**SPI**

Team #17

Master

**Master interface** :

**Start** : when start equal to one -> start the program if the value of the start change before the end of transmission of the previous data it doesn't effect on the working of master .

**Reset** : It reset master data (masterdatatosend).

**Slaveselect** , **CS**: slaveselect is a 2-bits input , CS is an 3-bits output when slaveselect = 00 -> CS=011 to make the first slave connected to master to receive data from master when slaveselect = 01 -> CS=101 to make the second slave connected to master to receive data from master when slaveselect = 10 -> CS=110 to make the third slave connected to master to receive data from master otherwise -> CS=111 slaves doesn't receive any data from master.

And CS also control the transmission operation of slave as when the master finish, CS change to 3 bits 111 and when the start is 1 when we wanna send a new data to the slave, we set CS again.

**Masterdatatosend** : it receive data from outside (user).

**Masterdatatoreceived** : its show materdatatosend value to outside (user).

**MOSI**: master output slave input (transmit data from master to slave).

**MISO** : master input slave output (transmit data from slave to master )

**CLK**,**SCLK** : its master clock that assigned to SCLK slave clock.

**Master data transmission:**

The master send the clock signal and select the required slave by enabling the CS signal (CS=0) I.e. The master sends a 0 signal to select the slave. both master and slave their data sampled on negative edge of the clock and shifting on the positive edge of the clock, MOSI transmit data from master to slave , MISO transmit data from slave to master .

**Master data transmission logic**

First we make an integer counter c and initialize it with a number bigger than 9 and at the first start of the master we make it equal zero or at the starting of the new operation of new data(while shifting of data we increment c with one until c equal to 8 this mean that all data are transmit from master to slave) .

Working of master : if start is equal to one master is enabled to receive data and transmit it , select which slave will work , if reset is equal to one -> masterDatasend will be equal to zero . ,at the positive edge of the clock if (c==0&&start==1) masterdatatosend will recive data ,at positive edge of the clock data will shifted out until transmit all data (8bits) , at negative edge master will receive data from slave until (8bits),master will wait until start equal one again and receive new data.

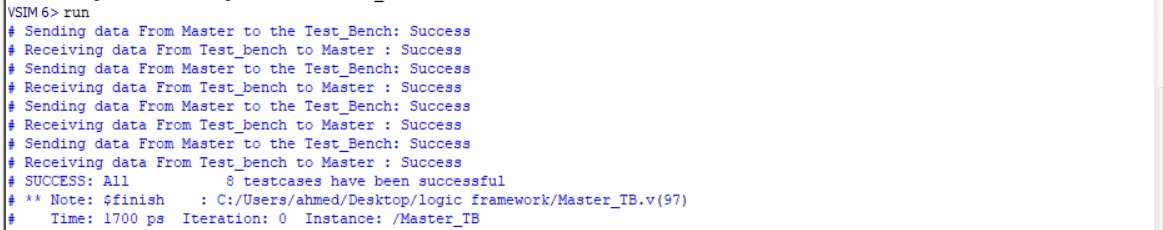
**Master testbench:**

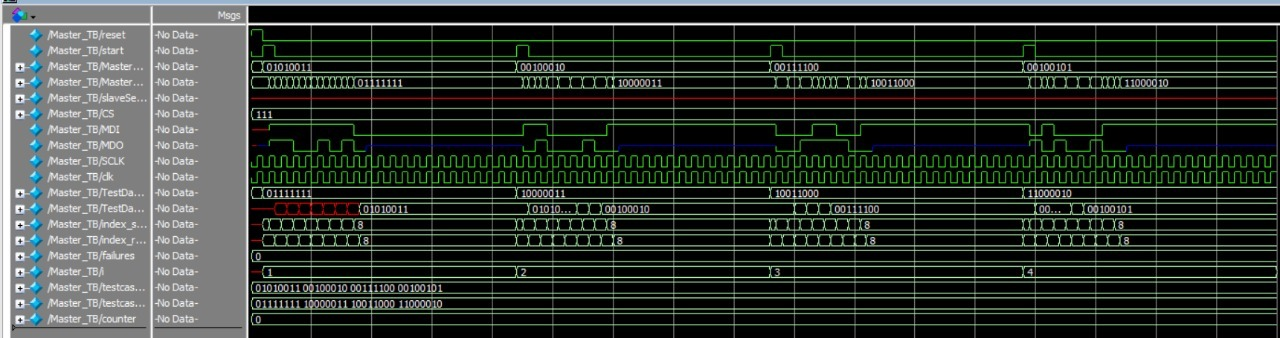
All cases are succeeded.

In this test bench, we made a register that receive and send the data from and to master.

At every rising edge we send the data from the master to this test bench.

At every falling edge we make a variable that send a data to master instead of slave.





**Salve interface** :

sDTSend : (slave data to send ) its received data from master

slaveDatarecieved : its show sDTSend value to outside (user).

SCLK: received clock signal from master clock.

CS: its an input if CS=0, slave receive data from master otherwise Slave doesn't receive data from master.

SDI: slave input transmit data from master to slave.

SDO: slave output transmit data from slave to master.

Reset : make sDTSend to be equal zero .

**Slave data transmission:**

The slave receive the clock signal from master and receive CS signal from master , slave their data sampled on negative edge of the clock and shifting on the positive edge of the clock, SDI transmit data from master to slave , SDO transmit data from slave to master .

**Slave data transmission logic:**

First we make an integer counter c and initialize it with zero logic ( while shifting of data we increment c with one until c equal to 8 this mean that all data are received from master (8bits)) .

Working of slave :,at the positive edge of the clock if (CS==0) slaveDataToSend will receive data , at positive edge data will shifted out until transmit all data (8bits) , at negative edge slave will receive data from master until (8bits)

**Slave testbench:**

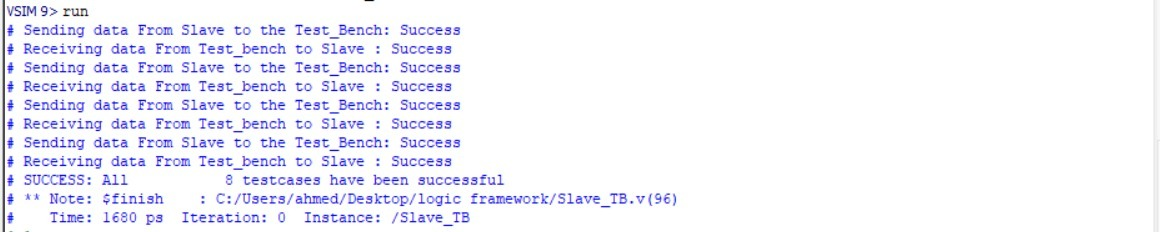
All cases are succeeded.

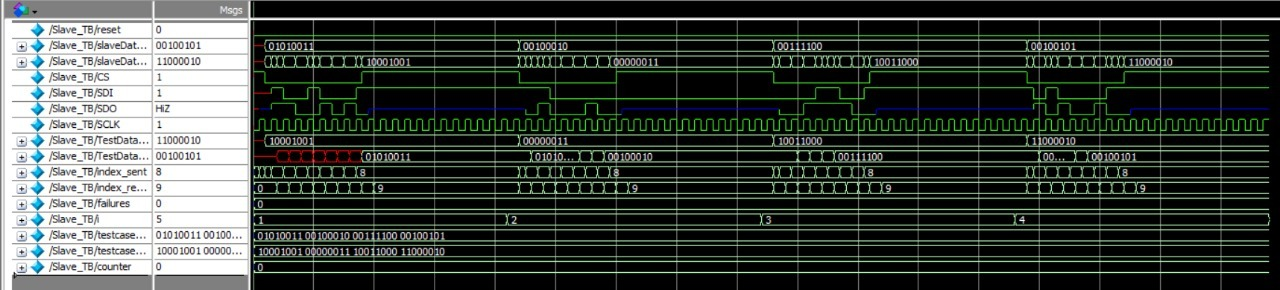
In this test bench, we made a register that receive and send the data from and to slave.

At every rising edge we send the data from the master to this test bench.

At every falling edge we make a variable that send a data to master instead of slave.

And we need to control the CS to start and end the transmission of data from and to slave.





**Development TB:**

All cases are succeeded .

