

Cairo University Computer Engineering Faculty of Engineering Frist Year

Serial peripheral Interface

Dr. Ihab Talkhan

Omar mahammed

Mostafa wael

Menna Allah Ahmed Ali Mohammed

Nada Elsayed Mohammed

**2020**

# Design process:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MODE | CPOL | CPHA | Data sampling on | Data shifted out |
| 0 | 0 | 0 | Rising | Falling |
| 1 | 0 | 1 | Falling | Rising |
| 2 | 1 | 1 | Falling | Rising |
| 3 | 1 | 0 | Rising | Falling |

## Master

### Ports:

1. Master has ports to be used by the test bench (Microcontroller itself) as:

input clk: general clock input to the master.

input reset: it is an indicator to assign the transmitted data register to data in and assign the receiving data to 0.

input reg [1:0] MODE: it tells the master the mode it must work on it.

input reg [7:0] data\_in: the transmitted register is assigning with this data, then the transmitted register can shift this data out to the slave using MOSI.

input reg [1:0] CS: this tells the master what exactly slave it will transmit or receive the data to or from it.

input [1:0] RW: this tells the master whether it read the data from slave (10) or write the data on slave (01) or both (11) or nothing (00).

output wire [7:0] data\_out: this data that came from slave and stored in receiving data register, this wire is assigning to the receiving data register when it receives 8 bits.

1. Ports to be connected to salves:

input MISO: this port that the master can read the data from slave on it.

output reg MOSI: this port that the master can write the data to slave on it.

output CS1bar: that indicator if slave 1 is active or no.

output CS2bar: that indicator if slave 2 is active or no.

output CS3bar: that indicator if slave 3 is active or no.

output wire sclk: this the clock signal that master send to the slave.

output wire sreset: this indicator whether reset the slave or no, and this important for test bench.

output wire [1:0] sMODE: this input to slave to tells it what mode it should work on, and this important for test bench.

### Internal containers:

1. Flags:

reg start\_writting: it is a flag that is force master to neglect the edge that must shifted the data out till it do the sampling data on first.

1. Reading and receiving containers:

integer RX\_bit\_count: it is a counter that is count the number of bits had been read in the master from the slave.

reg [7:0] RX\_temp\_byte: it is temporary shift register that receives the data from the MISO port.

reg [7:0] RX\_byte: it is the main register that store data after 8 bits these would shifted to the temporary register (RX\_temp\_byte).

reg RX\_done: it is an indicator that indicate when the main receive register have already data (RX\_byte), this used in transform the data from the (RX\_byte) to the (data\_out) that’s out from master.

1. Writing and transmitting containers

reg [2:0] TX\_bit\_count: it is a counter that is count the number of bits the master wrote it in the slave.

reg [7:0] TX\_temp\_byte: it is temporary shift register that transmit the data to the MOSI port.

reg TX\_done: it indicates if the bit is shifted out or not.

### Logic:

* The master works in mode 3 and makes a relation between the sclk and clk to allow it to do so.
* The master has 4 modes depends on the clock polarity and clock phase, If the mode is mode 0 or 3 then the value sclk will be clk bar, else the value of sclk is equal to clk, this is to keep enter in the positive edge to sample the data in.

It uses internal flag call (start\_writting) to ensure that the leading edge is the rising edge. Hence, it avoids any corrupted data.

* It has two statues working simultaneously which are reading statue -for receiving- and writing statue -for transmitting- as follows:
* Reading mode:

Data enters through the MISO bit by bit, the master collects and shifts the data in a temporary container (RX\_temp\_byte) until the bit counter (RX\_bit\_count) reaches 8 bits then assigns this byte to (RX\_byte) and assigning the (RX\_done) flag to true.

When the (RX\_done) is true then, data is shifted out through (data\_out) port to the microcontroller.

* Writing mode:
  + - 1. Data enters as a whole byte through (data\_in) port and is saved in a temporary container (TX\_temp\_byte), every clock cycle one bit of this byte is shifted out through the MISO until the bit counter (TX\_bit\_count) reaches 8 bits then, the (TX\_done) flag is assigned to true.
      2. When the (TX\_done) is true then, the Master will read a new byte and start the whole operation again.

## Slave

### Ports:

1. Salve has ports to be used by the test bench (e.g. sensor, modules) as:

input [7:0] data\_in: the transmitted register (T\_data) is assigning with this data, then the transmitted register can shift this data out to the slave using MOSI.

output reg [7:0] data\_out: the data out is assigning to the received data register (R\_data).

1. Ports to be connected to master:

input clk: the clock signal the master sends it.

input MOSI: is the port that slave can receive the data master had sent on it.

output reg MISO: is the port that slave can send the data to master on it.

input CS: it is an indicator that indicates whether slave is active or no.

input [1:0] MODE: it is the mode that slave will work on and it's an input from the master, and this important for test bench.

input reset: this is indicator to reset the slave, and lets the transmitted data assigning to a new data existing in, the current process of sampling and shifting out is stopped, and this important for test bench.

### Internal containers:

1. Flags:

reg entered: it is an indicator that indicates whether the bit is transmitted or not, to make sure the shifting out on miso happened first. (because of in testing, we assign MOSI=MISO in slave testbench).

reg done: initialized with “0” as soon as it turned “1” that indicates that slaves received all 8-bits and its data has been shifted out through miso line to the master.

reg is\_read: while it is active low (0), slave continues reading data from data\_in port, it turned “1” as soon as the chip select is activated.

1. Reading and receiving containers:

reg [7:0] R\_data: this is the shift register that receives data from MOSI.

1. Writing and transmitting containers:

reg [7:0] T\_data: this is the shift register transmits data on MISO.

1. Common containers:

integer count: it is counter which counts number of bits that had been received and after 8 bits it assigns the data out to the received data.

### Logic:

* Before the activation of the chip select the data\_in which will be shifted out to the master later is sampling at the positive edge.
* when chip select of one of them is activated the corresponding Slave object starts the processes.
* At the positive edge of the clk:
* the slave samples the data on MOSI line and shifts right the R\_data register then stores the MOSI data >R\_data= {MOSI, R\_data [7:1]} and the count increases.
* the slave transfers LSB bit through MISO line then shifts left the T\_data register -> T\_data= {T\_data [6:0], 1'bx}

## Integration

### Ports:

### it has ports to be used by the test bench (microcontroller and the connected modules) as:

input clk: the general clock signal that sends to the master.

input reset: the reset indicator that sends to the master.

input reg [7:0] data\_in\_to\_master: this is the data in to the master and it will transmit it to the slave.

wire [7:0] data\_out\_from\_master: this is the data received from the slave into the master.

input reg [7:0] data\_in\_slave1: this is the data in to the slave 1 and it will transmit it to the master.

input reg [7:0] data\_in\_slave2: this is the data in to the slave 2 and it will transmit it to the master.

input reg [7:0] data\_in\_slave3: this is the data in to the slave 3 and it will transmit it to the master.

wire [7:0] data\_out\_slave1: this is the data out from the slave 1 that it received it from the master.

wire [7:0] data\_out\_slave2: this is the data out from the slave 2 that it received it from the master.

wire [7:0] data\_out\_slave3: this is the data out from the slave 3 that it received it from the master.

input reg [1:0] CS: this is the chip select indicate the master to the slave that will be read, write or both from, into it.

input [1:0] RW: this indicates the master if it will read from slave, write in it, both or nothing.

input reg [1:0] MODE: this indicates the master the mode that it will work on it.

wire MOSI: this is the port that transmit the data from master to slave.

wire MISO: this is the port that transmit the data from slave to master.

wire MISO1: this is the data out from slave 1, and the controller choose if assign MISO to MISO1 or not depending on the active slave (CS).

wire MISO2: this is the data out from slave 2, and the controller choose if assign MISO to MISO2 or not depending on the active slave (CS).

wire MISO3: this is the data out from slave 3, and the controller choose if assign MISO to MISO3 or not depending on the active slave (CS).

wire CS1bar: this is the output chip select from the master that can active the slave1 or not.

wire CS2bar: this is the output chip select from the master that can active the slave2 or not.

wire CS3bar: this is the output chip select from the master that can active the slave3 or not.

wire sclk: this the output clock from master and send it to the slave.

wire sreset: this the output reset from master and send it to the slave.

wire [1:0] sMODE: this the output mode from master and send it to the slave.

### Internal modules:

1. 1\*Master.
2. 3\*Slave

### Logic:

* It connects the ports of both the Master and the slave together.
* It uses an internal multiplexer (ternary operator) to assign the MISO line to the correct slave according to the CS.

# Test benches:

## Master

It has four test cases in which (MISO) is connected to (MOSI) to check that data is transmitted correctly:

### Test case #1:

Mode to be tested: mode #0

Operation to be done: sending one byte to salve #3 and reading it again.

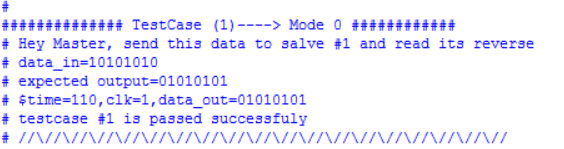
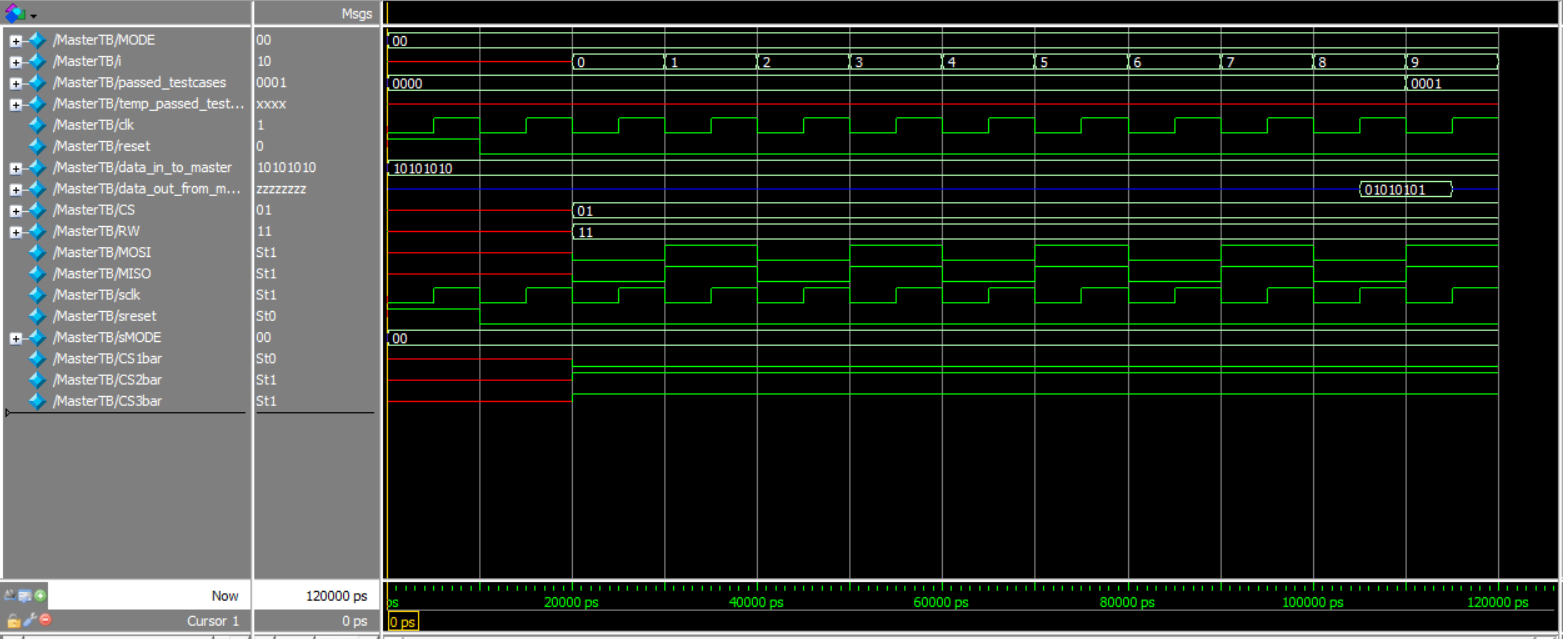
Aim of test case: testing reading and writing data.

Figure Master\_testcase1\_output

Figure Master\_testcase1\_waveform

### Test case #2:

Mode to be tested: mode #1

Operation to be done: sending two bytes to salve #2 and reading them again.

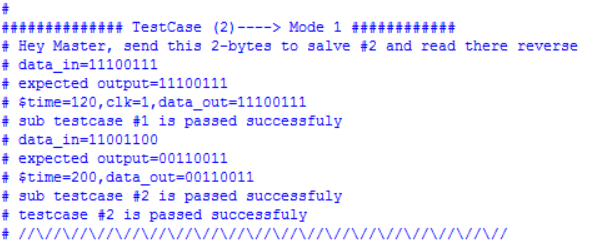
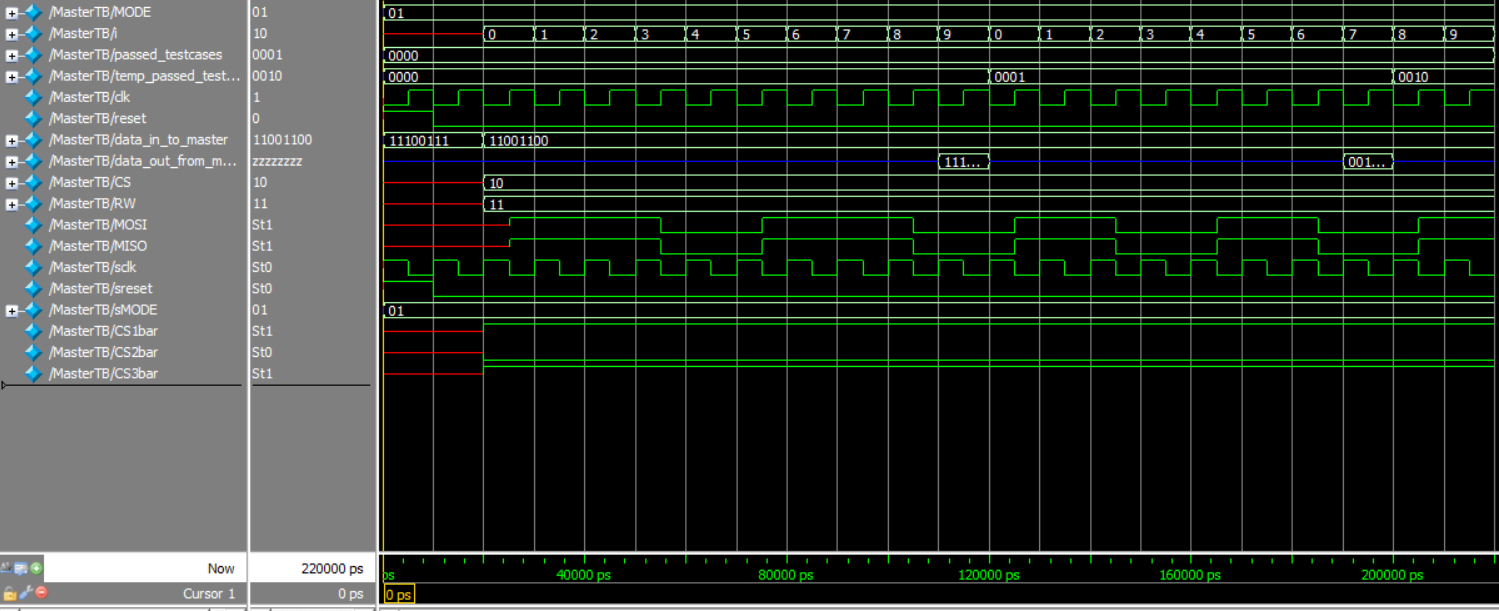
Aim of test case: testing reading and writing multiple of bytes.

Figure Master\_testcase2\_output

Figure Master\_testcase2\_waveform

### Test case #3:

Mode to be tested: mode #2

Operation to be done: sending one byte to salve #1 and reading it again.

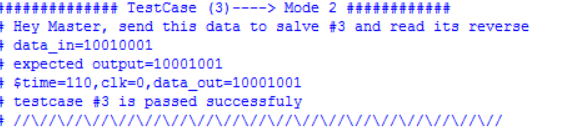
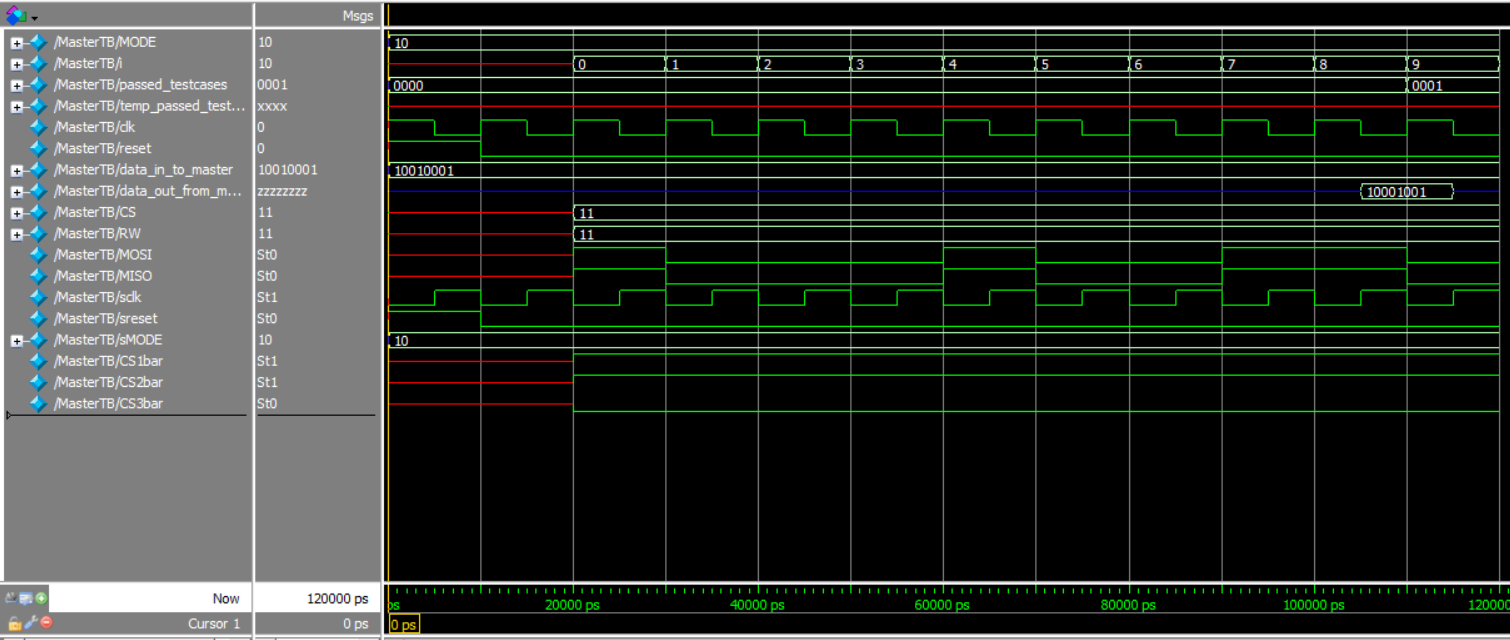
Aim of test case: testing reading and writing data and switching slaves is done correctly.

Figure 6 Master\_testcase3\_waveform

Figure 5 Master\_testcase3\_output

### Test case #4:

Mode to be tested: mode #3

Operation to be done: sending one byte to salve #3 and showing the details of sending these data and read nothing.

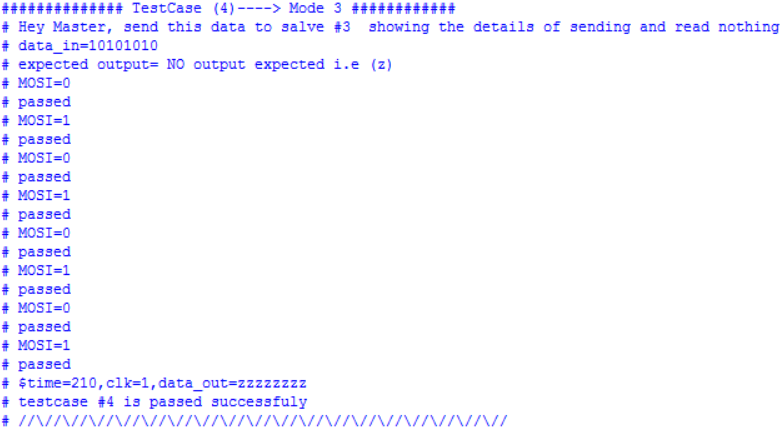
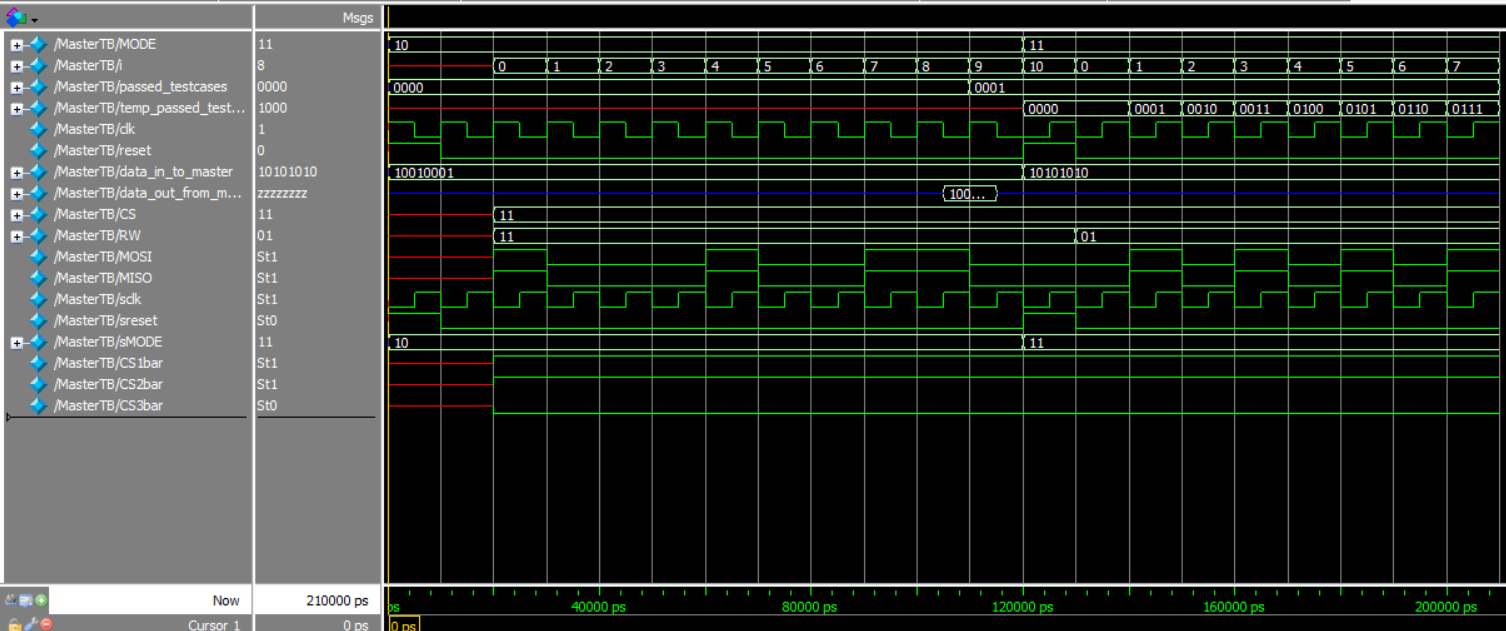
Aim of test case: testing writing data only and that blocking reading functions well

Figure 8 Master\_testcase4\_waveform

Figure 7 Master\_testcase4\_output

### Test cases: 1,2,3,4 followed by each other:

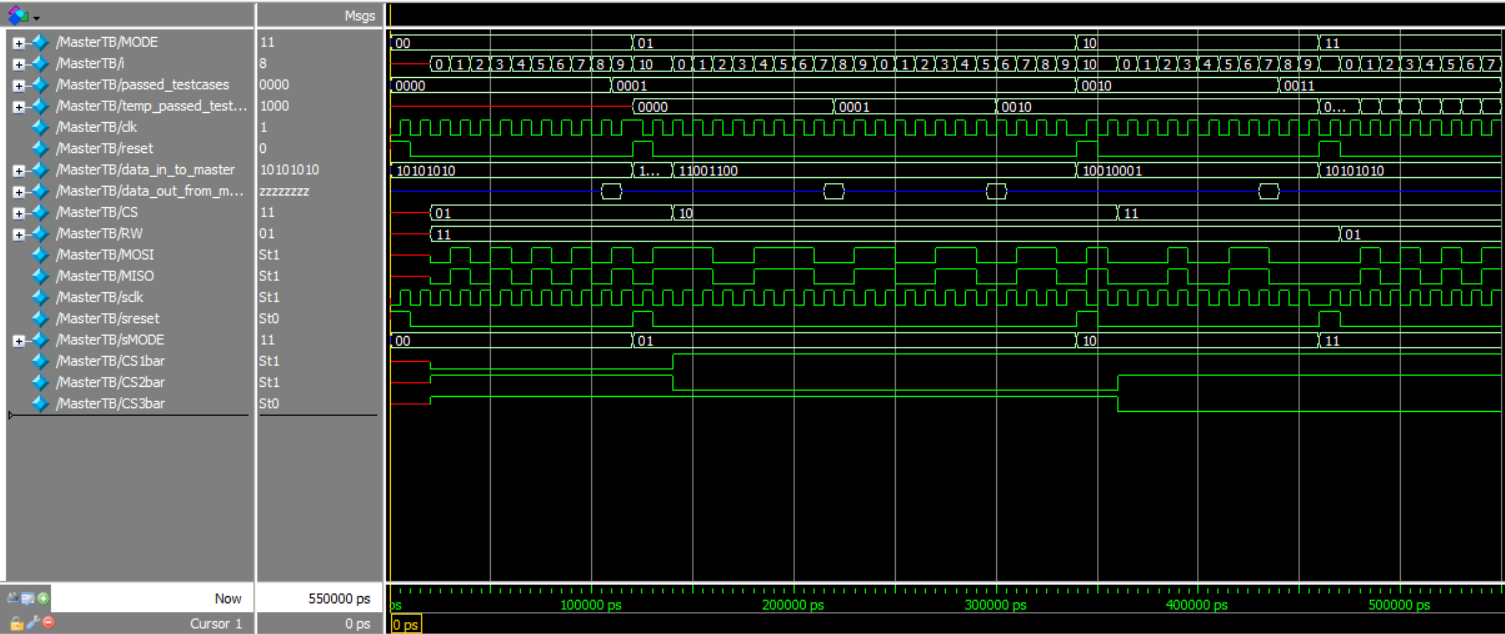


Figure Master\_testcases1,2,3,4\_waveform

## Slave

In all test cases we assign that MOSI = MISO to ensure that the data is transmitted correctly to the master and at the same time the slave receives the data correctly.

### Testcase1: tests MODE ”2”

In this test case the current data\_in in slave =8’b10101010, due to we assign mosi=miso the shifted-out bit from the slave will be transmitted to the slave again through the mosi.

The output is expected to be reversed because of the internal register that shifts out data to mosi begins with MSB then “follows shifting left”.

* T\_data={T\_data[6:0],1'bx}.

But the register which stores data received “follows shifting right”

* R\_data={MOSI,R\_data[7:1]}.

So, expected output-->8’b 01010101.

Exact output--> 8’b 01010101.

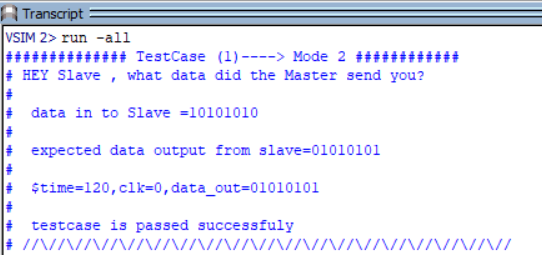


Figure testcase (1) \_wave

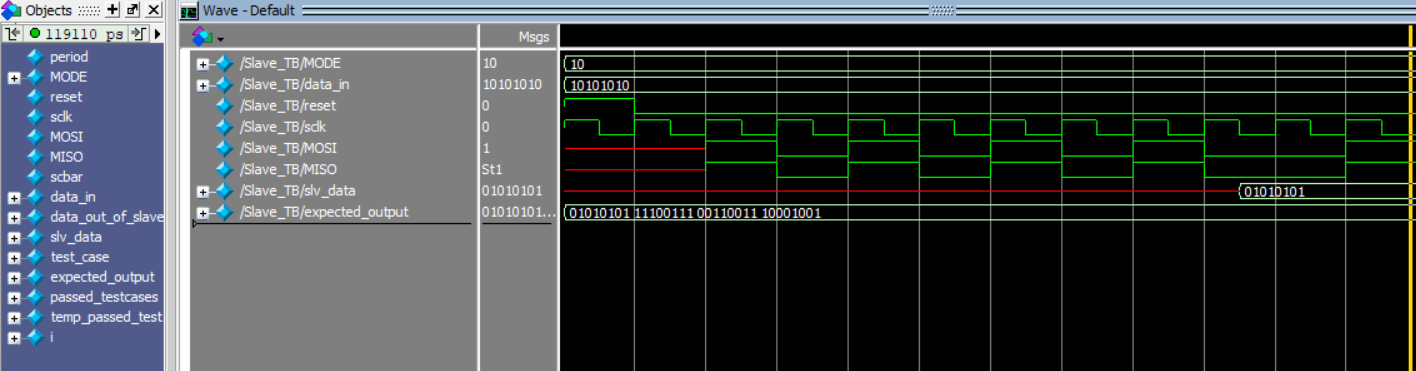


Figure testcase1\_output

### Testcase2: tests MODE ”1”

Current data\_in slave -->8’b 11100111

So, expected output-->8’b 11100111” data is reversed for the same reason shown above”.

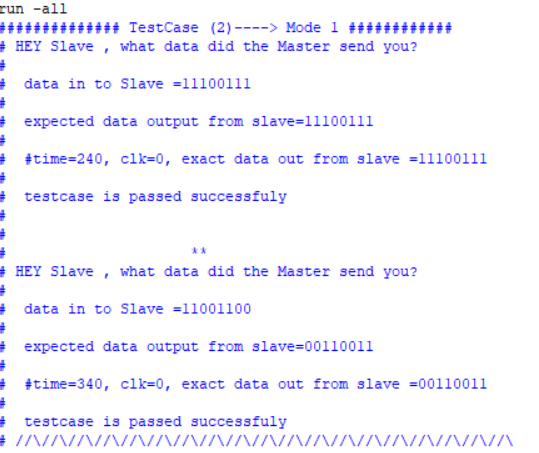
Exact output--> 8’b 11100111.

\*\* At the same test case after complete 8 clk cycle, CS=1.

Slave reads new data\_in =8’b 11001100, then CS=0.

So, expected output-->8’b 00110011” data is reversed for the same reason shown above”.

Exact output--> 8’b 00110011.



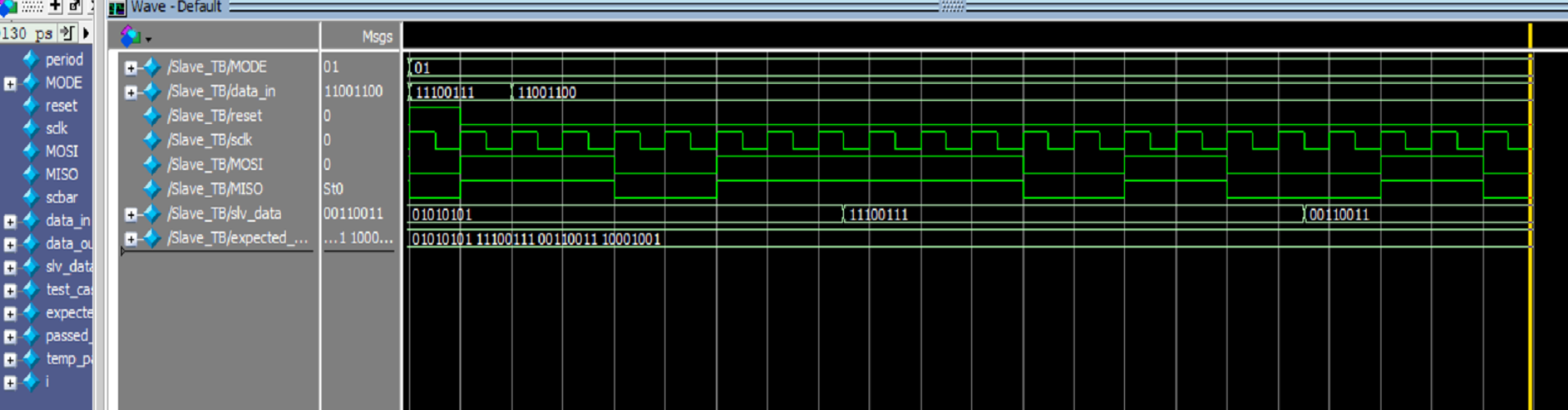
Figure testcase2\_output

Figure testcase (2) \_wave

### Testcase3: tests MODE ”0”

Current data\_in slave -->8’b 10010001

So, expected output-->8’b 10001001” data is reversed for the same reason shown above”.

Exact output--> 8’b 10001001.

### 

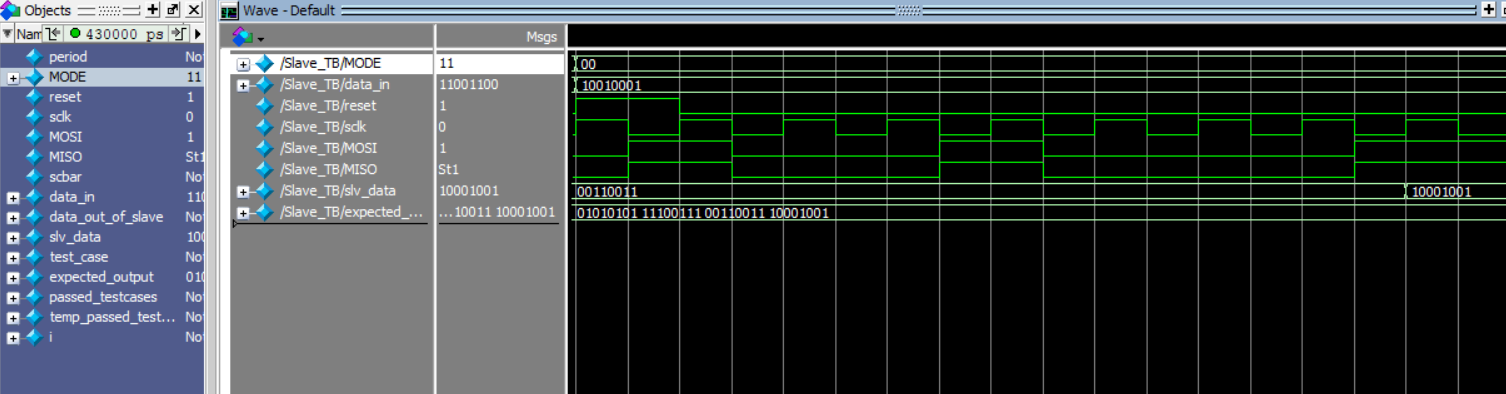
Figure testcase3\_output

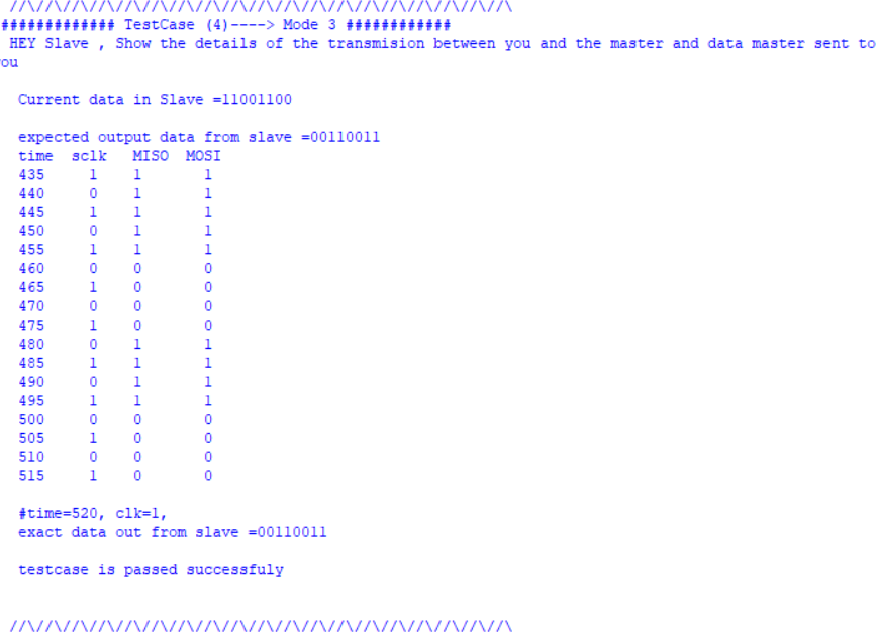
Figure testcase (3) \_wave

### Testcase4: tests MODE ”3”

Current data\_in slave -->8’b 11001100

So, expected output-->8’b 00110011” data is reversed for the same reason shown above”.

Exact output--> 8’b 00110011.



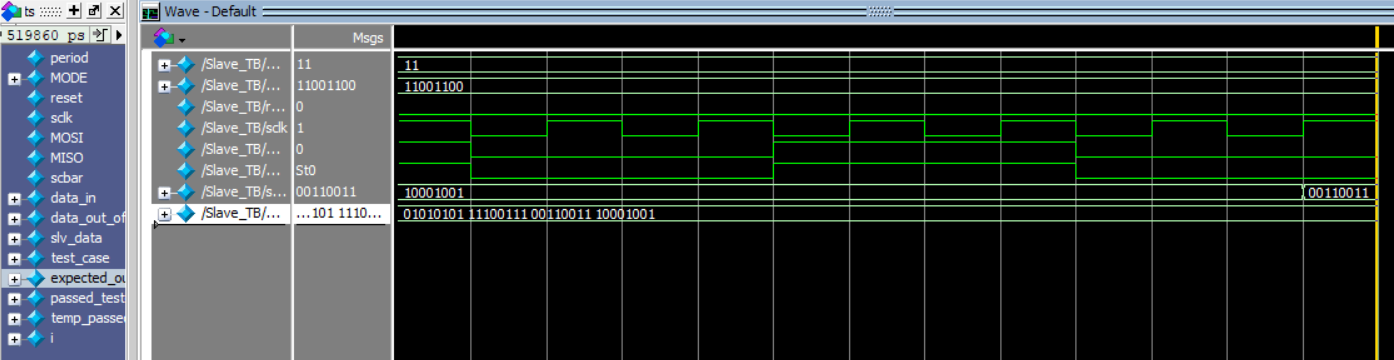
Figure testcase4\_output

Figure testcase4\_wave

## Integration

It has four test cases:

### Test case #1:

Mode to be tested: mode #1

Operation to be done: sending one byte to slave #1 and receiving another byte from it.

Aim of test case: testing reading and writing data.

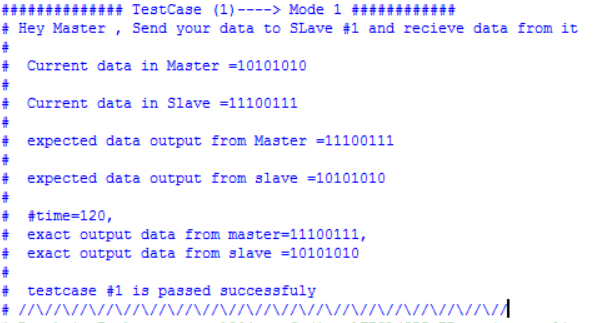


Figure SPI\_testcase1\_output

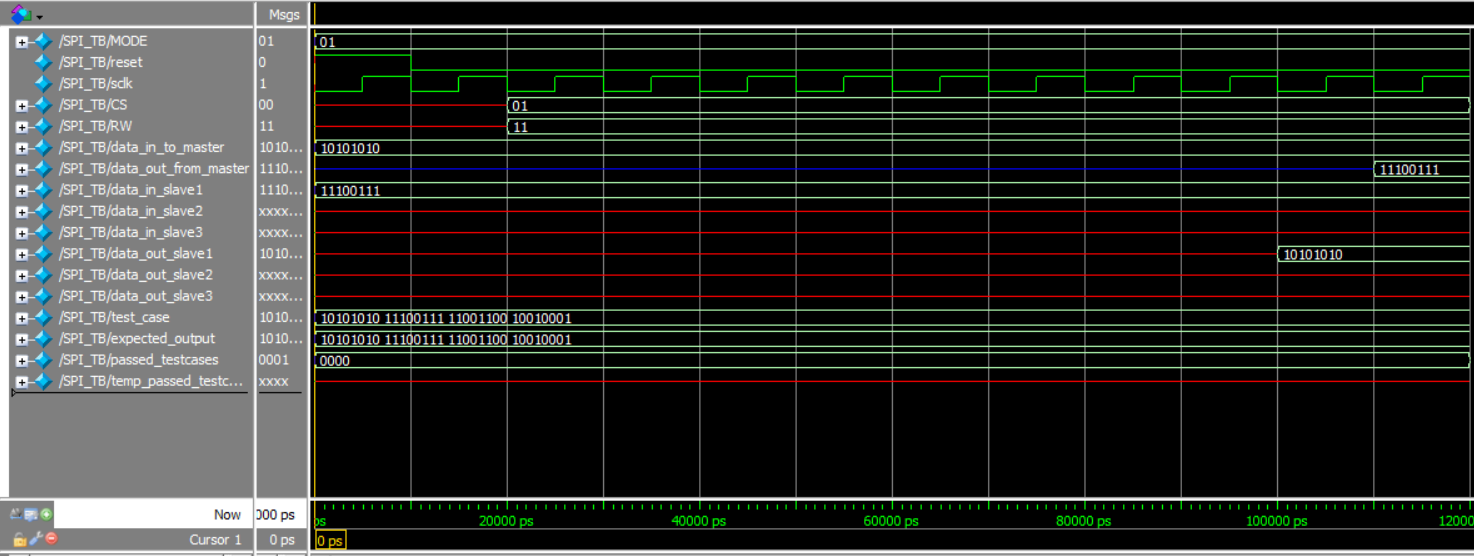


Figure SPI\_testcase1\_waveform

### Test case #2:

Mode to be tested: mode #0

Operation to be done: sending one byte to slave #2 and receiving another byte from it.

Aim of test case: testing reading and writing data.

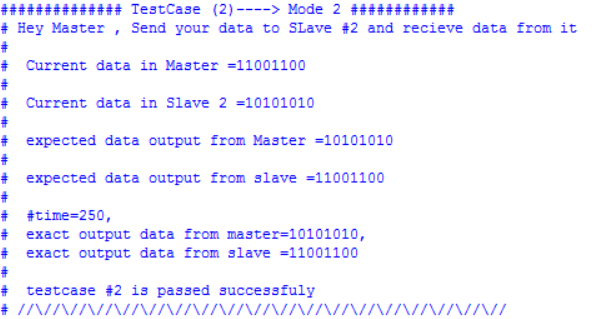


Figure SPI\_testcase2\_output

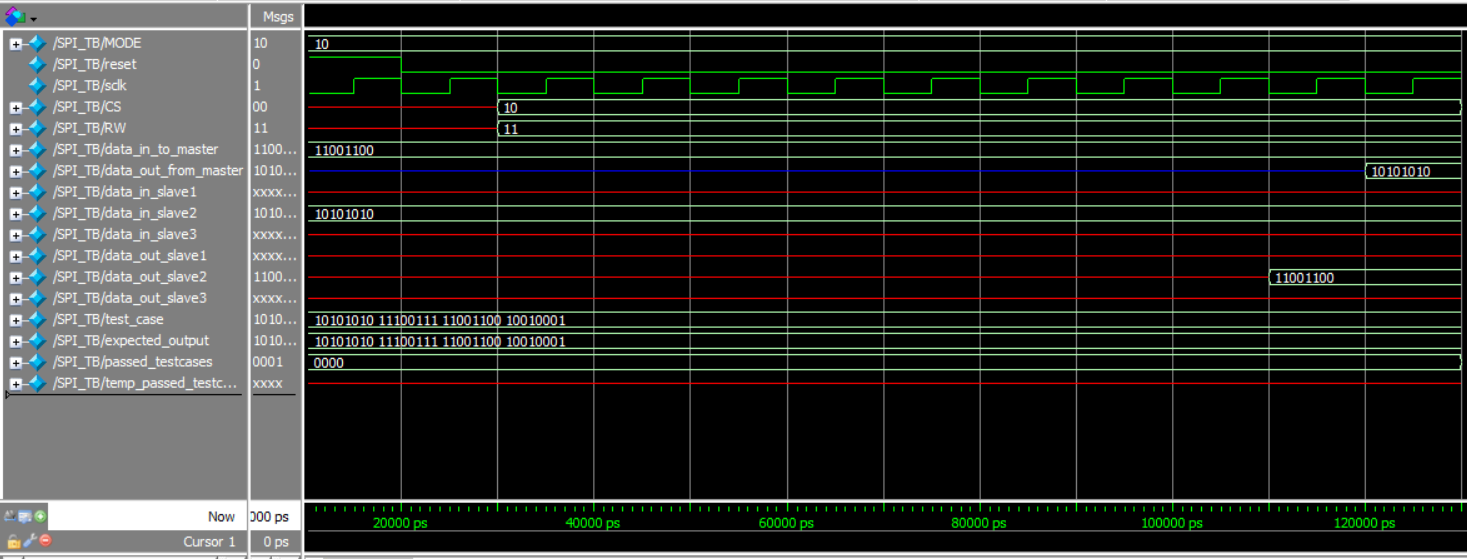


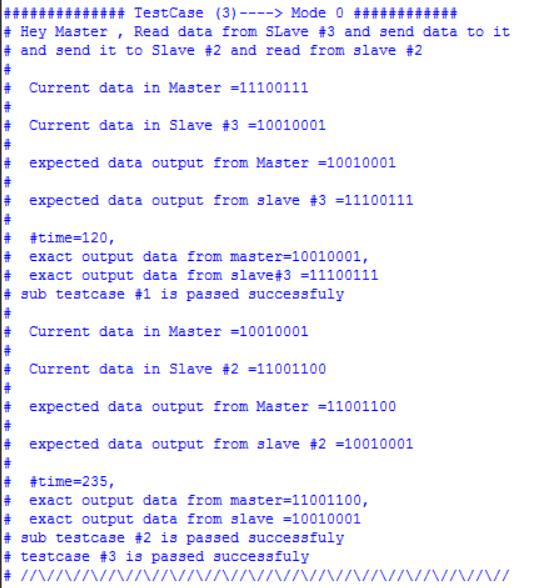
Figure SPI\_testcase2\_waveform

### Test case #3:

Mode to be tested: mode #2

Operation to be done: sending and reading from slave #3 and sending the readings to slave #2 and read another one.

Aim of test case: testing reading and writing data and switching slaves is done correctly.



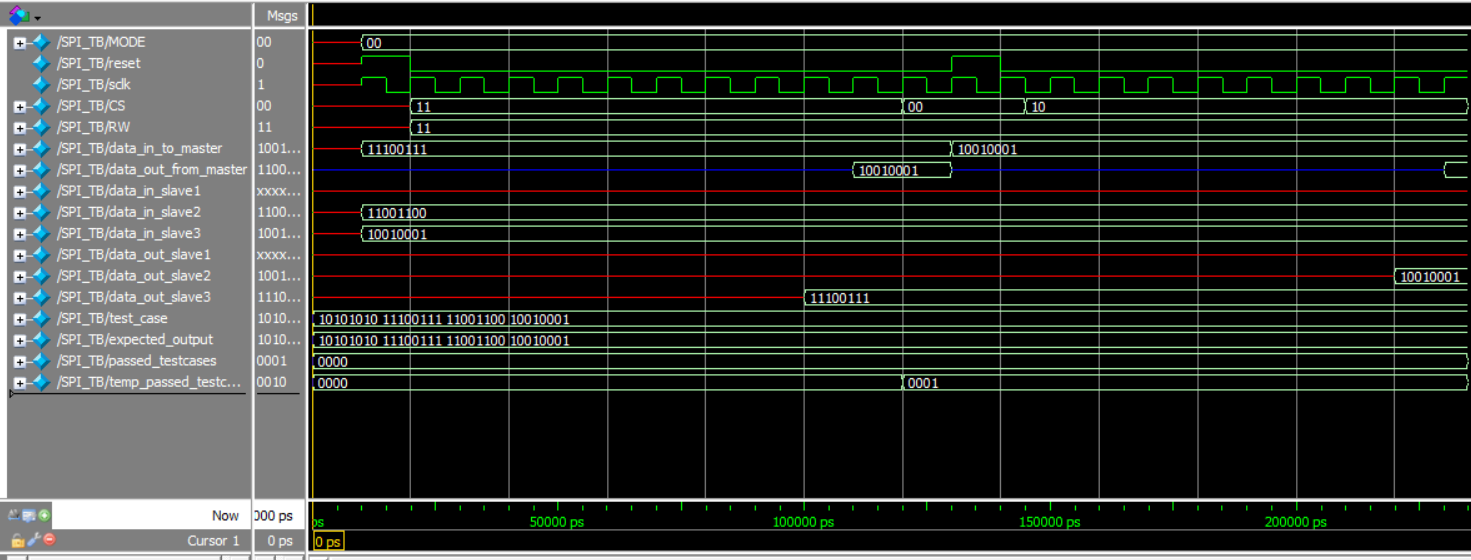
Figure SPI\_testcase3\_output

Figure SPI\_testcase3\_waveform

### Test case #4:

Mode to be tested: mode #3

Operation to be done: Reading data from Slave #1 and sending it to Slave #2 then Reading data from Slave#2 sending it to slave #3

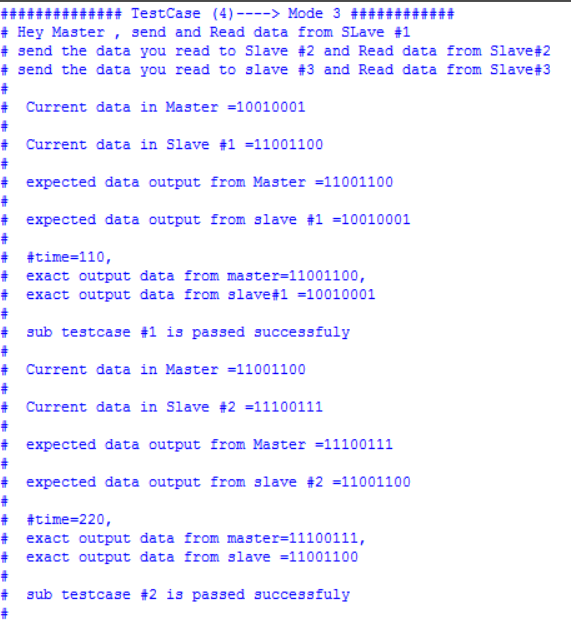
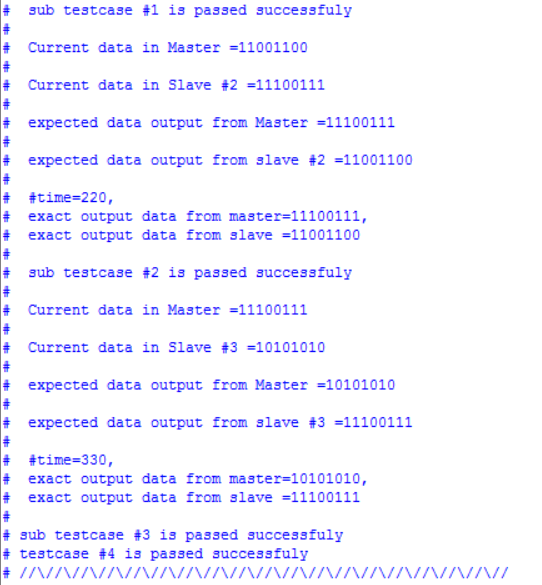
Aim of test case: testing reading and writing data and switching slaves is done correctly.

Figure 26 SPI\_testcase4\_waveform

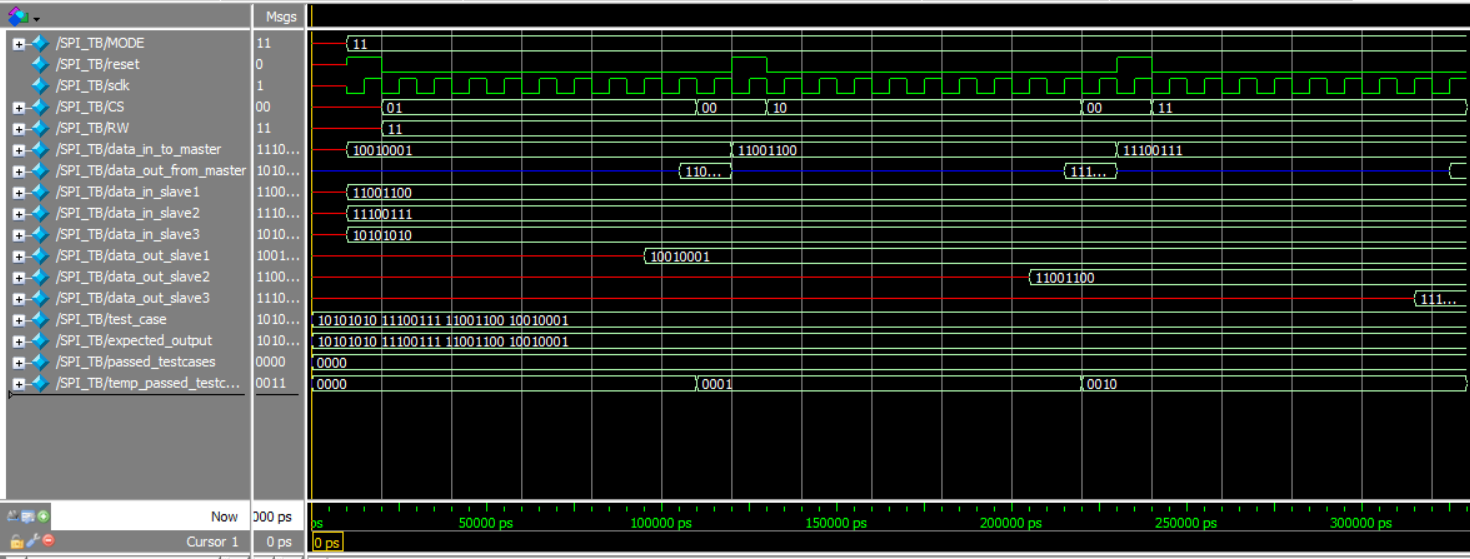


Figure SPI\_testcase4\_output

Figure 24 SPI\_testcase4\_output

### Test cases 1,2,3,4 followed by each other:

Figure 27 SPI\_testcases1,2,3,4\_waveform

|  |  |  |  |
| --- | --- | --- | --- |
| POC | SPI | I2C | UART |
| Short for | Serial peripheral interface | Inter integrated circuit | Universal asynchronous receive transmit |
| Max speed | 4 Mbps | 1 Mbps | 115.2 Kbps |
| Max distance | 0.1 m | 0.5 m | 15 m |
| Maximum possible number of devices to be connected to the bus | Theoretically unlimited | 127 | 2 (point to point) |
| Master | Single master protocol | Multi master protocol | NA |
| Wires | 3+n x CS | 2 | 2 |
| Synchronization | Synchronous | Synchronous | Synchronous |
|  |  |  |  |
| duplicity | Full -duplex | Half-duplex | Full-duplex |
| Examples | * Serial EEPROMS * Digital to Analog Converter (ADC) * Digital Signal Processor (DSP) | * Serial EEPROMS * Digital to Analog Converter (ADC) | * Bluetooth |