]	Master input data



Wishbone Interface:

The Wishbone Master is a module that initiates and controls transactions on the Wishbone bus. It generates the necessary control signals and manages the timing and synchronization of data transfer between the master and the other modules connected to the bus. In the case of SPI, the Wishbone Master is responsible for controlling the SPI communication process and interacting with the other components involved.

Port	Width	Direction	Description
wb_clk_i	1	Input	Master clock input
wb_rst_i	1	Input	Asynchronous active low reset
wb_int_o	1	Output	Interrupt signal request
wb_cyc_i	1	Input	Valid bus cycle
wb_stb_i	1	Input	Strobe/core select
wb_adr_i	32	Input	Address bit
wb_we_i	1	Input	Write enable
wb_dat_i	32	Input	Data input
wb_dat_o	32	Output	Data output
wb_ack_o	1	Output	Normal bus termination
wb_stall_o	1	Output	Stall communication

SPI MASTER CORE:

The SPI Master Core is the central controller of the SPI system. It manages the overall SPI communication process, coordinating data transfer between the master and slave devices. The SPI Master Core interacts with the Clock Generator, Shift Register, and other control logic to facilitate efficient and reliable communication.

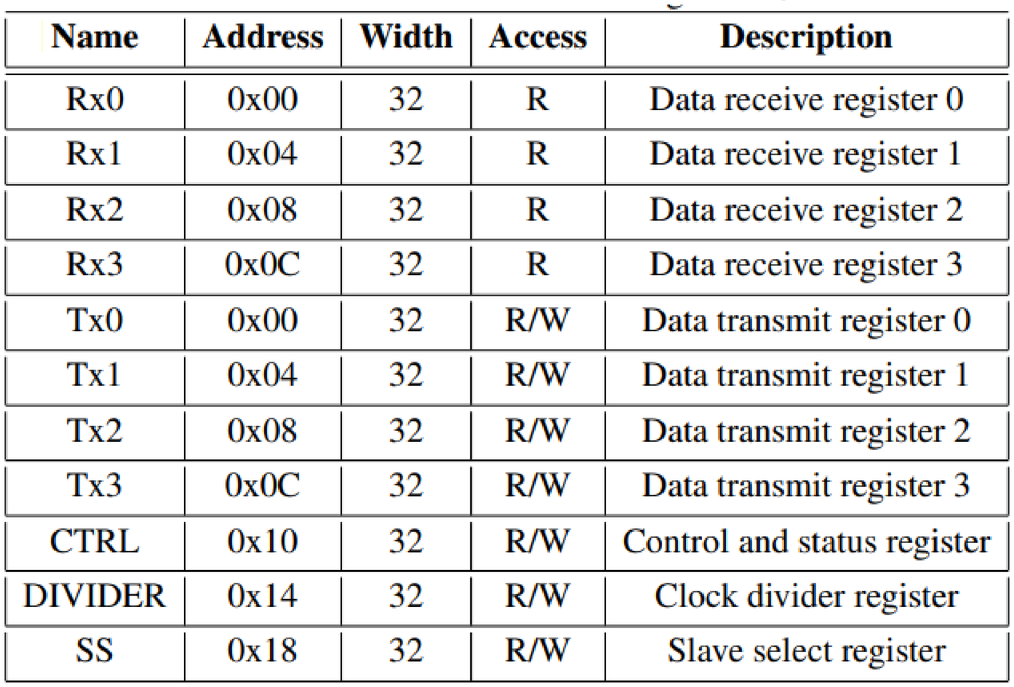
The SPI Master Core generates the necessary control signals to control the data transfer. It manages the selection of slave devices using the Slave Select (SS) lines. By activating the appropriate SS line, the SPI Master Core communicates with a specific slave device while keeping other slave devices inactive.

The core controls the timing of the data transfer by generating the clock signal (SCLK) and determining the data sampling and shifting points. It ensures that data is transmitted and received correctly, handling any necessary handshaking protocols or error detection mechanisms.

Additionally, the SPI Master Core may include configuration registers to set parameters such as clock polarity, clock phase, data order (MSB or LSB first), and other SPI-specific settings. These registers allow customization of the SPI communication protocol to meet the requirements of different slave devices.

The role of the top-level module is to get the basic structure of high-speed reusable SPI bus sub-components to work smoothly. Therefore, the top-level of the SPI module controls normal operation of clock generator module and serial data transmission module

The SPI master core uses the register



Name	Address	Width	Access	Description
Rx0	0x00	32	R	Data receive register 0
Rx1	0x04	32	R	Data receive register 1
Rx2	0x08	32	R	Data receive register 2
Rx3	0x0C	32	R	Data receive register 3
Tx0	0x00	32	R/W	Data transmit register 0
Tx1	0x04	32	R/W	Data transmit register 1
Tx2	0x08	32	R/W	Data transmit register 2
Tx3	0x0C	32	R/W	Data transmit register 3
CTRL	0x10	32	R/W	Control and status register
DIVIDER	0x14	32	R/W	Clock divider register
SS	0x18	32	R/W	Slave select register

1. **TxX Register:** Holds the transmitted data during the SPI transfer. The number of registers used depends on the character length specified in the CTRL register.
2. **ASS Register:** Determines the behavior of the slave select (SS) signal. If set, the SS signal is generated automatically by the SPI controller. If cleared, the SS signals are controlled by writing and clearing bits in the SS register.
3. **DIVIDER Register:** Divides the system clock frequency to generate the serial clock (s_clk) for SPI communication.
4. **SS Register:** Controls the activation state of the slave select lines. When ASS is cleared, writing 0x1 to a bit location activates the corresponding SS line.
5. **IE Register:** Enables the interrupt output to be set active once after a transfer is finished. The interrupt signal is cleared after a read or write operation to any register.
6. **LSB Register:** Determines the bit transmission and reception order. When set, the least significant bit is sent/received first.
7. **Tx_NEG and Rx_NEG Registers:** Determine the timing of the MOSI and MISO signals relative to the serial clock edge.
8. **GO_BSY Register:** Starts the transfer when set to 0x1 and remains set during the transfer. It is automatically cleared after the transfer is finished.
9. **CHAR_LEN Register:** Specifies the number of bits to be transmitted in one transfer, allowing up to 64 bits.

SPI SLAVE:

1. Role of the SPI Slave: The SPI Slave is a peripheral device that communicates with an SPI Master. It receives commands and data from the master and responds accordingly. The slave device can be a sensor, memory chip, display, or any other device that needs to communicate with the master using the SPI protocol.

2. Connection to the SPI Bus: The SPI Slave is connected to the SPI bus, which consists of four lines: Master Out Slave In (MOSI), Master In Slave Out (MISO), Serial Clock (SCLK), and Slave Select (SS). The MOSI line is used by the master to send data to the slave, while the MISO line is used by the slave to send data to the master. The SCLK line provides the clock signal for synchronized communication, and the SS line is used to select the specific slave device with which the master wants to communicate.

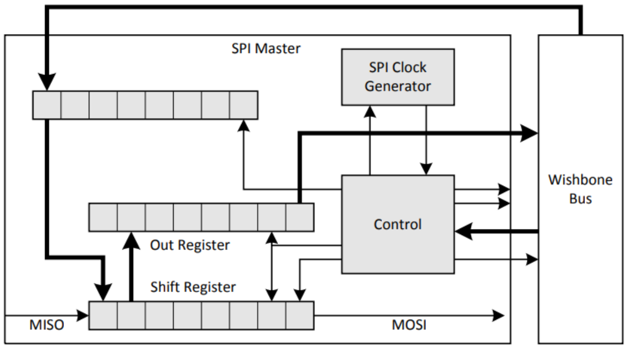
3. Data Transfer: The SPI Slave receives data from the SPI Master through the MISO line and sends data back to the master through the MOSI line. The data is transferred in a serial fashion, one bit at a time, synchronized with the clock signal. The order of data transmission can be either Most Significant Bit (MSB) first or Least Significant Bit (LSB) first, depending on the configuration.

4. Slave Select (SS): The Slave Select (SS) line is used by the master to select the specific slave device it wants to communicate with. Each slave device on the bus typically has its own SS line, allowing the master to individually address and communicate with multiple slaves. When the SS line corresponding to a particular slave is asserted by the master (pulled low), that slave is activated and ready to receive commands and data. The other slave devices remain inactive during this time.

5. Communication Protocol: The SPI Slave follows a communication protocol defined by the SPI standard. The specific details of the protocol, such as clock polarity, clock phase, and data order (MSB or LSB first), should be configured to match the requirements of the SPI Master. The SPI Slave interprets the commands and data received from the master according to the protocol specifications and performs the requested operations or returns the requested information.

6. Interrupts and Handshaking: Some SPI Slaves support interrupt capabilities to signal the master about specific events or data availability. The slave can raise an interrupt line to inform the master about data readiness or any other conditions that require attention. Additionally, handshaking protocols like flow control signals (e.g., Ready, Acknowledge) can be implemented to ensure proper synchronization and data integrity during communication.

SUB-BLOCK MODULE:



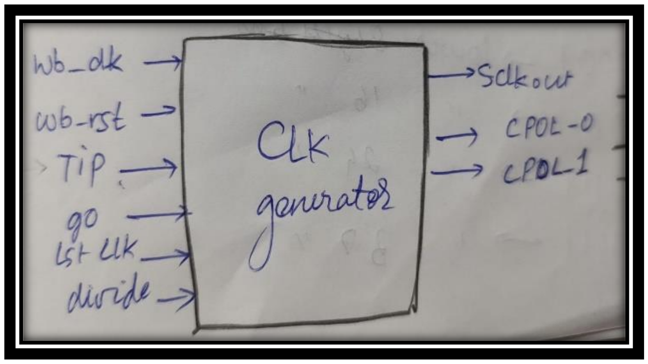
SPI CORE consists of CLOCK GENERATOR and SHIFT REGISTER which form the Sub-block of the SPI

CLOCK GENERATOR:

The clk_gen is responsible for the generation of the clock signal from the external system clock wb_clk_i, in accordance with different frequency factor of the clock register and produce the output signal s_clk_o. Since there is no response mechanism for Serial Peripheral Interface, in order to ensure the reliability of timing, the clk_gen module can generate reliable serial clock transmission with odd or even frequency division in the register. Clock divider is essential part of digital ASIC and FPGA design, the idea here is to produce frequency relevant to the communication system. Even frequency division is achieved in order to save resources. The core generates the s_clk_o by dividing the wb_clk_i; Arbitrary clock output frequency is achieved by changing the value of the divider. The clock signal generated by the Clock Generator is distributed to both the SPI Master and Slave devices. The SPI Master uses the clock signal to determine the rate at which it sends data on the MOSI line and samples data on the MISO line. The SPI Slave uses the clock signal to synchronize its data transmission on the MISO line and data reception on the MOSI line. The Clock Generator may also synchronise its clock signal with other system clocks in a bigger system to maintain overall system synchronisation and guarantee accurate timing of SPI communication with other components

The expression of sclk_o and wb_clk_i is as follows

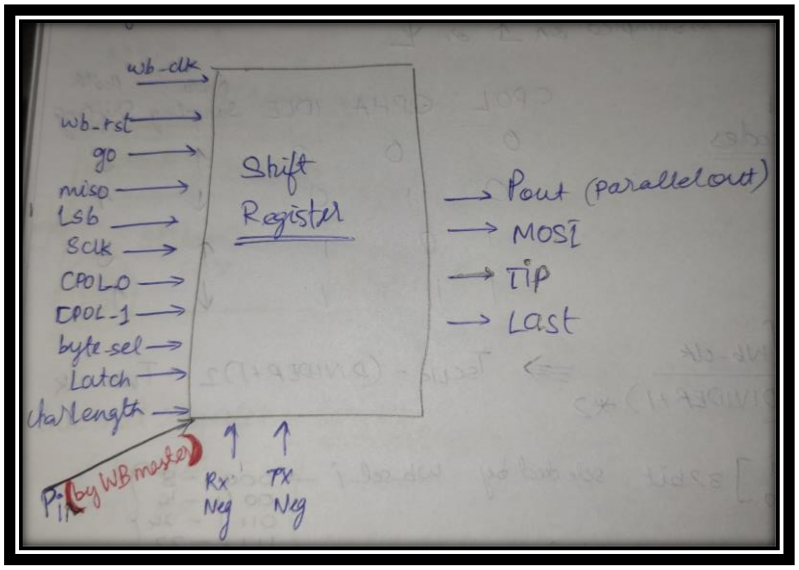
$$f_{sclk} = f_{wbclk}/(DIVIDER+1)*2$$



BLOCK DIAGRAM OF CLK_GENERATOR

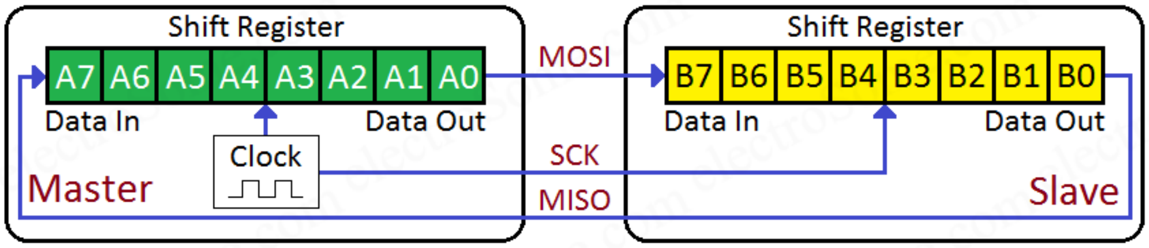
SHIFT REGISTER:

Serial data transfer module forms the data transfer core module. It is responsible for converting input parallel data into serial output data to transmit at MOSI and convert input MISO serial data into parallel out. The Receive and Transmit register share same flip-flops. It means that what data is received from the input data line in one data transfer will be transmitted on the output line in the next transfer if no write access to the transmit register was performed between the transfers. The advantage of this is it uses fewer hardware resources, therefore, lesser power consumption. SPI Master core in host side acts as a slave device to receive input data, and at the same time as the master device transmits output data



BLOCK DIAGRAM OF SHIFT REGISTER

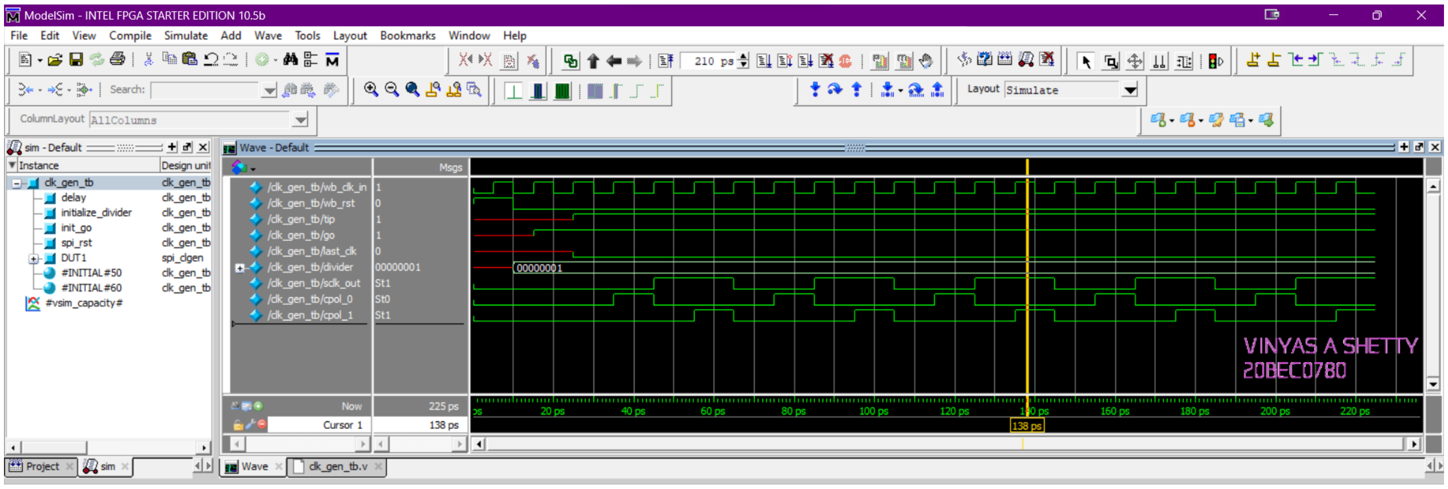
The transmissions involve two shift register of a pre-configured word size are present one each at master and slave ends. Both the shift registers act as a ring buffer . While shifting out the data usually the least significant bit from the master is sent to the most significant bit position of the slave receive register, and at the same time, the least significant bit of the slave goes to the vacant least significant bit. Both master and slave register acting in a left shift register fashion and the register values are exchanged with respect to SCLK . If more data needs to be exchanged, then the shift registers are loaded with new data, the and the process is repeated. Finally, after the data values are transmitted then master stops toggling the SCLK and it deselects the slave.



\

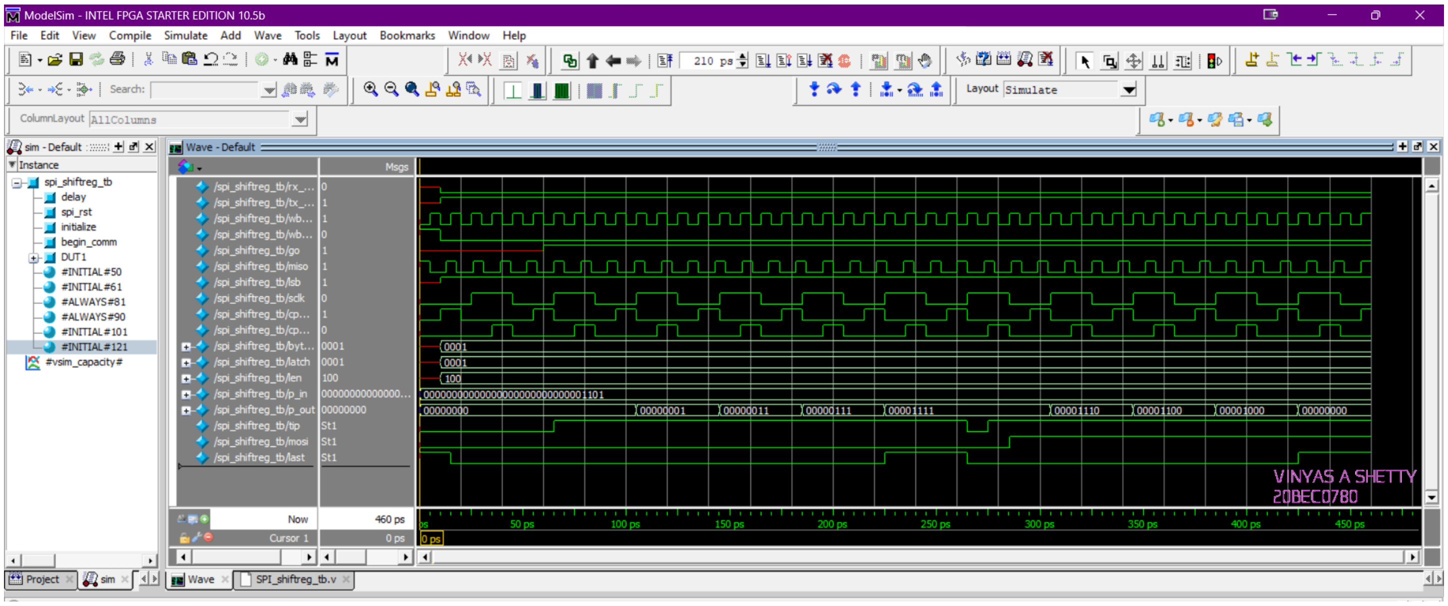
OUTPUT WAVEFORMS:

CLOCK GENERATOR::-SUB-BLOCK OF SPI CORE

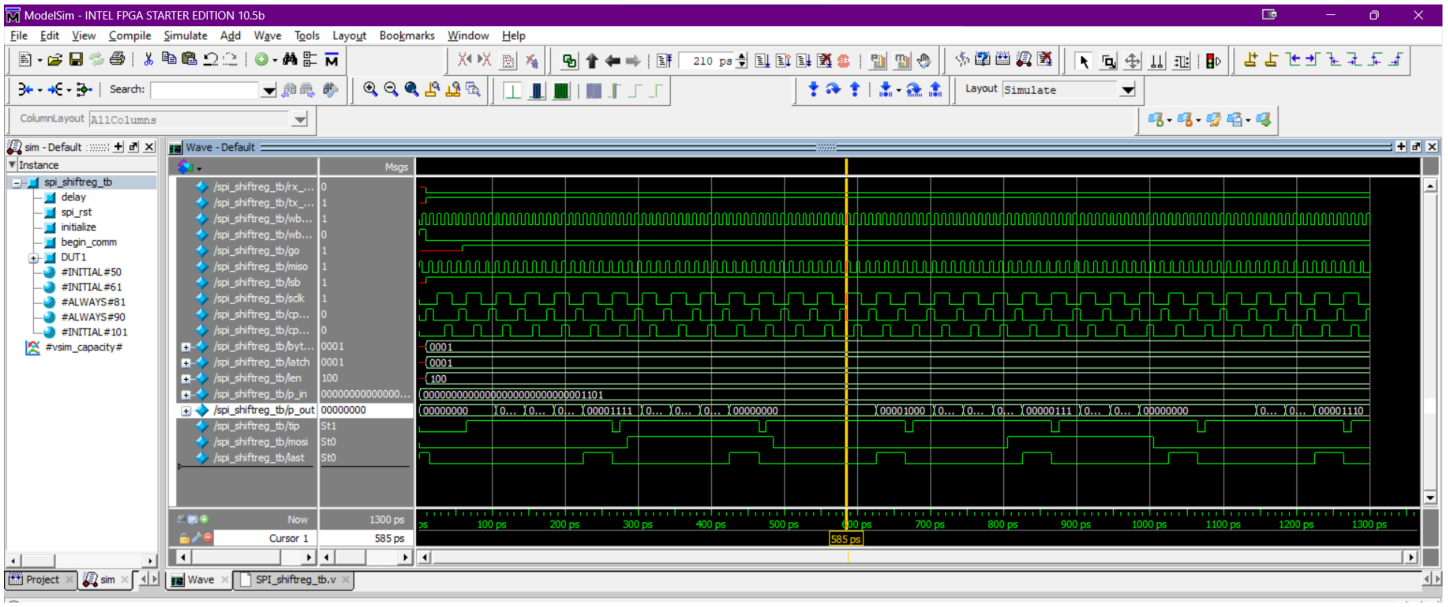

ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
VINYAS A SHETTY
20BEC0780

```
/dk_gen_tb/wb_clk_in    1
/dk_gen_tb/wb_rst       0
/dk_gen_tb/tip          1
/dk_gen_tb/go           1
/dk_gen_tb/last_clk     0
/dk_gen_tb/divider      00000001
/dk_gen_tb/sclk_out     St1
/dk_gen_tb/cpol_0       St0
/dk_gen_tb/cpol_1       St1

Now       225 ps
Cursor 1  138 ps
```

SHIFT REGISTER::-SUB-BLOCK OF SPI CORE


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
VINYAS A SHETTY
20BEC0780

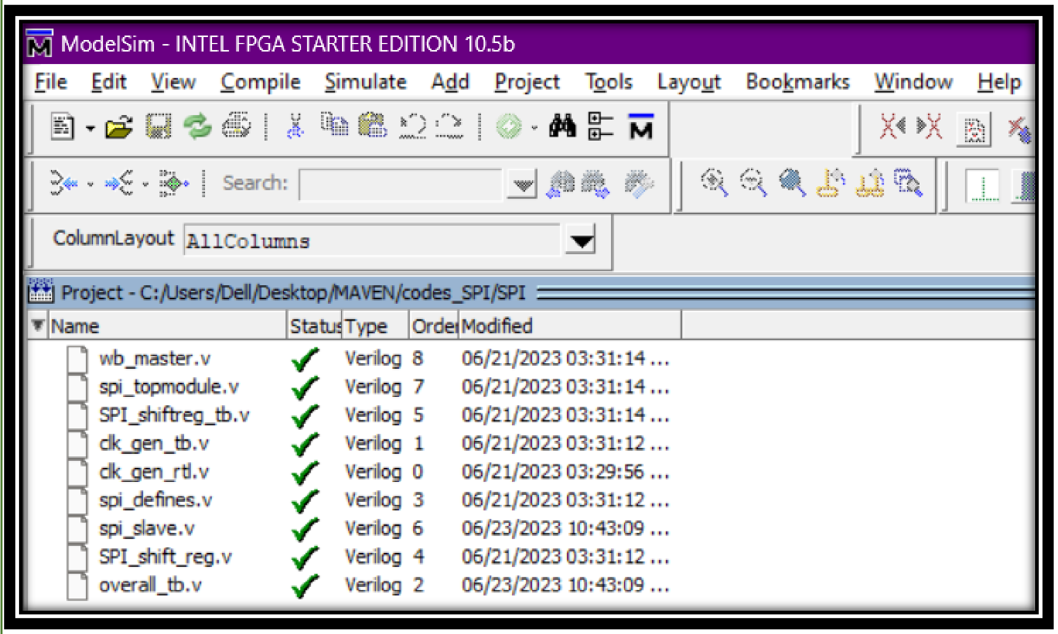
```
/spi_shiftreg_tb/rx_...   0
/spi_shiftreg_tb/tx_...   1
/spi_shiftreg_tb/wb...    1
/spi_shiftreg_tb/wb...    0
/spi_shiftreg_tb/go       1
/spi_shiftreg_tb/miso     1
/spi_shiftreg_tb/lsb      1
/spi_shiftreg_tb/sclk     0
/spi_shiftreg_tb/cp...    1
/spi_shiftreg_tb/cp...    0
/spi_shiftreg_tb/byt...   0001
/spi_shiftreg_tb/latch    0001
/spi_shiftreg_tb/len      100
/spi_shiftreg_tb/p_in     00000000000000...
/spi_shiftreg_tb/p_out    00000000
/spi_shiftreg_tb/tip      St1
/spi_shiftreg_tb/mosi     St1
/spi_shiftreg_tb/last     St1

Now       460 ps
Cursor 1  0 ps
```


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help

```
/spi_shiftreg_tb/rx_...   0
/spi_shiftreg_tb/tx_...   1
/spi_shiftreg_tb/wb...    1
/spi_shiftreg_tb/wb...    0
/spi_shiftreg_tb/go       1
/spi_shiftreg_tb/miso     1
/spi_shiftreg_tb/lsb      1
/spi_shiftreg_tb/sclk     1
/spi_shiftreg_tb/cp...    0
/spi_shiftreg_tb/cp...    0
/spi_shiftreg_tb/byt...   0001
/spi_shiftreg_tb/latch    0001
/spi_shiftreg_tb/len      100
/spi_shiftreg_tb/p_in     00000000000000...
/spi_shiftreg_tb/p_out    00000000
/spi_shiftreg_tb/tip      St1
/spi_shiftreg_tb/mosi     St0
/spi_shiftreg_tb/last     St0

Now       1300 ps
Cursor 1  585 ps
```

TOP MODULE ::

- WISHBONE MASTER +
- CLOCK GENERATOR +
- SHIFT REGISTER +
- SLAVE


ModelSim - INTEL FPGA STARTER EDITION 10.5b
File Edit View Compile Simulate Add Project Tools Layout Bookmarks Window Help
Search:
ColumnLayout AllColumns

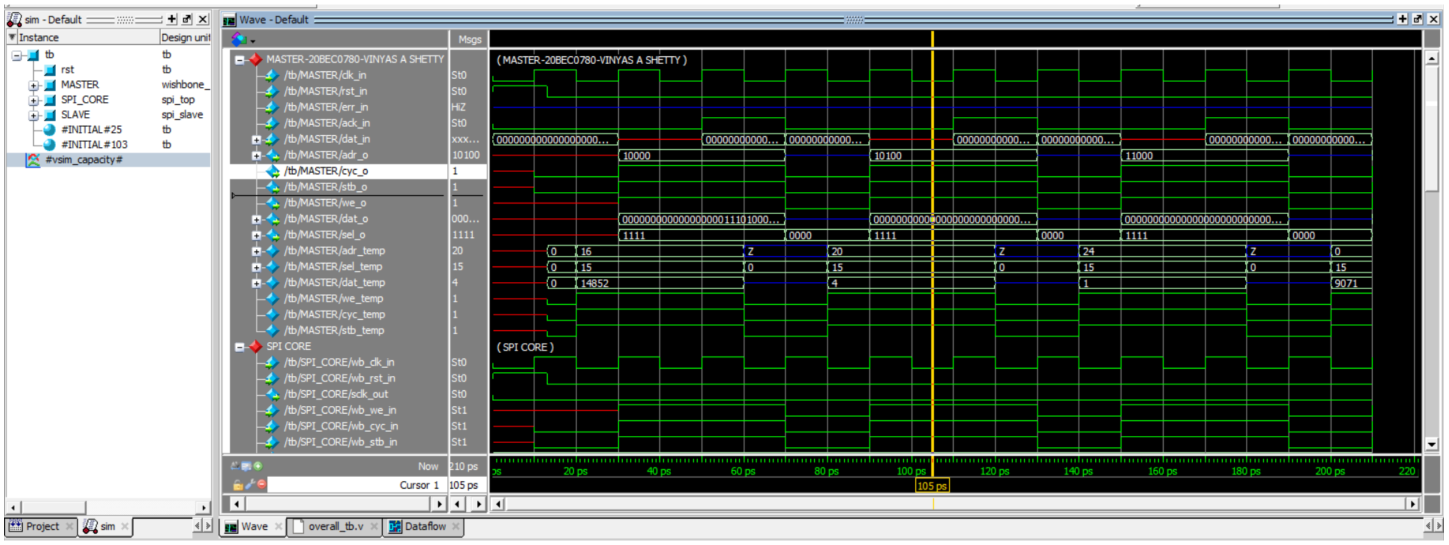
Project - C:/Users/Dell/Desktop/MAVEN/codes_SPI/SPI

Name	Status	Type	Order	Modified
wb_master.v	✓	Verilog	8	06/21/2023 03:31:14 ...
spi_topmodule.v	✓	Verilog	7	06/21/2023 03:31:14 ...
SPI_shiftreg_tb.v	✓	Verilog	5	06/21/2023 03:31:14 ...
clk_gen_tb.v	✓	Verilog	1	06/21/2023 03:31:12 ...
clk_gen_rtl.v	✓	Verilog	0	06/21/2023 03:29:56 ...
spi_defines.v	✓	Verilog	3	06/21/2023 03:31:12 ...
spi_slave.v	✓	Verilog	6	06/23/2023 10:43:09 ...
SPI_shift_reg.v	✓	Verilog	4	06/21/2023 03:31:12 ...
overall_tb.v	✓	Verilog	2	06/23/2023 10:43:09 ...

All the verilog files compiled together for the final verification using final testbench

WAVEFORM:

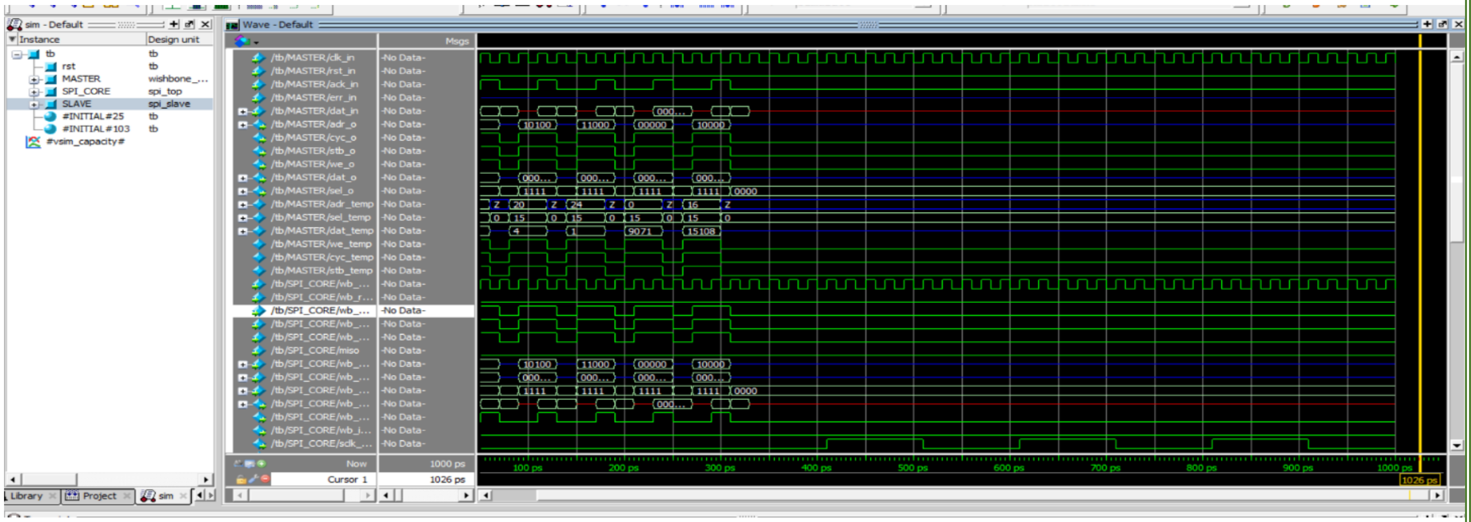
MASTER

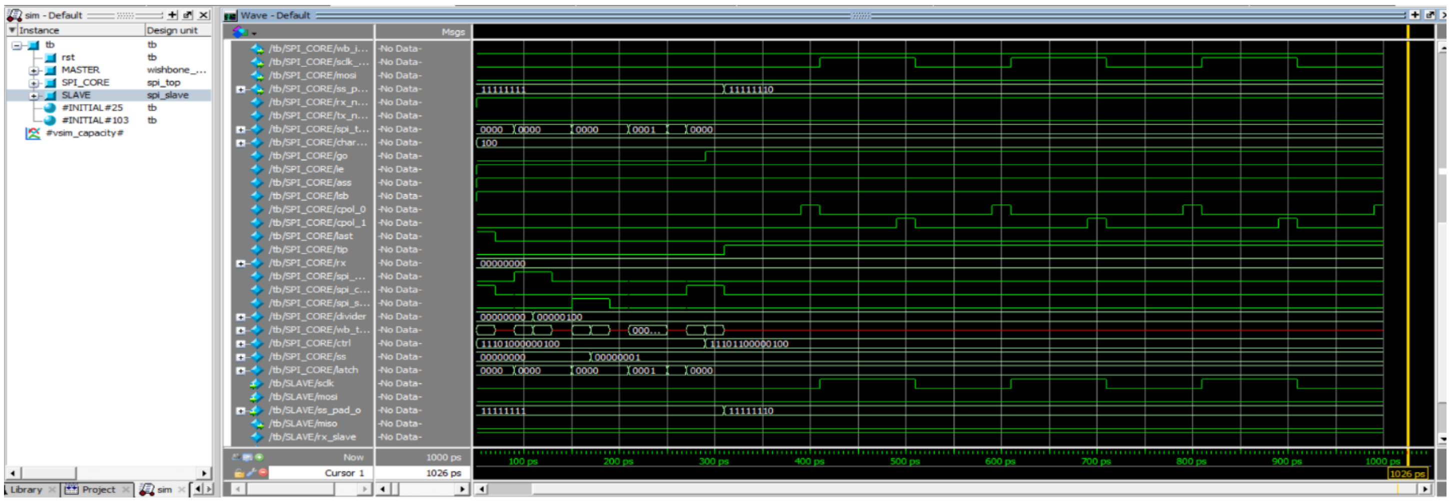


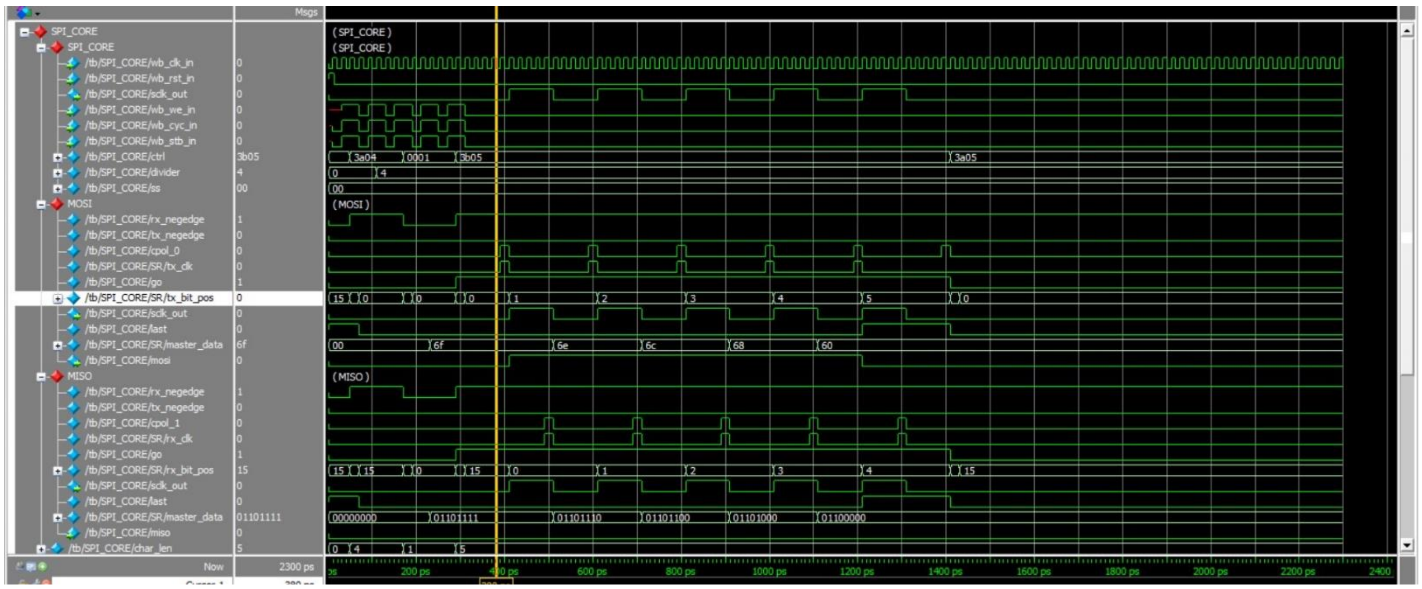
Instance	Design unit
tb	tb
rst	tb
MASTER	wishbone_
SPI_CORE	spi_top
SLAVE	spi_slave
#INITIAL#25	tb
#INITIAL#103	tb
#vsim_capacity#	

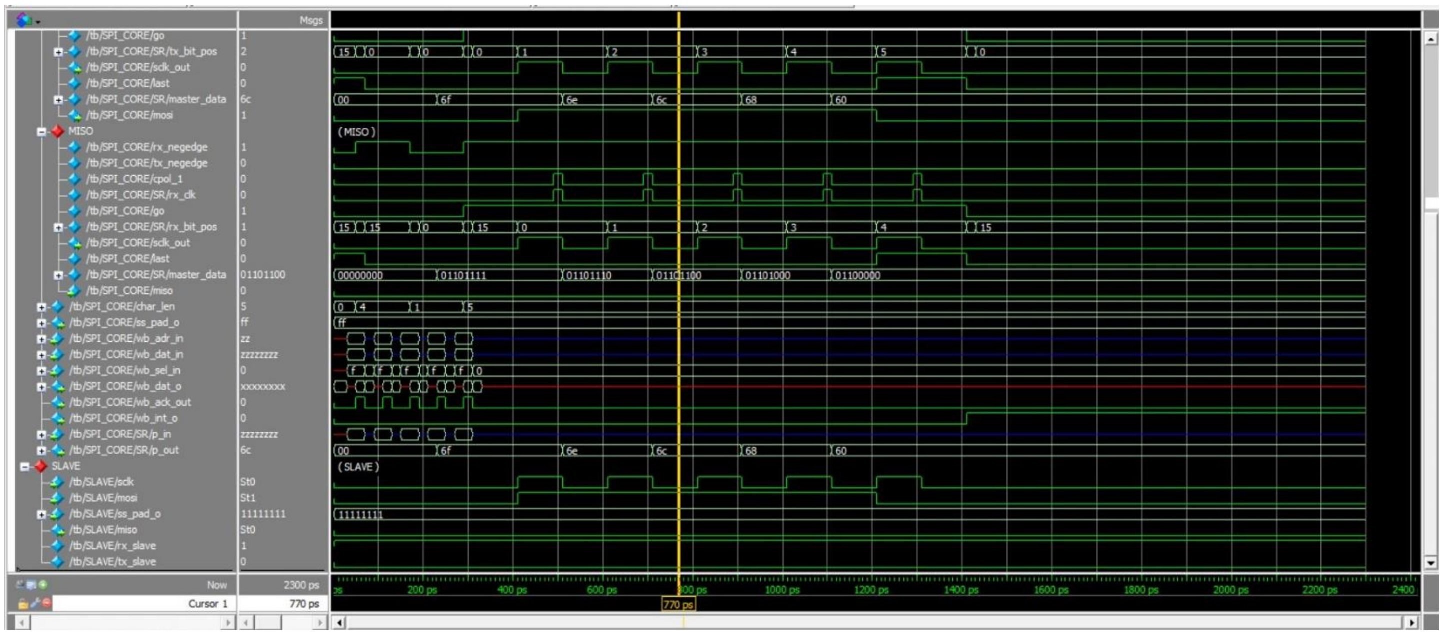
	Msgs
MASTER-20BEC0780-VINYAS A SHETTY	
/tb/MASTER/clk_in	St0
/tb/MASTER/rst_in	St0
/tb/MASTER/err_in	HiZ
/tb/MASTER/ack_in	St0
/tb/MASTER/dat_in	xxx...
/tb/MASTER/adr_o	10100
/tb/MASTER/cyc_o	1
/tb/MASTER/stb_o	1
/tb/MASTER/we_o	1
/tb/MASTER/dat_o	000...
/tb/MASTER/sel_o	1111
/tb/MASTER/adr_temp	20
/tb/MASTER/sel_temp	15
/tb/MASTER/dat_temp	4
/tb/MASTER/we_temp	1
/tb/MASTER/cyc_temp	1
/tb/MASTER/stb_temp	1
SPI CORE	
/tb/SPI_CORE/wb_clk_in	St0
/tb/SPI_CORE/wb_rst_in	St0
/tb/SPI_CORE/sclk_out	St0
/tb/SPI_CORE/wb_we_in	St1
/tb/SPI_CORE/wb_cyc_in	St1
/tb/SPI_CORE/wb_stb_in	St1

SPI CORE + MOSI +MISO +SLAVE









```
                                  Msgs
/tb/SPI_CORE/go                   1
/tb/SPI_CORE/SR/tx_bit_pos        2
/tb/SPI_CORE/sclk_out             0
/tb/SPI_CORE/last                 0
/tb/SPI_CORE/SR/master_data       6c
/tb/SPI_CORE/mosi                 1
MISO
/tb/SPI_CORE/rx_negedge           1
/tb/SPI_CORE/tx_negedge           0
/tb/SPI_CORE/cpol_1               0
/tb/SPI_CORE/SR/rx_clk            0
/tb/SPI_CORE/go                   1
/tb/SPI_CORE/SR/rx_bit_pos        1
/tb/SPI_CORE/sclk_out             0
/tb/SPI_CORE/last                 0
/tb/SPI_CORE/SR/master_data       01101100
/tb/SPI_CORE/miso                 0
/tb/SPI_CORE/char_len             5
/tb/SPI_CORE/ss_pad_o             ff
/tb/SPI_CORE/wb_adr_in            zz
/tb/SPI_CORE/wb_dat_in            zzzzzzzz
/tb/SPI_CORE/wb_sel_in            0
/tb/SPI_CORE/wb_dat_o             xxxxxxxx
/tb/SPI_CORE/wb_ack_out           0
/tb/SPI_CORE/wb_int_o             0
/tb/SPI_CORE/SR/p_in              zzzzzzzz
/tb/SPI_CORE/SR/p_out             6c
SLAVE
/tb/SLAVE/sclk                    St0
/tb/SLAVE/mosi                    St1
/tb/SLAVE/ss_pad_o                11111111
/tb/SLAVE/miso                    St0
/tb/SLAVE/rx_slave                1
/tb/SLAVE/tx_slave                0
                           Now    2300 ps
                      Cursor 1    770 ps
```