SPI (Serial Peripheral Interface)

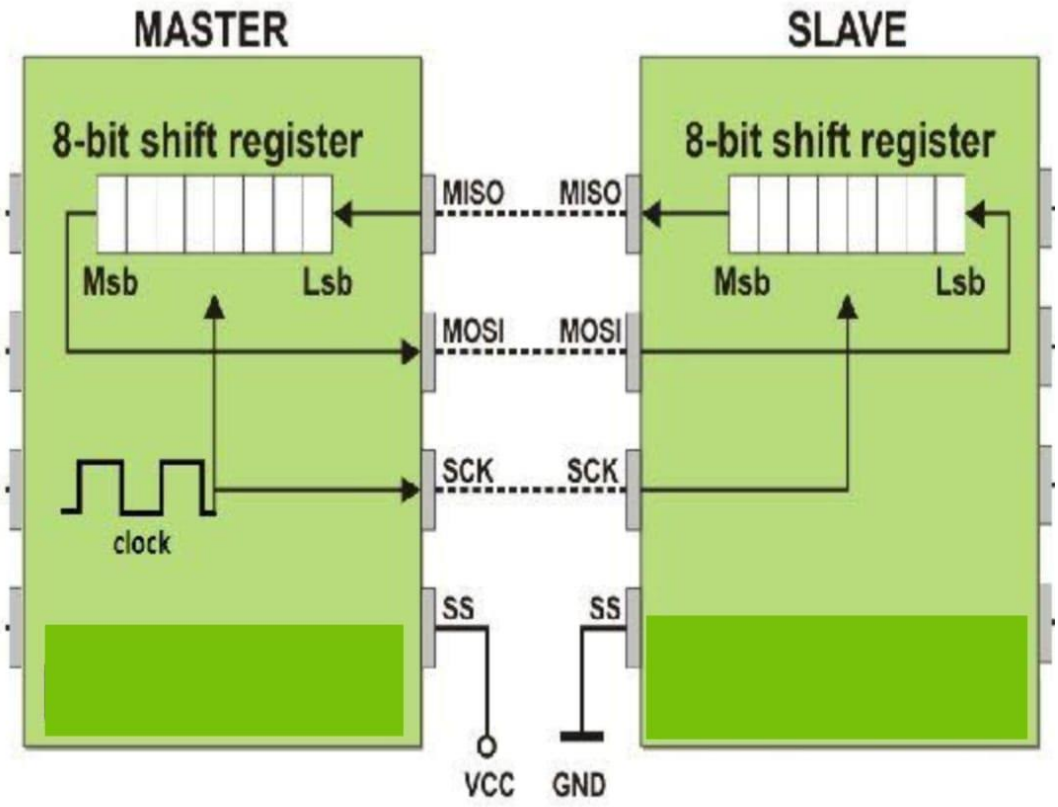
- **Brief**

Serial peripheral interface (SPI) is one of the most widely used interfaces between microcontroller and peripheral ICs such as sensors, ADCs, DACs, shift registers, SRAM, and others.

The data from the master or the slave is synchronized on the rising or falling clock edge. Both master and slave can transmit data at the same time.

This module work in SPI Mode 1, CPOL = 0, CPHA = 1: CLK idle state = low, data sampled on the falling edge and shifted on the rising edge.

- **Interface**



Master

Master is the controller of the SPI interface as it can choose which slave will receive master's data. When the master wants to send data to the slave. At first, it loads the data into its shift register then select its target slave, that selection done by chip select (CS) as master has (CS) lines for each slave.

```
23
24  always @(posedge start) begin
25          if(count >8 )count =0;
26          case(slaveSelect)
27                  0: CS = 3'b011;
28                  1: CS = 3'b101;
29                  2: CS = 3'b110;
30          endcase
31  end
32
```

For ex: if we want to send master's data to the first slave we set (CS1) (LOW) while others are (HIGH) and after (8) clock, master's data will be sent to that slave and so on with others.

At first always cycle if (there is a start signal) then assign (counter = 0).

The master is also responsible to set CS to 111 when the counter reach 8 so that no transmission will happen and all slaves will be disabled.

```
32
33  always @ (posedge clk) begin
34          if(count>=8) CS =3'b111;
35
36  end
37
```

In the negative edge:

We check that if (there is a selected slave) then shift (Master data received) from its first bit to (MOSI) that will be input to selected slave.

```
19
20  always @ (negedge clk)begin
21          if(CS) MOSI <= masterDataReceived[0];
22  end
23
```

In the positive edge:

We have three conditions after we check them, we make (Master Data Received to) equal to (Next).

- First condition: if (reset ==1) →→ assign (Next=0 & counter=0).
- Second condition: else if(counter==0) →→assign (Next= master Data To Send) & increment the counter.
- Third condition: else if (counter<8) →→ Shifting data & increment counter.

As we are focus on mode (1) that has CPOL=0 & CPHA=1

➔ The first clock signal rising can be used to prepare the data.
➔ The clock idle state is zero.
➔ The data on MISO and MOSI lines must be stable while the clock is low and can be changed when the clock is high.
➔ The data is captured on the clock's high-to-low propagated on low-to-high clock transition.

```
37
38  always @ (posedge clk) begin
39
40          if(reset) begin
41                  next =0;
42                  count =0;
43          end
44          else if(count ==0) begin
45                  next=masterDataToSend;
46                  count = count +1;
47          end
48          else if(count <=8)begin
49                  next = {MISO,masterDataReceived[7:1]};
50                  count = count +1;
51          end
52  masterDataReceived<=next;
53  end
54  endmodule
55
```

The Slave Module has 7 chips:

1. Inputs

i- Reset: will make all variables = 0 idle state.

ii- SCLK: The clock generated by the master for the transmission. The master uses the "clk" to generate this signal. Both the master and the slave can only use this signal for synchronizing the transmission.

iii- CS: The chip select signal used by the master to select a slave. If a slave is selected, the master should set its corresponding CS to 0 (active low).

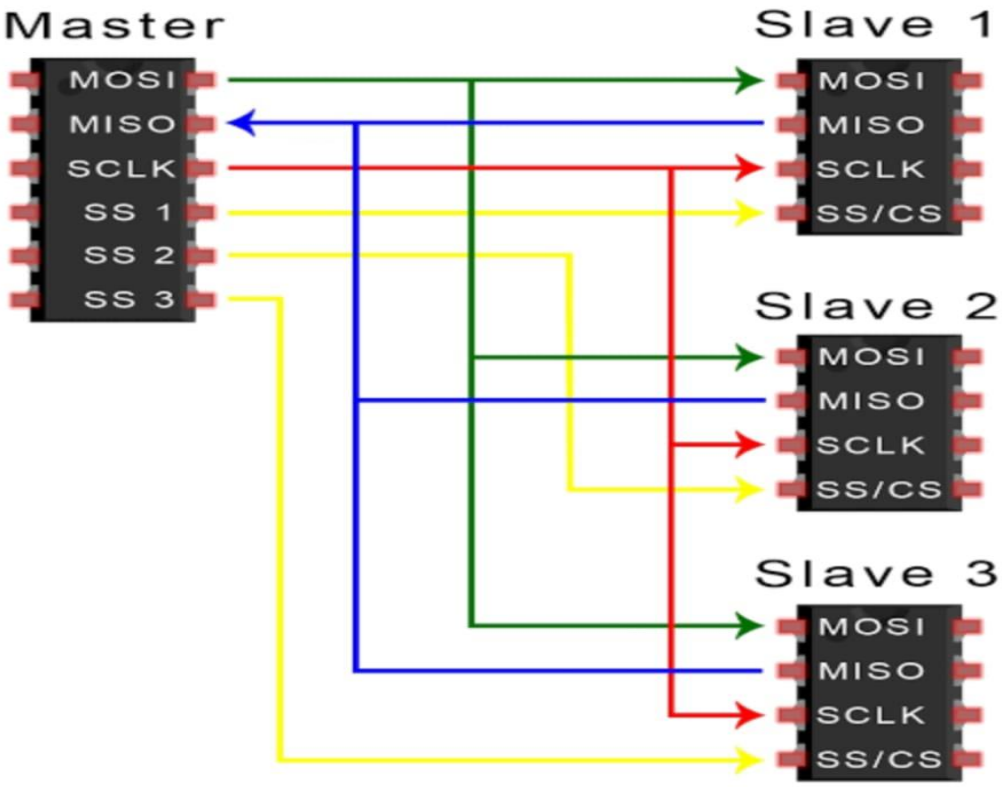
iv- SlaveDataToSend :8bits=1Byte refer to What data should the slave send to the master during the transmission, it will be read then transmitted bit by bit to Master.

v- MOSI: Master Out Slave In it refers to the data signal going from the master to the slave.

2. Outputs

i- MISO: Slave Out Master In it the data signal going from the slave to the master.

ii- SlaveDataReceived: 8bits=1Byte refer to What data did the slave receive from the master during the past transmission, it will be the joker variable it will read SlaveDataToSend then transmitted bit by bit to Master in the same time it received MOSI from master and push it until all 8bits of SlaveDataToSend transmitted and all 8bits of MasterDataToSend are received.



The Design Process:

I need Dummy variables

a- Count: this variable very important, it refers to number of shifting operation that happen it will start from 1 to 8 to transmit 1 Byte then if it reaches 8 that mean the process completed successfully.

b- NextSlaveDataToSend: this variable for storing the value of current SlaveDataReceived in single count step.

```
15
16  always @(CS) begin
17          count = 0;
18  end
19
```

If Slave select (CS) is changed (Disconnected or already Connected) it means that this Slave will be Refreshed with initial values 0.

```
19
20  always @ (posedge CS)begin
21          MISO <= 1'bz;
22          count = 'bz;
23  end
24
```

If The Slave is disabled and disconnected to the Master, then the count will be (Z) mean stop Transmit and disable count.

MISO is common input to Master from 3 Slaves so to avoid corruption of data on MISO wire so high impedance (Z) will solve this corruption.

```
24
25  always @(negedge SCLK)begin
26          if(!CS) MISO<=slaveDataReceived[0];
27  end
28
```

ON THE NEGATIVE EDGE (SCLK):

The slave will transmit (shifted out serially onto the MISO) the last left bit to the Master.

```
28
29  always @ (posedge SCLK) begin
30
31          if(reset) begin
32                  next_slaveDataToSend =0;
33                  count = 0;
34          end
35          else if(CS==0)begin
36                  if(count==0)  begin
37                          next_slaveDataToSend = slaveDataToSend;
38                          count = count +1;
39                  end
40                  else if(count<=8)begin
41                          next_slaveDataToSend = {MOSI,slaveDataReceived[7:1]};
42                          count = count +1;
43                  end
44
45          end
46          slaveDataReceived <= next_slaveDataToSend;
47
48
49  end
50
```

ON THE POSITIVE EDGE (SCLK):

- ➔ Fist condition: if (reset =1) so I will make SlaveDataReceived = 0 and Counter = 0.
- ➔ Second condition: if (Counter = 0) it means the beginning of transmission so, I will read SlaveDataToSend (the data that will be transmitted) then start transmission on counter from 1step to 8 steps of transmission.
- ➔ Otherwise: the slave will be in Sampling mode received the data on the (MOSI) and then push it in the last right bit and Shift SlaveDataReceived by 1 bit so the new SlaveDataReceived is shifted.

Test Benches

- First: the test bench of the Master

First, we declare the variables which we need in the test bench including:

- master data to send - master data received - MISO - MOSI

- slave select - reset - start - clock

and other needed variables.

In declaration for each variable, we give it the appropriate size and type according to what will be needed in the process of transformation of data.

After that we take one object from master module and give it all data needed

Then we start to initialize the variables we have: Master, master data received, master data to send, reset and other variables by giving them appropriate values to work by

And we declare and initialize a vector to use in transforming.

We code two for loops for the transformation process the first one is to make a testcase for the vector initialize above before.

The second for loop is to receive the data in each clock: MISO and MOSI and the finish the transferring of data.

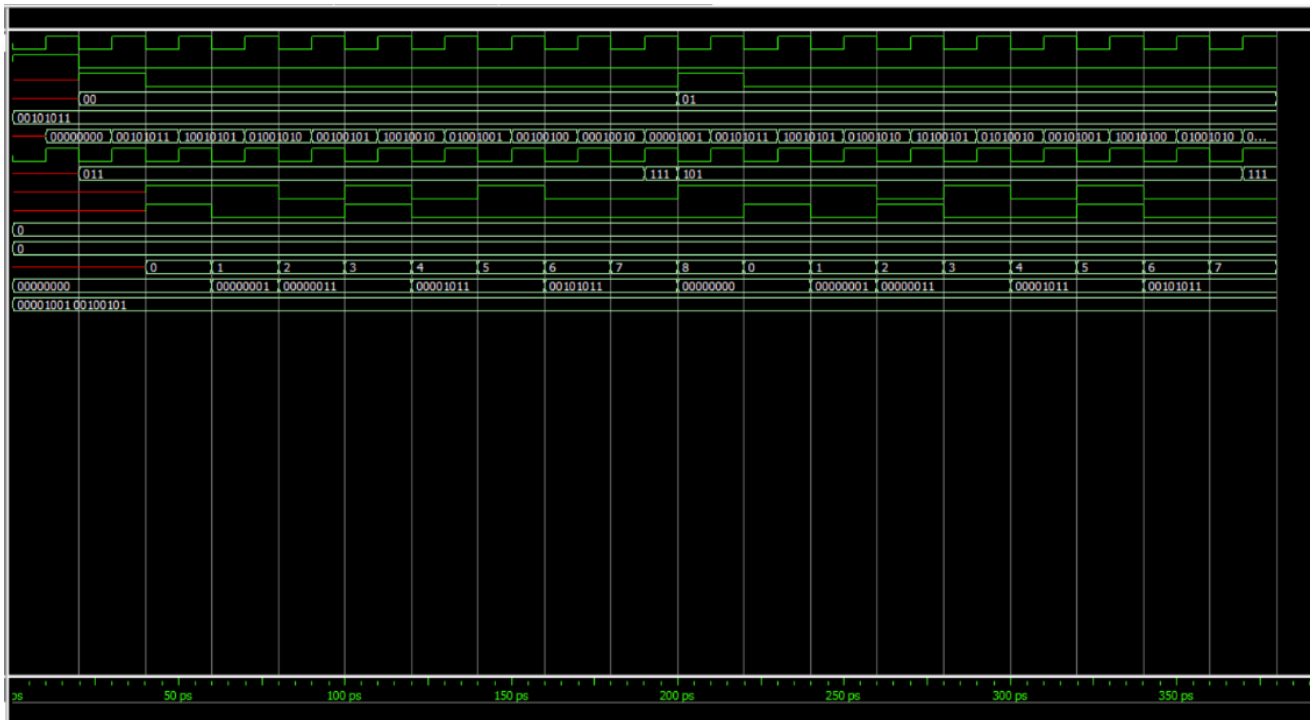
By ending of the for loop we put some if statements to see if the master and the slave has received the data successfully or any failures has happened and also, we count the number of failures and store it in the failure variable declared before

If the master received successfully the device will display (master received the data successfully from the slave) on the screen else if there are any failures will display statement to warn the user and the happens with the slave till all data transformed

After this we ask to see if there are already any failures in the process, if we found we display the number of them to the user, and if there is not, we display (SUCCESS: All testcases have been successful).

In the of the test bench, we assign the clock every 10 picosecond and then we arrive to the end of the test bench.

```
VSIM 18> run
# From Slave 0 to Master: Success
# From Master to Slave 0: Success
# From Slave 1 to Master: Success
# From Master to Slave 1: Success
# SUCCESS: All testcases have been successful
```



- Second: the test bench of the slave

In the beginning we declare all the attributes that we will use In the test bench including:

- slave data to send - slave data received - MISO - MOSI

-Master - reset - period - clock

and other needed variables.

For each variable in declaration, we give it the appropriate size according to what will be needed in the process of transformation of data.

After that we take two objects from slave module and give them all data needed

Then we start to initialize the variables we have Master, slave data received, slave data to send, reset and other variables by giving them appropriate values to work by

And we declare and initialize two vectors to use in transforming.

We code two for loops for the transformation process the first one is to make a testcase for the vector initialize above before.

The second for loop is to receive the data in each clock: MISO and MOSI and the finish the transferring of data.

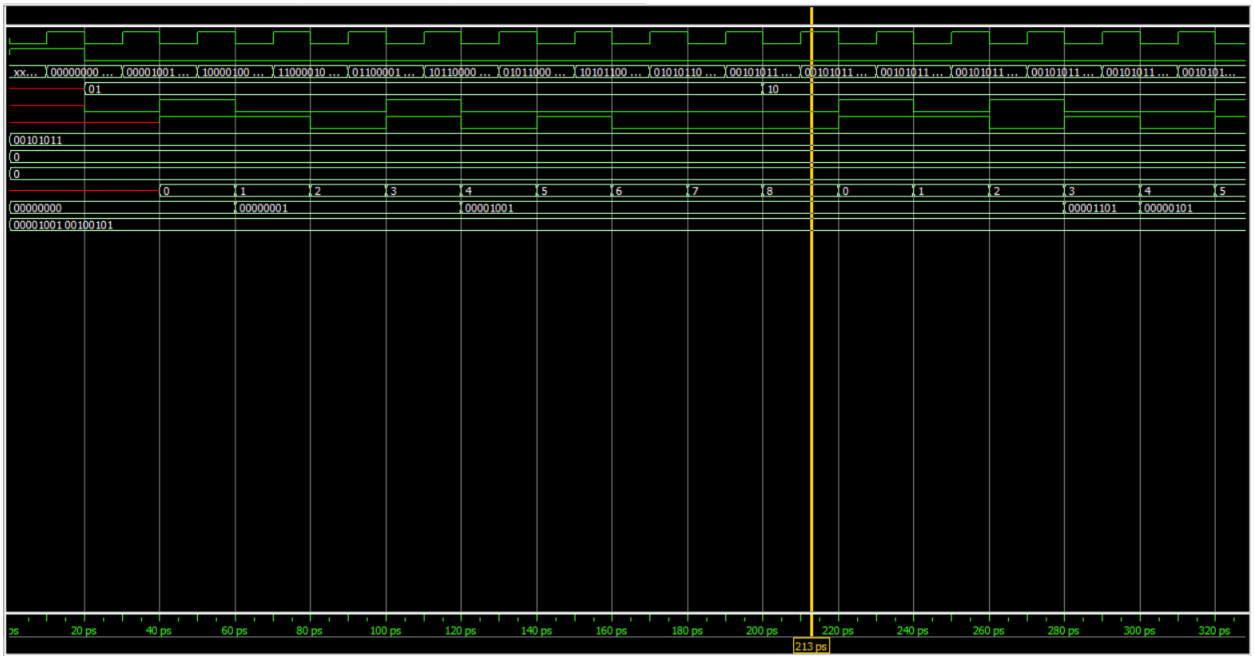
By ending of the for loop we put some if statements to see if the master and the slave has received the data successfully or any failures has happened and also we count the number of failures and store it in the failure variable declared before

If the master received successfully the device will display (master received the data successfully from the slave) on the screen else if there are any failures will display statement to warn the user and the happens with the slave till all data transformed

After this we ask to see if there are already any failures in the process, if we found already any failures in the process we display the number of them to the user, and if there is not, we display (SUCCESS: All testcases have been successful).

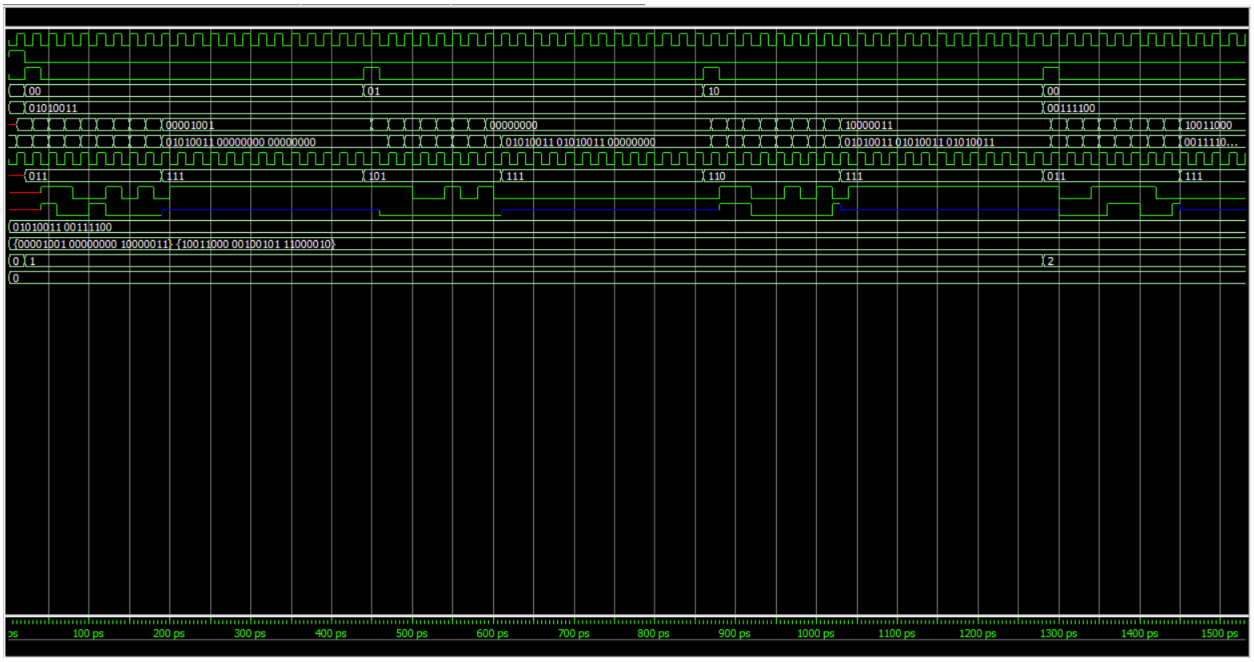
In the of the test bench, we assign the clock every 10 picosecond and then we arrive to the end of the test bench.

```
VSIM 7> run
# From to Master Slave          0: Success
# From Slave          0 to Master: Success
# From to Master Slave          1: Success
# From Slave          1 to Master: Success
# SUCCESS: All testcases have been successful
```



Simulation Result (of the given testbench development_tb**)**

```
VSIM 11> run
# Running test set           1
# From Slave 0 to Master: Success
# From Master to Slave 0: Success
# From Slave 1 to Master: Success
# From Master to Slave 1: Success
# From Slave 2 to Master: Success
# From Master to Slave 2: Success
# Running test set           2
# From Slave 0 to Master: Success
# From Master to Slave 0: Success
# From Slave 1 to Master: Success
# From Master to Slave 1: Success
# From Slave 2 to Master: Success
# From Master to Slave 2: Success
# SUCCESS: All          12 testcases have been successful
```



(The wave of first transmission)