**Logic Project Report**

**Team #12**

**Master module:**

* ‘done’ reg: indicates the last negative edge in the transmission.
* ‘done2’ reg: indicates the last positive edge in the transmission.
* ‘count’ reg: counter from 0 to 8 to indicate the 8 cycles of transmission.
* ‘in’ reg: temporary register to hold ‘MISO’ signal at the negative edge of the serial clock until it’s registered in ‘masterDataReceived’ at the positive edge of the serial clock.
* ‘data’ reg: internal register to hold ‘masterDataToSend’ signal, so that we can shift the data to be send from the master.
* ‘enable’ reg: works as active high enable to ‘SCLK’ signal.

Initially: ‘done’, ‘done2’, ‘count’, ‘enable’ are **zero**, CS is **[1, 1, 1]** & masterDataReceived is **[0, 0, 0, 0, 0, 0, 0, 0].**

When the circuit is reseted, it goes to the initial state, as well as, ‘in’ becomes **zero**, ‘MOSI’ signal carries the first bit in the data to be send from the master.

When the ‘masterDataToSend’ signal changes, the ‘data’ reg carries the new signal.

When the counter ‘count’ becomes **8**, ‘done’ reg becomes **1** indicating the last negative edge of ‘SCLK’ signal.

At the negative edge of ‘start’ signal (after 1 cycle from its positive edge), we reset the data to start transmitting and receiving, enable ‘SCLK’ and activate the selected slave.

At the positive edge of ‘SCLK’ (as far as the circuit is not in the reset state), we insert the ‘MISO’ signal (‘in’ reg) in ‘masterDataReceived’.

At the negative edge of ‘SCLK’ (as far as the circuit is not in the reset state and the circuit is enabled), we register the ‘MISO’ signal in the ‘in’ reg, shift out ‘data’ reg, ‘increment ‘count’ counter and update ‘MOSI’ signal (we don’t update it after last bit to be transmitted).

**NOTE:** according to the timminig diagram of mode 1, ‘MOSI’ signal is updated after the negative edge of ‘SCLK’, so the master shift out the data after the negative edge of ‘SCLK’.

**Master\_TB module:**

* ‘index’ integer counter from 0 to 8 to know when the start should equal 1.
* ‘slave\_memory’ reg to take the miso signal from it and send it to the master.
* ‘slave\_received\_data’ reg to receive the mosi signal from the master.
* Initially: clk=0 , reset=1, slaveSelect= 2'b00 (to know which slave should the master deal with) , masterDataToSend = 8'b01101001 , slave\_memory= 8'b11011010

slave\_received\_data= 8'b00000000 , index=0.

* At the negative edge of start and index>2 (it means the start was =1 and became =0)

We make miso signal equal the zero bit of slave\_memory.

* With the negative edge of the clk and (index not equal one and smaller than nine)

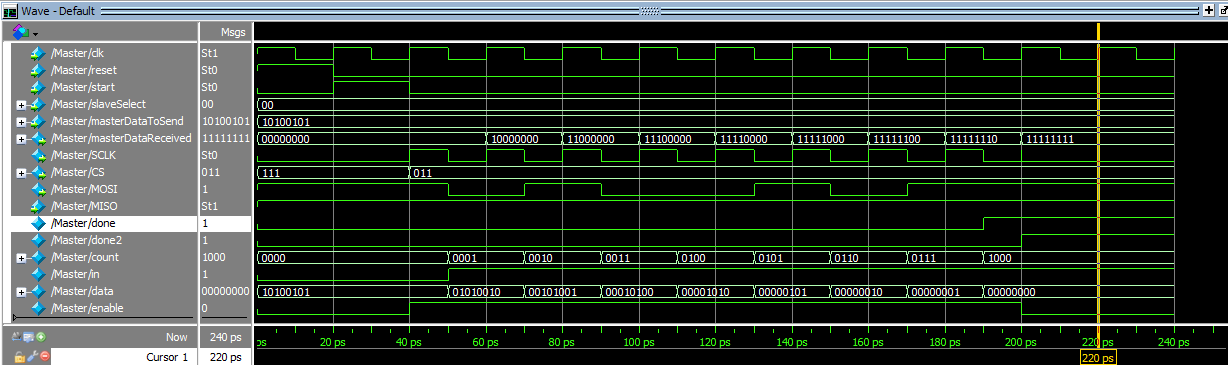
We make (start and reset)=0 and increment the index by 1.

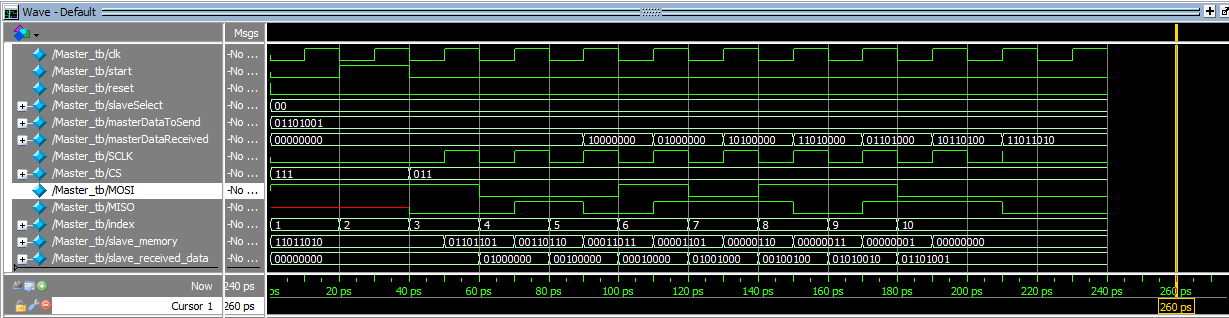
If (index =1) make the start =1 and increment the index by 1.

* With the positive edge of SCLK make the miso signal equal the zero bit of slave\_memory then shift slave\_memory right.
* With the negative edge of SCLK and (index > 1 && index <9) (To avoid first negative edge at time=0 && last negative edge when SCLK goes back to ZERO)

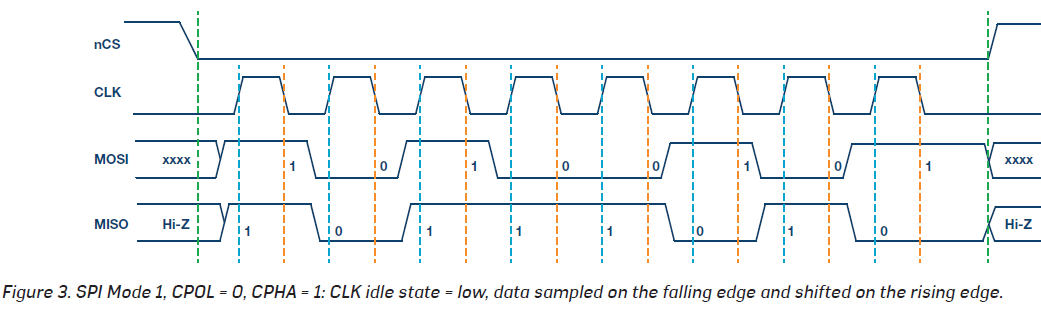
Make the bit number seven of slave\_received\_data = MOSI signal the shift slave\_received\_data

Right.

Master



Master\_tb



­Timming diagram of mode 1

**Slave module:**

* ‘index’ integer: counter from 0 to 8 to indicate the 8 cycles of transmission.
* ‘flag’ reg: to avoid the first falling edge before the start of receiving.

Initially: ‘MISO’ is ‘1'bz’.

When the circuit is reset **slaveDataReceived** and **index** becomes **zero**.

When the ‘**slaveDataReceived** signal changes, the ‘**index’** integer becomes zero.

When the counter ‘**index’** becomes **8**, itindicates that it is the last negative edge of ‘SCLK’ signal.

At the negative edge of ‘start’ signal (after 1 cycle from its positive edge), we reset the data to start transmitting and receiving, enable ‘SCLK’ and activate the selected slave.

At the positive edge of **‘SCLK’** we increment the **index** then we check **CS** (to know if the slave is selected by the master), if it is selected (**CS** = 0) we set the **‘MISO’** signal to the corresponding bit (according to the **index** integer) of **slaveDataToSend**, if it is not selected, we make its **MISO** a high impedance (**z**).

At the negative edge of **‘SCLK’** (as far as the Slave is Selected), we shift **slaveDataReceived** a single bit to the right and assign **MOSI** to the 7th bit.

**NOTE:** We added flag reg because the program starts with a fallen edge.

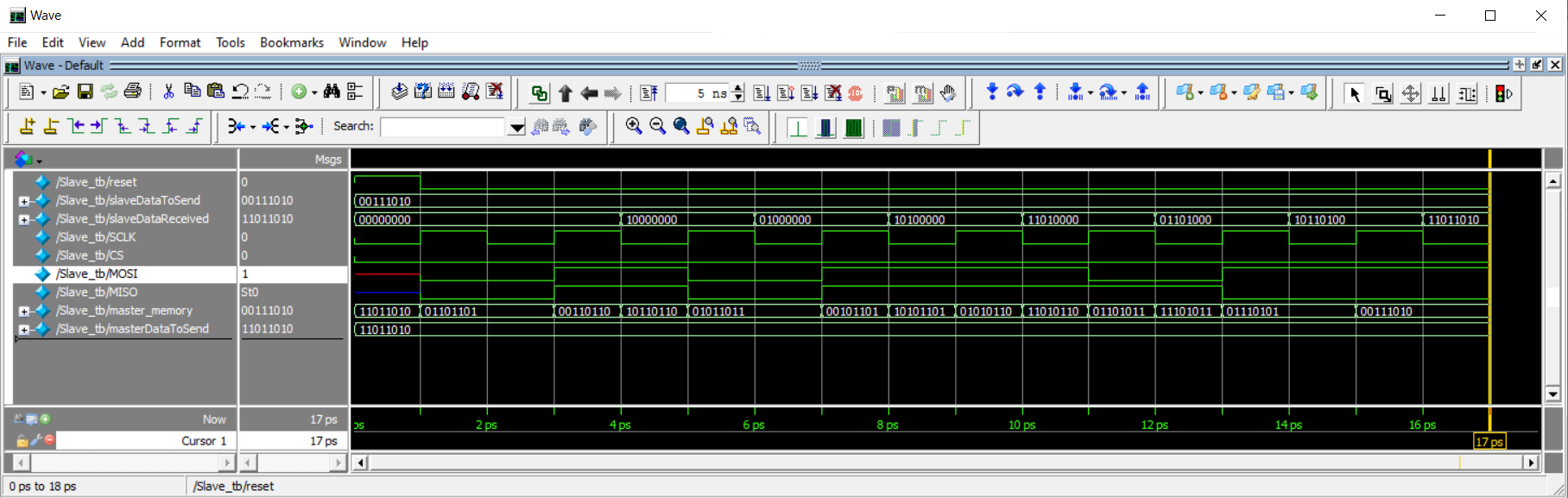
**Slave\_tb:**

* Reg 'master\_memory' is the data in the master that will be sent to Slave and will store the data that come from slave in it.

Steps:

1. Initial reset with one to make the slaveDataReceived (Slave memory) equals to zero and after 1delay it set to be zero.
2. Initial master\_memory and slaveDataToSend, and make masterDataToSend equals to master\_memory.
3. At positive edge the least significance digit in master\_memory sent to slave by MOSI wire and shift the memory right (Shifting). In slave the same process will done and sent data to master by MISO wire.
4. At negative edge the master\_memory reads data that comes from slave by MISO wire and store it in the most significance digit on the master memory (Sampling).In slave, the data reads by MOSI wire.

Note: the if condition used to make sure that the signal that will be read is NOT ( X signal ) if exist.

1. After 8 cycles we check if the master\_memory equals to the slaveDataToSend and the slaveDataReceived equals to masterDataToSend. If equals, that mean the slave and the master send and receive from each other correctly and print Success, if NOT print Failure.

Graphical user interface, application

Description automatically generatedSlave\_tb

Development\_tb