SPI
(Serial Peripheral Interface)
Design Procedure Report

Part I: Design Process

a. *Master:*

First, we assign **clk** to **SCLK**.
If SPI module is **reset**, master is put in the idle state, puts all slaves also in idle state, and waits for a **start** signal to proceed in data transmission process.
Otherwise, it starts the transmission provided that **start** signal is 1:

i. Set **masterDataToSend_temp** to equal **masterDataToSend**.
j. Check **slaveSelect** to make the selected slave **CS** equals 0.
k. Repeat the following 8 times:
    - Writing in the **MOSI** at positive edge of **SCLK** the ith bit in **masterDataToSend_temp**, then at the negative edge, it reads from **MISO** and sets this data to ith bit in **masterDataReceived**.
    - Incrementing **i** as it starts from 0.
l. Set its **state** to **idle** waiting for another **start** signal to repeat the process.

b. *Slave:*

Same as master,
If **reset** -> module is put in the idle state, waiting for the **CS** signal to be 0.
Otherwise, it starts transmission if and only if **CS** signal is set to 0:

i. Set **masterDataToSend_temp** to equal **masterDataToSend**.
j. Check **slaveSelect** to make the selected slave **CS** equals 0.
k. Repeat the following 8 times:
    - Writing in the **MISO** at positive edge of **SCLK** the ith bit in **slaveDataToSend_temp**, then at the negative edge, it reads from **MOSI** and sets this data to ith bit in **slaveDataReceived**.
    - Incrementing **i**.
l. Set the slave **state** to **idle** waiting for **CS** to be 0 again.

c. *Master_TB:*

At the beginning, set **reset** to 1 and initialize **start** with 0 to put the module master in the idle state.

After 1 clock cycle, we set **reset** to 0, and **start** to 1, so that the data exchange occurs, repeating for 3 times using 3 test cases (8'b01010011, 8'b00111100, 8'b11111111) and printing whether it is a success or not.

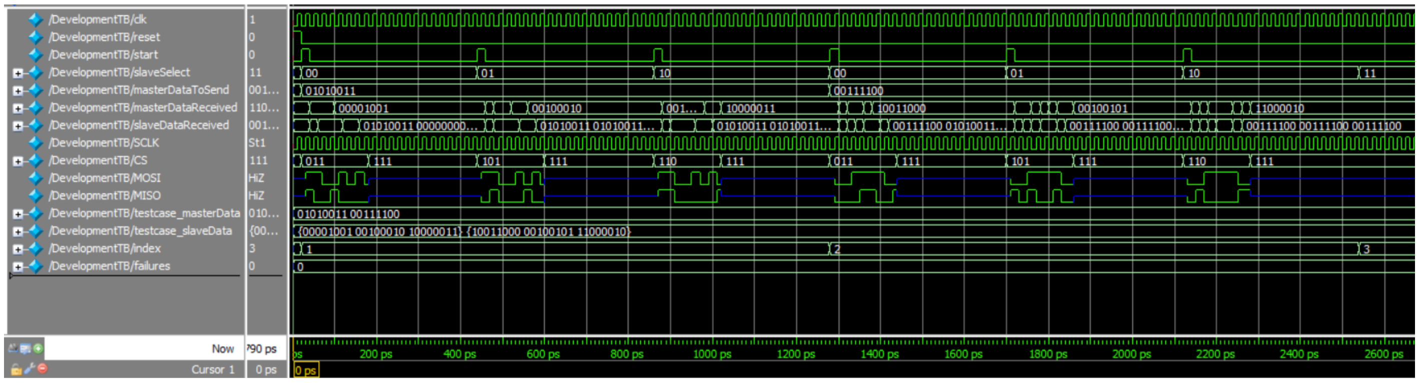
d. *Slave_TB:*

At the beginning, set **reset** to 1 and initialize **CS** with 1 to put the module slave in the idle state.
After 1 clock cycle, we set **reset** to 0, and **CS** to 0, so that the data exchange occurs, repeating for 3 times using 3 test cases (8'b10101010, 8'b11110000, 8'b 00110011) and printing whether it is a success or not.

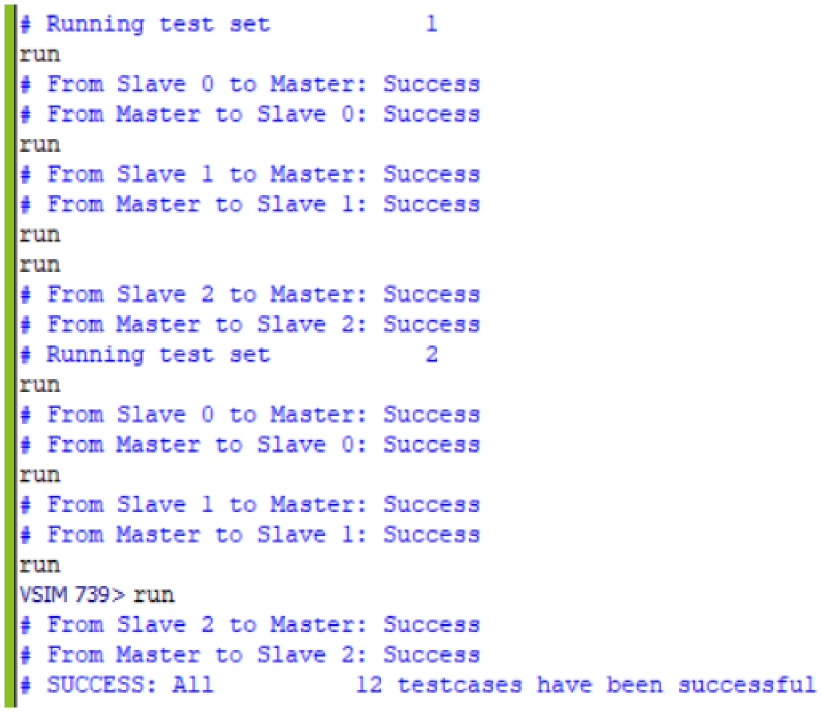
Part II: Simulation Results

a. Development_TB (SPI):

- Waveform:



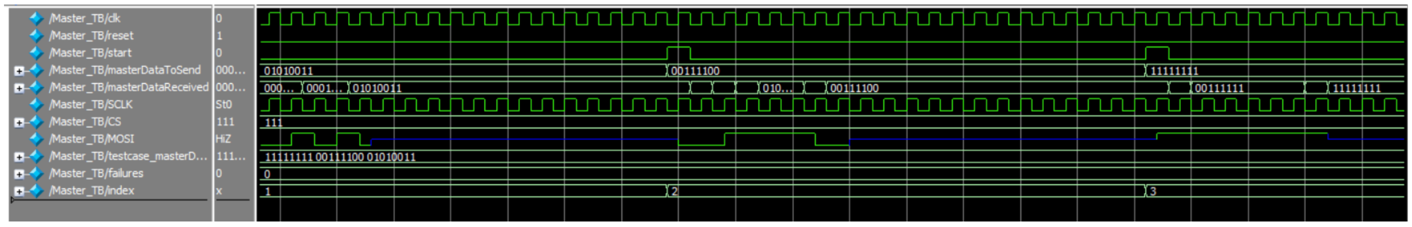
- Output:



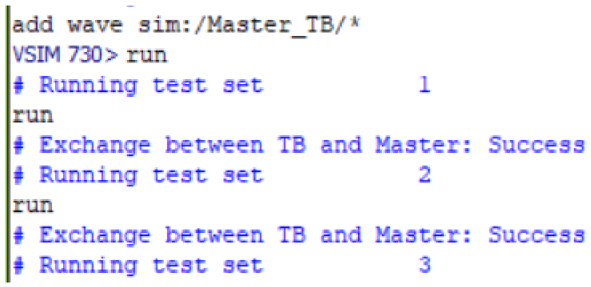
```
# Running test set           1
run
# From Slave 0 to Master: Success
# From Master to Slave 0: Success
run
# From Slave 1 to Master: Success
# From Master to Slave 1: Success
run
run
# From Slave 2 to Master: Success
# From Master to Slave 2: Success
# Running test set           2
run
# From Slave 0 to Master: Success
# From Master to Slave 0: Success
run
# From Slave 1 to Master: Success
# From Master to Slave 1: Success
run
VSIM 739> run
# From Slave 2 to Master: Success
# From Master to Slave 2: Success
# SUCCESS: All          12 testcases have been successful
```

b. Master_TB:

- Waveform:



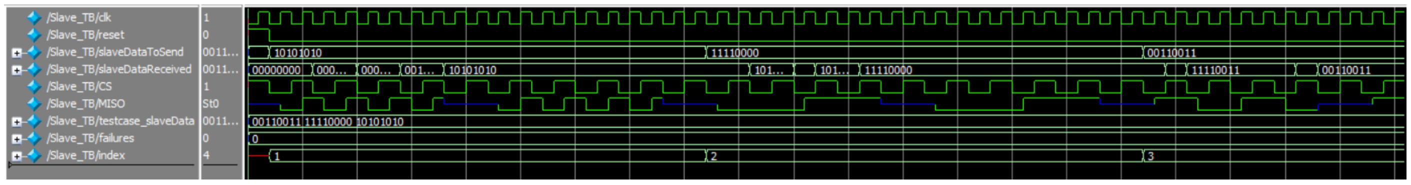
- Output:



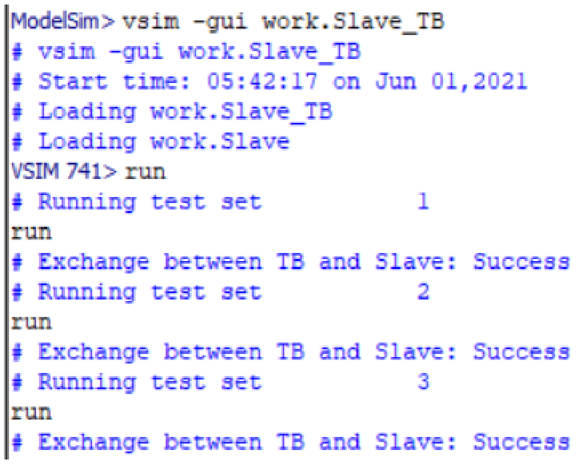
```
add wave sim:/Master_TB/*
VSIM 730> run
# Running test set            1
run
# Exchange between TB and Master: Success
# Running test set            2
run
# Exchange between TB and Master: Success
# Running test set            3
```

c. *Slave_TB:*

- Waveform:



- Output:



```
ModelSim> vsim -gui work.Slave_TB
# vsim -gui work.Slave_TB
# Start time: 05:42:17 on Jun 01,2021
# Loading work.Slave_TB
# Loading work.Slave
VSIM 741> run
# Running test set            1
run
# Exchange between TB and Slave: Success
# Running test set            2
run
# Exchange between TB and Slave: Success
# Running test set            3
run
# Exchange between TB and Slave: Success
```