Cairo University
Faculty of Engineering
Computer Engineering Dept.
Two-Semester - Spring 2021
CMP1030 - Logic Design

Team : 11

Name	Sec	BN	E-mail	ID
Aya Ahmed musad	1	14	aya.husien01@eng-St.cu.edu.eg	9202338
Doaa Ashraf Hamdy	1	22	doaa.ahmed01@eng-st.cu.edu.eg	9202519
Norhan Reda Abd El-wahed	2	31	Norhan.ahmed01@eng-st.cu.edu.eg	9203639
Hoda Gamal Hamouda	2	33	hoda.ismail00@eng-st.cu.edu.eg	9203673

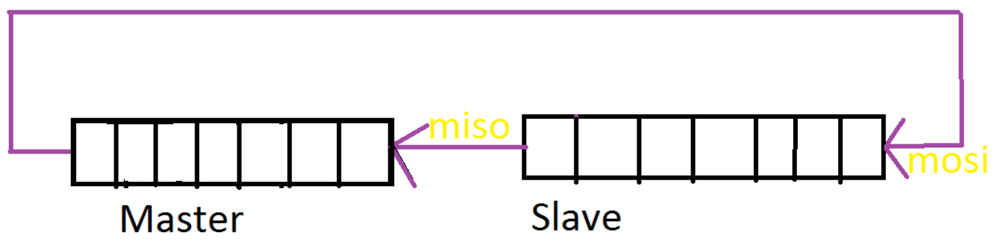
WORK DISTRIBUTION

Norhan Reda Abd El-wahed -> MASTER

Doaa Ashraf Hamdy -> SLAVE

Hoda Gamal Hamouda -> SLAVE TEST BENCH

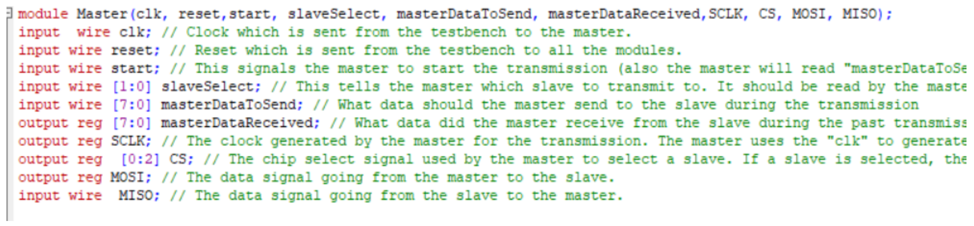
Aya Ahmed musad -> MASTER TEST BENCH



MASTER

Inputs : clk , reset , start , slave select , masterdata to send , MISO (master in slave out) .

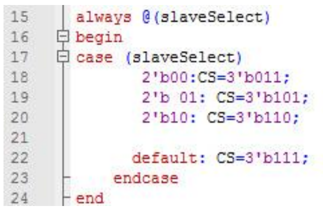
Outputs : sclk (serial clock) ,cs (chip select), MOSI(master out slave in), master data received.



```
module Master(clk, reset,start, slaveSelect, masterDataToSend, masterDataReceived,SCLK, CS, MOSI, MISO);
input  wire clk; // Clock which is sent from the testbench to the master.
input wire reset; // Reset which is sent from the testbench to all the modules.
input wire start; // This signals the master to start the transmission (also the master will read "masterDataToSe
input wire [1:0] slaveSelect; // This tells the master which slave to transmit to. It should be read by the maste
input wire [7:0] masterDataToSend; // What data should the master send to the slave during the transmission
output reg [7:0] masterDataReceived; // What data did the master receive from the slave during the past transmiss
output reg SCLK; // The clock generated by the master for the transmission. The master uses the "clk" to generate
output reg  [0:2] CS; // The chip select signal used by the master to select a slave. If a slave is selected, the
output reg MOSI; // The data signal going from the master to the slave.
input wire  MISO; // The data signal going from the slave to the master.
```

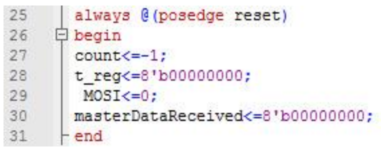
A temp variable -> t_reg[7:0] is used

to perform the shifting operation on it and an integer count to stop after counting 8 bits



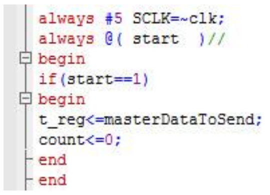
```
always @(slaveSelect)
begin
case (slaveSelect)
       2'b00:CS=3'b011;
       2'b 01: CS=3'b101;
       2'b10: CS=3'b110;

      default: CS=3'b111;
    endcase
end
```

This block of code ,selects which slave to communicate with corresponding to the value of the slaveSelect



```
always @(posedge reset)
begin
count<=-1;
t_reg<=8'b00000000;
 MOSI<=0;
masterDataReceived<=8'b00000000;
end
```

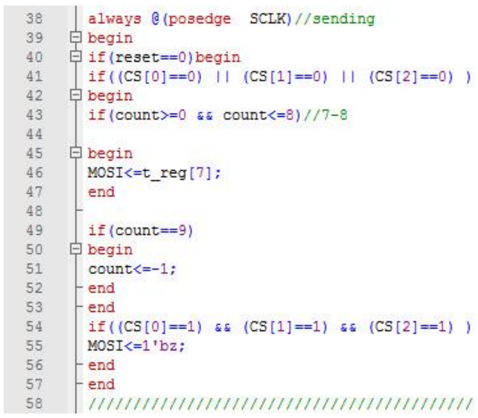
This block resets the master



```
always #5 SCLK=~clk;
always @( start  )//
begin
if(start==1)
begin
t_reg<=masterDataToSend;
count<=0;
end
end
```

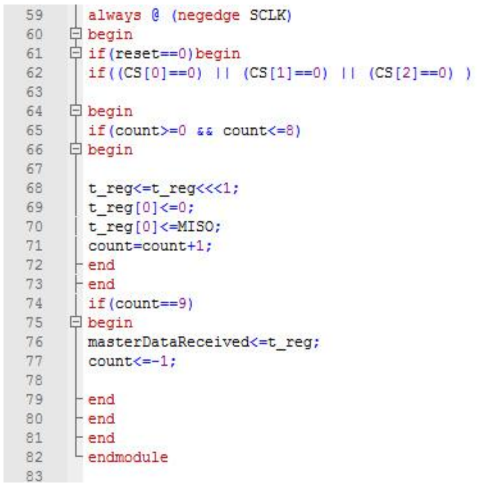
In the first line I generate the SCLK from the CLK

And then check the start to start communicate With the data to send



```
always @(posedge  SCLK)//sending
begin
if(reset==0)begin
if((CS[0]==0) || (CS[1]==0) || (CS[2]==0) )
begin
if(count>=0 && count<=8)//7-8

begin
MOSI<=t_reg[7];
end

if(count==9)
begin
count<=-1;
end
end
if((CS[0]==1) && (CS[1]==1) && (CS[2]==1) )
MOSI<=1'bz;
end
end
//////////////////////////////////////////
```

At the positive edge of the SCLK w assign data to MOSI if it is connected to a slave



```
always @ (negedge SCLK)
begin
if(reset==0)begin
if((CS[0]==0) || (CS[1]==0) || (CS[2]==0) )

begin
if(count>=0 && count<=8)
begin

t_reg<=t_reg<<<1;
t_reg[0]<=0;
t_reg[0]<=MISO;
count=count+1;
end
end
if(count==9)
begin
masterDataReceived<=t_reg;
count<=-1;

end
end
end
endmodule
```

At the negative edge of the SCLK I shift the value of t_reg one bit to the left with the value of the miso sent to the master if it is connected to a slave And then increment the count with one .

CODE

```
module Master(clk, reset,start, slaveSelect, masterDataToSend, masterDataReceived,SCLK, CS, MOSI, MISO);

input  wire clk; // Clock which is sent from the testbench to the master.

input wire reset; // Reset which is sent from the testbench to all the modules.

input wire start; // This signals the master to start the transmission (also the master will read "masterDataToSend" in order to send it to the slave).

input wire [1:0] slaveSelect; // This tells the master which slave to transmit to. It should be read by the master when "start" becomes high.

input wire [7:0] masterDataToSend; // What data should the master send to the slave during the transmission

output reg [7:0] masterDataReceived; // What data did the master receive from the slave during the past transmission

output reg 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.

output reg  [0:2] 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).

output reg MOSI; // The data signal going from the master to the slave.

input wire  MISO; // The data signal going from the slave to the master.


reg [7:0] t_reg;

integer count;
```

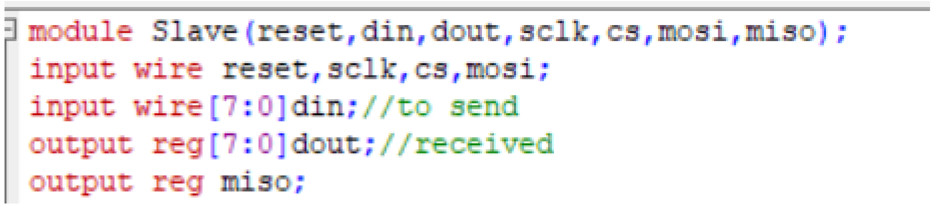
```
always @(slaveSelect)

begin

case (slaveSelect)

       2'b00:CS=3'b011;

       2'b 01: CS=3'b101;

       2'b10: CS=3'b110;


      default: CS=3'b111;

    endcase

end

always @(posedge reset)

begin

count<=-1;

t_reg<=8'b00000000;

 MOSI<=0;

masterDataReceived<=8'b00000000;

end

always #5 SCLK=~clk;

always @( posedge start )//

begin

t_reg<=masterDataToSend;

count<=0;

end

always @(posedge  SCLK)//sending

begin
```

```
if(reset==0)begin

if((CS[0]==0) || (CS[1]==0) || (CS[2]==0) )

begin

if(count>=0 && count<=8)//7-8


begin

MOSI<=t_reg[7];

end


if(count==9)

begin

count<=-1;

end

end

if((CS[0]==1) && (CS[1]==1) && (CS[2]==1) )

MOSI<=1'bz;

end

end

//////////////////////////////////////////

always @ (negedge SCLK)

begin

if(reset==0)begin

if((CS[0]==0) || (CS[1]==0) || (CS[2]==0) )


begin
```

```
if(count>=0 && count<=8)

begin


t_reg<=t_reg<<<1;

t_reg[0]<=0;

t_reg[0]<=MISO;

count=count+1;

end

end

if(count==9)

begin

masterDataReceived<=t_reg;

count<=-1;


end

end

end

endmodule
```

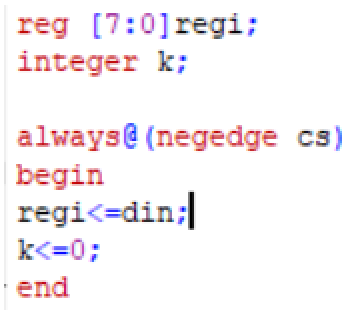
SLAVE

Inputs : reset , din(data to send), sclk (serial clock), cs (chip select), MOSI(master out slave in).
Outputs : MISO (master in slave out) , dout (data received)



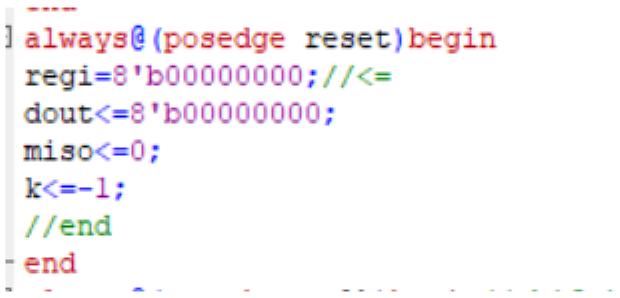
```
module Slave(reset,din,dout,sclk,cs,mosi,miso);
input wire reset,sclk,cs,mosi;
input wire[7:0]din;//to send
output reg[7:0]dout;//received
output reg miso;
```

The first always block works when cs is 0 because it is an active low signal , it makes the internal register regi , which was made for shifting , equal din and also k ,which is a counter , equal 0 .



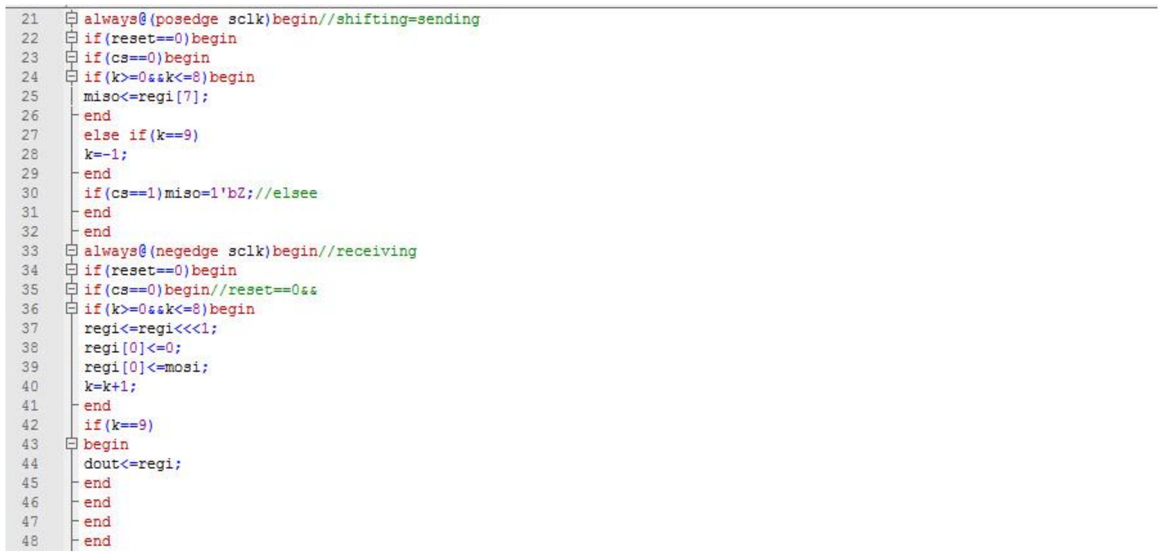
```
reg [7:0]regi;
integer k;

always@(negedge cs)
begin
regi<=din;
k<=0;
end
```

When reset =1 , it resets the circuit , and makes k = -1 because we will not start counting unless cs =0 .



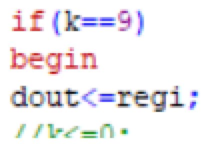
```
always@(posedge reset)begin
regi=8'b00000000;//<=
dout<=8'b00000000;
miso<=0;
k<=-1;
//end
end
```

At the positive edge of the clock , data is assigned to miso , at the negative edge we shift regi one bit for the bit to come from mosi and increments k with one .



```
always@(posedge sclk)begin//shifting=sending
if(reset==0)begin
if(cs==0)begin
if(k>=0&&k<=8)begin
miso<=regi[7];
end
else if(k==9)
k=-1;
end
if(cs==1)miso=1'bZ;//elsee
end
end
always@(negedge sclk)begin//receiving
if(reset==0)begin
if(cs==0)begin//reset==0&&
if(k>=0&&k<=8)begin
regi<=regi<<<1;
regi[0]<=0;
regi[0]<=mosi;
k=k+1;
end
if(k==9)
begin
dout<=regi;
end
end
end
end
```

If k =9 , this means we finished so we make dout = regi .



```
if(k==9)
begin
dout<=regi;
```

CODE

```
module Slave(reset,din,dout,sclk,cs,mosi,miso);

input wire reset,sclk,cs,mosi;

input wire[7:0]din;//to send

output reg[7:0]dout;//received

output reg miso;

reg [7:0]regi;

integer k;


always@(negedge cs)

begin

regi<=din;

k<=0;

end

always@(posedge reset)begin

regi=8'b00000000;

dout<=8'b00000000;

miso<=0;

k<=-1;

end

always@(posedge sclk)begin//shifting=sending

if(reset==0)begin

if(cs==0)begin

if(k>=0&&k<=8)begin

miso<=regi[7];
```

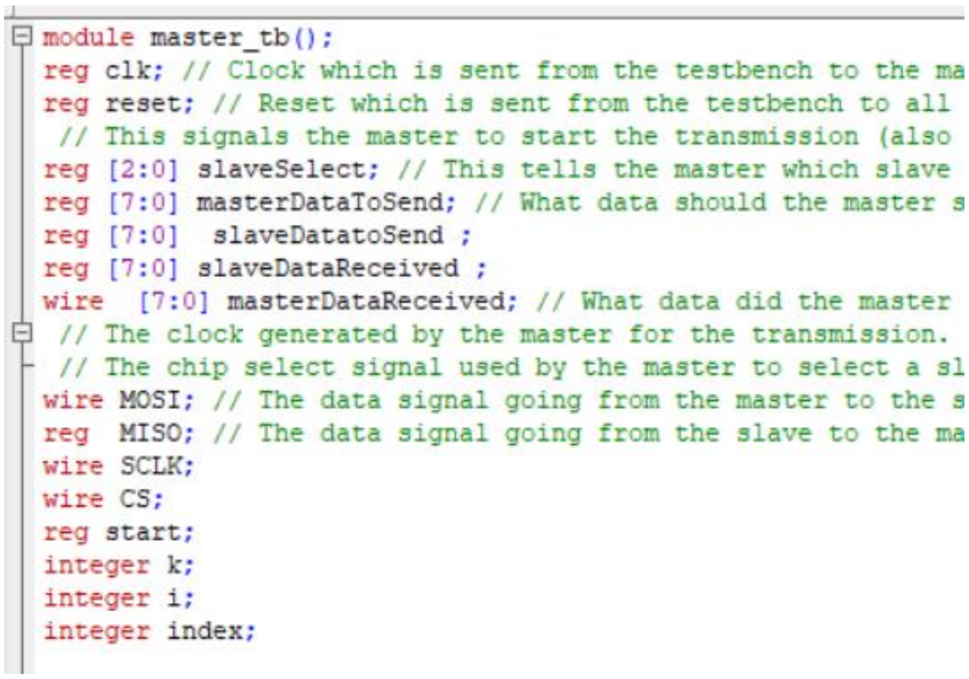
```
end

else if(k==9)

k=-1;

end

if(cs==1)miso=1'bZ;//elsee

end

end

always@(negedge sclk)begin//receiving

if(reset==0)begin

if(cs==0)begin//reset==0&&

if(k>=0&&k<=8)begin

regi<=regi<<<1;

regi[0]<=0;

regi[0]<=mosi;

k=k+1;

end

if(k==9)

begin

dout<=regi;

end

end

end

end

endmodule
```

MASTER_TEST BENCH

Inputs and outputs to instantiate a slave object : clk ,SCLK, reset , CS , masterDataTosend , masterDataReceived , MOSI , MISO , slaveSelect , start .

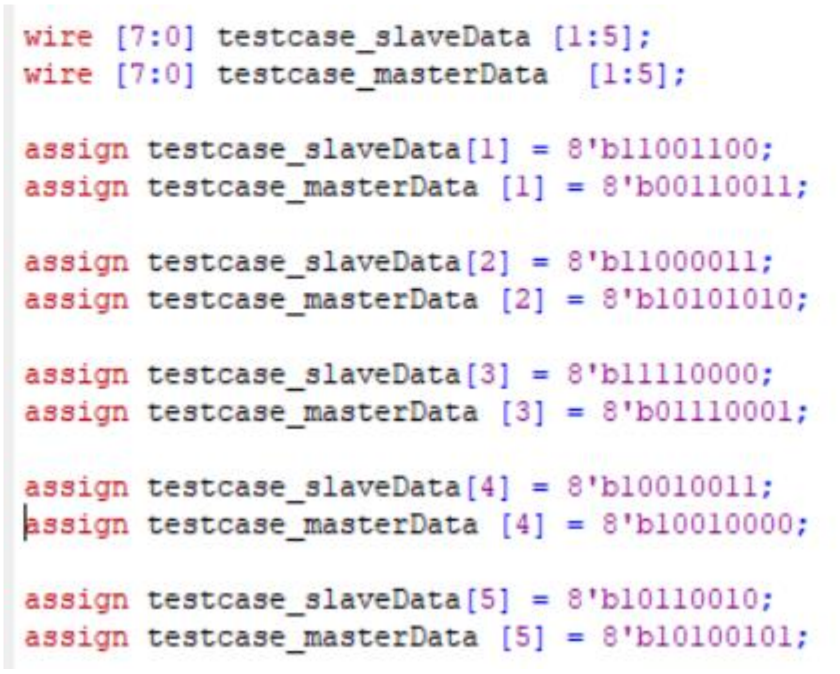
Counters : k , I , index .

Also we used two registers(slaveDataTosend , slaveDataRecieved) instead of the slave to check if the master is working correctly alone .



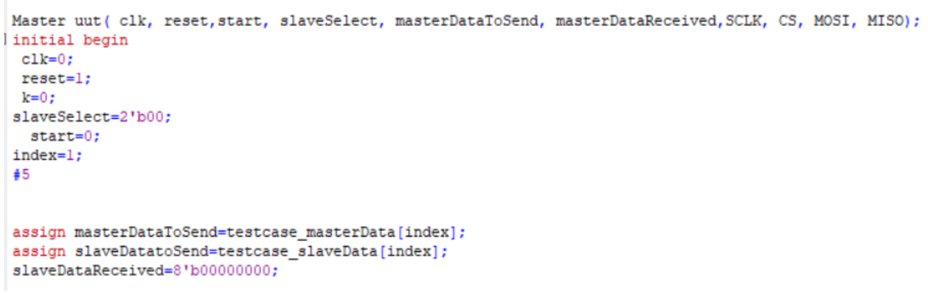
```
module master_tb();
reg clk; // Clock which is sent from the testbench to the ma
reg reset; // Reset which is sent from the testbench to all
 // This signals the master to start the transmission (also
reg [2:0] slaveSelect; // This tells the master which slave
reg [7:0] masterDataToSend; // What data should the master s
reg [7:0]  slaveDatatoSend ;
reg [7:0] slaveDataReceived ;
wire  [7:0] masterDataReceived; // What data did the master
 // The clock generated by the master for the transmission.
 // The chip select signal used by the master to select a sl
wire MOSI; // The data signal going from the master to the s
reg  MISO; // The data signal going from the slave to the ma
wire SCLK;
wire CS;
reg start;
integer k;
integer i;
integer index;
```

We then instantiate 2D arrays for our test cases and give them their values .



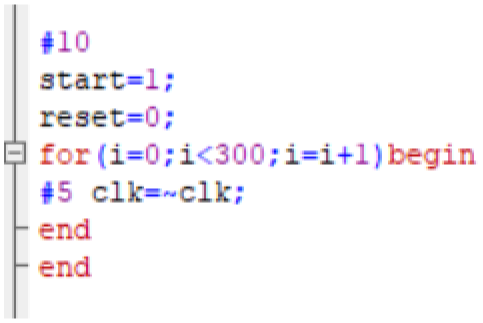
```
wire [7:0] testcase_slaveData [1:5];
wire [7:0] testcase_masterData  [1:5];

assign testcase_slaveData[1] = 8'b11001100;
assign testcase_masterData [1] = 8'b00110011;

assign testcase_slaveData[2] = 8'b11000011;
assign testcase_masterData [2] = 8'b10101010;

assign testcase_slaveData[3] = 8'b11110000;
assign testcase_masterData [3] = 8'b01110001;

assign testcase_slaveData[4] = 8'b10010011;
assign testcase_masterData [4] = 8'b10010000;

assign testcase_slaveData[5] = 8'b10110010;
assign testcase_masterData [5] = 8'b10100101;
```

Then we instantiate our master object and initially make clk =0 & reset =1 , slaveSelect =1 , index =1 (first test case) because simulation has not started yet. Wait for 5 delay and assign the slave & master test cases to masterDataTosend (as if there is a master) , and slaveDataTosend & initialy make slaveDataRecieved =0 .



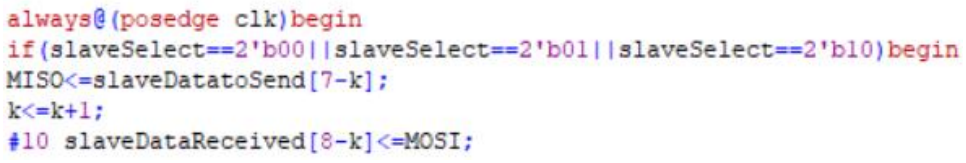
```
Master uut( clk, reset,start, slaveSelect, masterDataToSend, masterDataReceived,SCLK, CS, MOSI, MISO);
initial begin
 clk=0;
 reset=1;
 k=0;
slaveSelect=2'b00;
  start=0;
index=1;
#5


assign masterDataToSend=testcase_masterData[index];
assign slaveDatatoSend=testcase_slaveData[index];
slaveDataReceived=8'b00000000;
```

Wait for 10 delay and reset the circuit , then start a for loop for the clock .



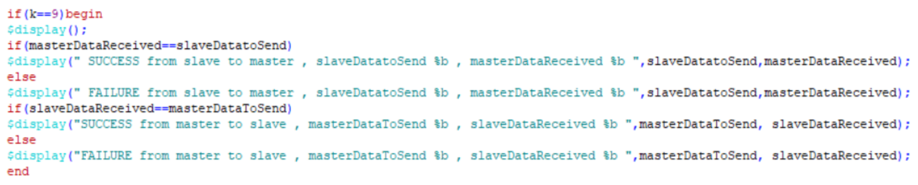
```
#10
start=1;
reset=0;
for(i=0;i<300;i=i+1)begin
#5 clk=~clk;
end
end
```

With every positive edge for the clock , we firstly check if slaveSelect =00||01||10 (as if there is a slave) . And start giving the most significant bit from slaveDataTosend to MISO , increment k , and wait for delay 10 to ensure that MOSI has his new value now and then gives it to the most significant bit of slaveDataRecieved .



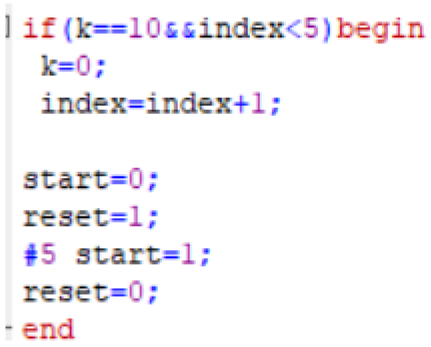
```
always@(posedge clk)begin
if(slaveSelect==2'b00||slaveSelect==2'b01||slaveSelect==2'b10)begin
MISO<=slaveDatatoSend[7-k];
k<=k+1;
#10 slaveDataReceived[8-k]<=MOSI;
```

If k=9 , this means we finished our 8 bits , we then check if masterDataReceived = slaveDataTosend , if yes , the means the transimition from slave to master was correct and we print success and our data aswell . We also check if slaveDataReceived = masterDataTosend , if yes , the means the transimition from slave to master was correct and we print success and our data aswell .



```
if(k==9)begin
$display();
if(masterDataReceived==slaveDatatoSend)
$display(" SUCCESS from slave to master , slaveDatatoSend %b , masterDataReceived %b ",slaveDatatoSend,masterDataReceived);
else
$display(" FAILURE from slave to master , slaveDatatoSend %b , masterDataReceived %b ",slaveDatatoSend,masterDataReceived);
if(slaveDataReceived==masterDataToSend)
$display("SUCCESS from master to slave , masterDataToSend %b , slaveDataReceived %b ",masterDataToSend, slaveDataReceived);
else
$display("FAILURE from master to slave , masterDataToSend %b , slaveDataReceived %b ",masterDataToSend, slaveDataReceived);
end
```

If k=10 , then we are sure that the first testcase has been finished , we also check the index because we have only 5 testcases , if both conditions are true we reset k to 0 to start again and increment inex to try the next test case , we also reset start to 0 and reset to 1 and wait for 5 delay and set the circuit by making reset =0 and and start =1.



```
if(k==10&&index<5)begin
  k=0;
  index=index+1;

 start=0;
 reset=1;
 #5 start=1;
 reset=0;
end
```

CODE

module master_tb();

reg clk; // Clock which is sent from the testbench to the master.

reg reset; // Reset which is sent from the testbench to all the modules.

// This signals the master to start the transmission (also the master will read "masterDataToSend" in order to send it to the slave).

reg [2:0] slaveSelect; // This tells the master which slave to transmit to. It should be read by the master when "start" becomes high.

reg [7:0] masterDataToSend; // What data should the master send to the slave during the transmission

reg [7:0] slaveDatatoSend ;

reg [7:0] slaveDataReceived ;

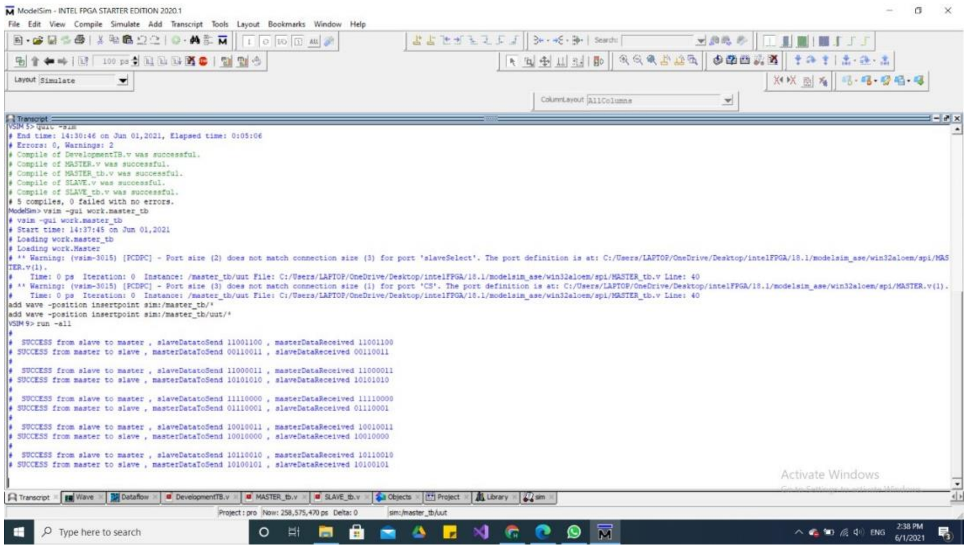
wire [7:0] masterDataReceived; // What data did the master receive from the slave during the past transmission

```
 // 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 synchro-
nizing the transmission.

 // 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).

wire MOSI; // The data signal going from the master to the slave.

reg  MISO; // The data signal going from the slave to the master.

wire SCLK;

wire CS;

reg start;

integer k;

integer i;

integer index;


wire [7:0] testcase_slaveData [1:5];

wire [7:0] testcase_masterData  [1:5];


assign testcase_slaveData[1] = 8'b11001100;

assign testcase_masterData [1] = 8'b00110011;


assign testcase_slaveData[2] = 8'b11000011;

assign testcase_masterData [2] = 8'b10101010;


assign testcase_slaveData[3] = 8'b11110000;

assign testcase_masterData [3] = 8'b01110001;


assign testcase_slaveData[4] = 8'b10010011;
assign testcase_masterData [4] = 8'b10010000;


assign testcase_slaveData[5] = 8'b10110010;

assign testcase_masterData [5] = 8'b10100101;




Master uut( clk, reset,start, slaveSelect, masterDataToSend, masterDataRe-
ceived,SCLK, CS, MOSI, MISO);

initial begin

 clk=0;

 reset=1;

 k=0;

slaveSelect=2'b00;

  start=0;

index=1;

#5




assign masterDataToSend=testcase_masterData[index];

assign slaveDatatoSend=testcase_slaveData[index];

slaveDataReceived=8'b00000000;


#10

start=1;

reset=0;

for(i=0;i<300;i=i+1)begin

#5 clk=~clk;
```

```
end

end


always@(posedge clk)begin

if(slaveSelect==2'b00||slaveSelect==2'b01||slaveSelect==2'b10)begin

MISO<=slaveDatatoSend[7-k];

k<=k+1;

#10 slaveDataReceived[8-k]<=MOSI;

if(k==9)begin

$display();

if(masterDataReceived==slaveDatatoSend)

$display(" SUCCESS from slave to master , slaveDatatoSend %b , masterDataReceived
%b ",slaveDatatoSend,masterDataReceived);

else

$display(" FAILURE from slave to master , slaveDatatoSend %b , masterDataReceived
%b ",slaveDatatoSend,masterDataReceived);

if(slaveDataReceived==masterDataToSend)

$display("SUCCESS from master to slave , masterDataToSend %b , slaveDataReceived
%b ",masterDataToSend, slaveDataReceived);

else

$display("FAILURE from master to slave , masterDataToSend %b , slaveDataReceived
%b ",masterDataToSend, slaveDataReceived);

end

if(k==10&&index<5)begin

 k=0;

 index=index+1;


start=0;
```

```
reset=1;

#5 start=1;

reset=0;

end

end



end

endmodule
```

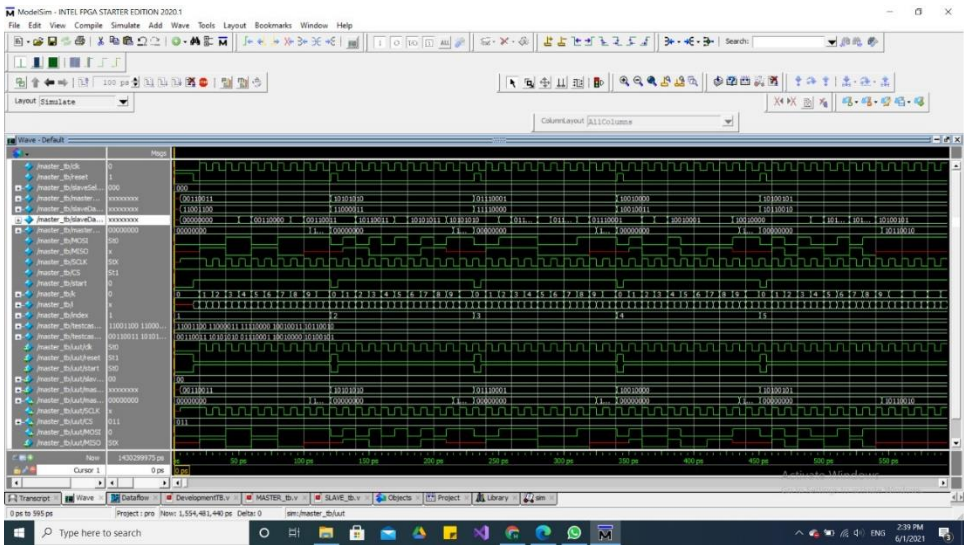
OUTPUT

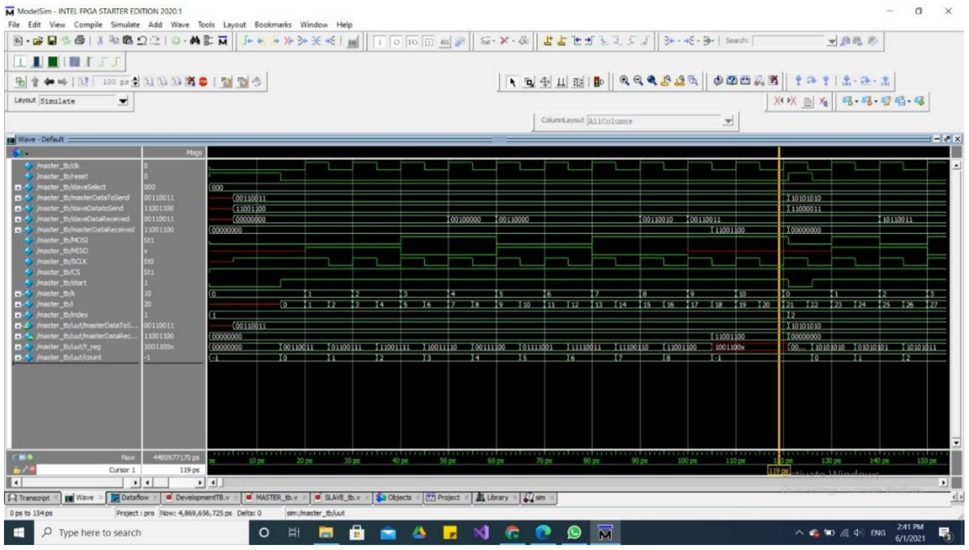


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
Layout Simulate
ColumnLayout AllColumns
Transcript

```
VSIM 5> quit -sim
# End time: 14:30:46 on Jun 01,2021, Elapsed time: 0:05:06
# Errors: 0, Warnings: 2
# Compile of DevelopmentTB.v was successful.
# Compile of MASTER.v was successful.
# Compile of MASTER_tb.v was successful.
# Compile of SLAVE.v was successful.
# Compile of SLAVE_tb.v was successful.
# 5 compiles, 0 failed with no errors.
ModelSim> vsim -gui work.master_tb
# vsim -gui work.master_tb
# Start time: 14:37:45 on Jun 01,2021
# Loading work.master_tb
# Loading work.Master
# ** Warning: (vsim-3015) [PCDPC] - Port size (2) does not match connection size (3) for port 'slaveSelect'. The port definition is at: C:/Users/LAPTOP/OneDrive/Desktop/intelFPGA/18.1/modelsim_ase/win32aloem/spi/MAS
TER.v(1).
#    Time: 0 ps  Iteration: 0  Instance: /master_tb/uut File: C:/Users/LAPTOP/OneDrive/Desktop/intelFPGA/18.1/modelsim_ase/win32aloem/spi/MASTER_tb.v Line: 40
# ** Warning: (vsim-3015) [PCDPC] - Port size (3) does not match connection size (1) for port 'CS'. The port definition is at: C:/Users/LAPTOP/OneDrive/Desktop/intelFPGA/18.1/modelsim_ase/win32aloem/spi/MASTER.v(1).
#    Time: 0 ps  Iteration: 0  Instance: /master_tb/uut File: C:/Users/LAPTOP/OneDrive/Desktop/intelFPGA/18.1/modelsim_ase/win32aloem/spi/MASTER_tb.v Line: 40
add wave -position insertpoint sim:/master_tb/*
add wave -position insertpoint sim:/master_tb/uut/*
VSIM 9> run -all
#
#  SUCCESS from slave to master , slaveDatatoSend 11001100 , masterDataReceived 11001100
# SUCCESS from master to slave , masterDataToSend 00110011 , slaveDataReceived 00110011
#
#  SUCCESS from slave to master , slaveDatatoSend 11000011 , masterDataReceived 11000011
# SUCCESS from master to slave , masterDataToSend 10101010 , slaveDataReceived 10101010
#
#  SUCCESS from slave to master , slaveDatatoSend 11110000 , masterDataReceived 11110000
# SUCCESS from master to slave , masterDataToSend 01110001 , slaveDataReceived 01110001
#
#  SUCCESS from slave to master , slaveDatatoSend 10010011 , masterDataReceived 10010011
# SUCCESS from master to slave , masterDataToSend 10010000 , slaveDataReceived 10010000
#
#  SUCCESS from slave to master , slaveDatatoSend 10110010 , masterDataReceived 10110010
# SUCCESS from master to slave , masterDataToSend 10100101 , slaveDataReceived 10100101
```

WAVE



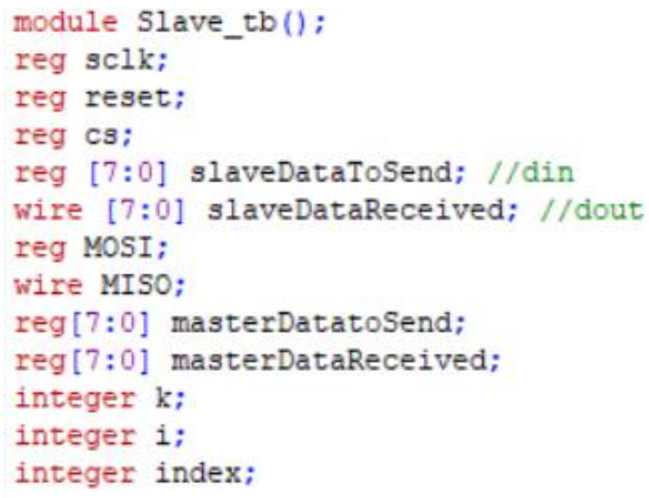


SLAVE_TEST BENCH

Inputs and outputs to instantiate a slave object : sclk , reset , CS , slaveDataTosend , slaveDataReceived , MOSI , MISO .

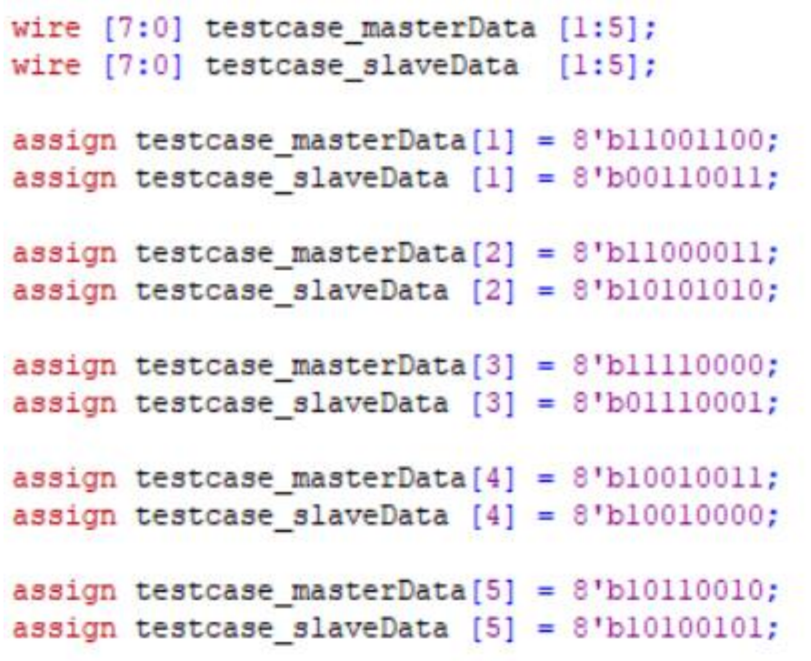
Counters : k , l , index .

Also we used two registers(masterDataTosend , masterDataRecieved) instead of the master to check if the slave is working correctly alone .



```
module Slave_tb();
reg sclk;
reg reset;
reg cs;
reg [7:0] slaveDataToSend; //din
wire [7:0] slaveDataReceived; //dout
reg MOSI;
wire MISO;
reg[7:0] masterDatatoSend;
reg[7:0] masterDataReceived;
integer k;
integer i;
integer index;
```

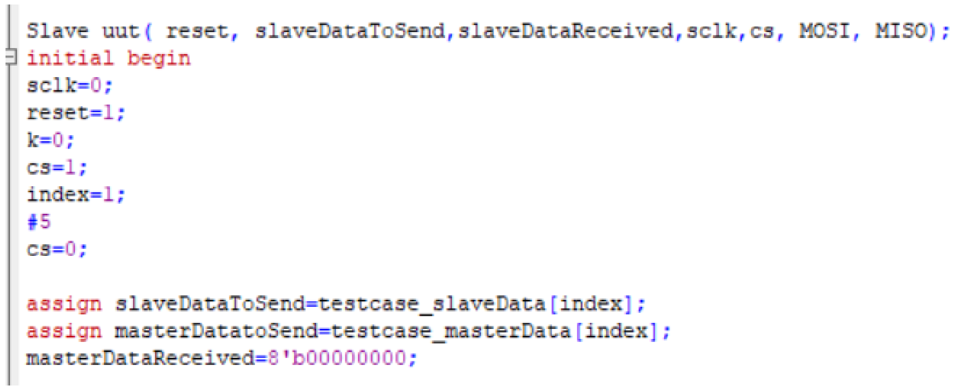
We then instantiate 2D arrays for our test cases and give them their values .



```
wire [7:0] testcase_masterData [1:5];
wire [7:0] testcase_slaveData  [1:5];

assign testcase_masterData[1] = 8'b11001100;
assign testcase_slaveData [1] = 8'b00110011;

assign testcase_masterData[2] = 8'b11000011;
assign testcase_slaveData [2] = 8'b10101010;

assign testcase_masterData[3] = 8'b11110000;
assign testcase_slaveData [3] = 8'b01110001;

assign testcase_masterData[4] = 8'b10010011;
assign testcase_slaveData [4] = 8'b10010000;

assign testcase_masterData[5] = 8'b10110010;
assign testcase_slaveData [5] = 8'b10100101;
```

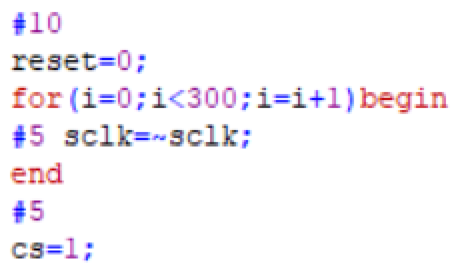
Then we instantiate our slave object and initially make sclk =0 & reset =1 , cs =1 , index =1 (first test case) because simulation has not started yet .

Wait for 5 delay and assign the slave & master test cases to masterDataTosend (as if there is a master) , and slaveDataTosend & initialy make masterDataRecieved =0 .



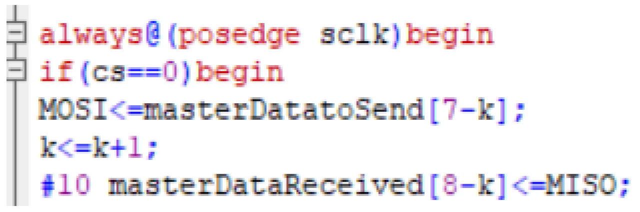
```
Slave uut( reset, slaveDataToSend,slaveDataReceived,sclk,cs, MOSI, MISO);
initial begin
sclk=0;
reset=1;
k=0;
cs=1;
index=1;
#5
cs=0;

assign slaveDataToSend=testcase_slaveData[index];
assign masterDatatoSend=testcase_masterData[index];
masterDataReceived=8'b00000000;
```

Wait for 10 delay and reset the circuit , then start a for loop for the serial clock and also make cs =1 (we have not started yet).



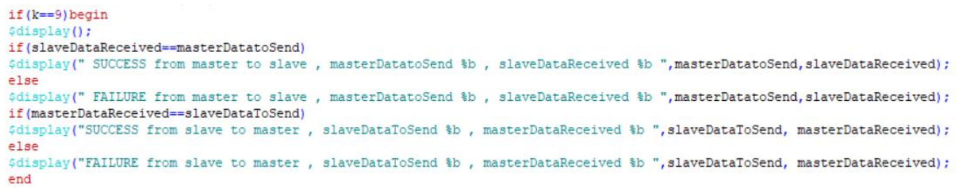
```
#10
reset=0;
for(i=0;i<300;i=i+1)begin
#5 sclk=~sclk;
end
#5
cs=1;
```

With every positive edge for the serial clock , we firstly check if cs =0 (active low signal) . And start giving the most significant bit from masterDataTosend to MOSI , increment k , and wait for delay 10 to ensure that MISO has his new value now and then gives it to the most significant bit of masterDataRecieved .



```
always@(posedge sclk)begin
if(cs==0)begin
MOSI<=masterDatatoSend[7-k];
k<=k+1;
#10 masterDataReceived[8-k]<=MISO;
```

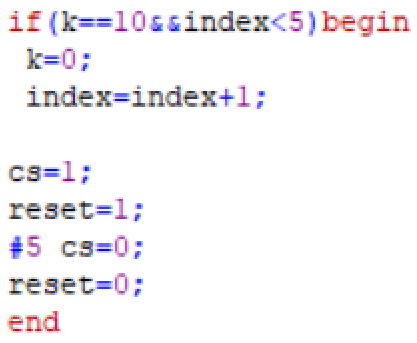
If k=9 , this means we finished our 8 bits , we then check if slaveDataReceived = masterDataTosend , if yes , the means the transimition from master to slave was correct and we print success and our data aswell . We also check if masterDataReceived = slaveDataTosend , if yes , the means the transimition from slave to master was correct and we print success and our data aswell .



```
if(k==9)begin
$display();
if(slaveDataReceived==masterDatatoSend)
$display(" SUCCESS from master to slave , masterDatatoSend %b , slaveDataReceived %b ",masterDatatoSend,slaveDataReceived);
else
$display(" FAILURE from master to slave , masterDatatoSend %b , slaveDataReceived %b ",masterDatatoSend,slaveDataReceived);
if(masterDataReceived==slaveDataToSend)
$display("SUCCESS from slave to master , slaveDataToSend %b , masterDataReceived %b ",slaveDataToSend, masterDataReceived);
else
$display("FAILURE from slave to master , slaveDataToSend %b , masterDataReceived %b ",slaveDataToSend, masterDataReceived);
end
```

If k=10 , then we are sure that the first testcase has been finished , we also check the index because we have only 5 testcases , if both conditions are true we reset k to 0 to start again and increment inex to try the next test case , we also reset cs to 1 and reset to 1 and wait for 5 delay and set the circuit by making

reset =0 and and cs =0 .



```
if(k==10&&index<5)begin
 k=0;
 index=index+1;

cs=1;
reset=1;
#5 cs=0;
reset=0;
end
```

CODE

```
module Slave_tb();

reg sclk;

reg reset;

reg cs;

reg [7:0] slaveDataToSend; //din

wire [7:0] slaveDataReceived; //dout

reg MOSI;

wire MISO;

reg[7:0] masterDatatoSend;

reg[7:0] masterDataReceived;
```

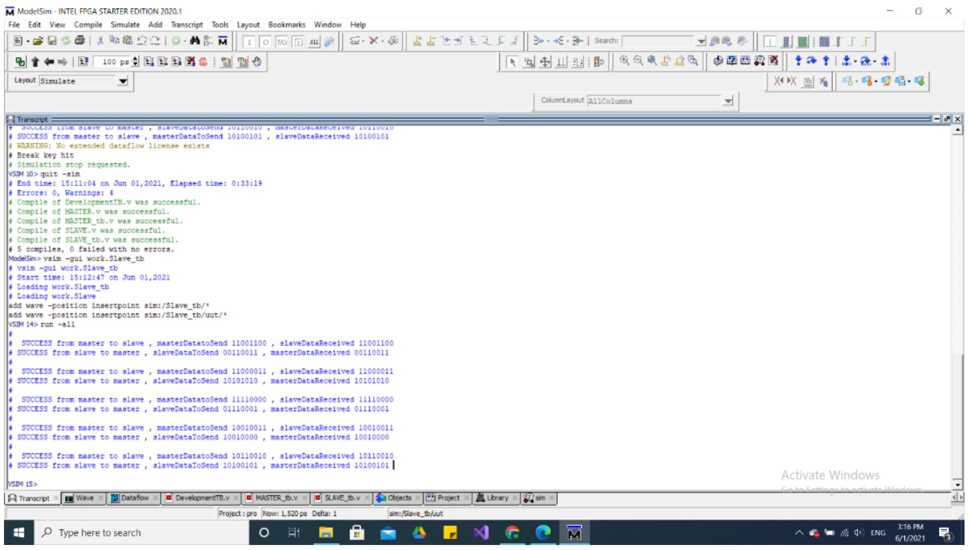
```
integer k;

integer i;

integer index;


wire [7:0] testcase_masterData [1:5];

wire [7:0] testcase_slaveData  [1:5];


assign testcase_masterData[1] = 8'b11001100;

assign testcase_slaveData [1] = 8'b00110011;


assign testcase_masterData[2] = 8'b11000011;

assign testcase_slaveData [2] = 8'b10101010;


assign testcase_masterData[3] = 8'b11110000;

assign testcase_slaveData [3] = 8'b01110001;


assign testcase_masterData[4] = 8'b10010011;

assign testcase_slaveData [4] = 8'b10010000;


assign testcase_masterData[5] = 8'b10110010;

assign testcase_slaveData [5] = 8'b10100101;



Slave uut( reset, slaveDataToSend,slaveDataReceived,sclk,cs, MOSI, MISO);

initial begin

sclk=0;
```

```
reset=1;

k=0;

cs=1;

index=1;

#5

cs=0;


assign slaveDataToSend=testcase_slaveData[index];

assign masterDatatoSend=testcase_masterData[index];

masterDataReceived=8'b00000000;


#10

reset=0;

for(i=0;i<300;i=i+1)begin

#5 sclk=~sclk;

end

#5

cs=1;

//#5

//cs=1;

end


always@(posedge sclk)begin

if(cs==0)begin

MOSI<=masterDatatoSend[7-k];

k<=k+1;
```

```
#10 masterDataReceived[8-k]<=MISO;

if(k==9)begin

$display();

if(slaveDataReceived==masterDatatoSend)

$display(" SUCCESS from master to slave , masterDatatoSend %b , slaveDataReceived
%b ",masterDatatoSend,slaveDataReceived);

else

$display(" FAILURE from master to slave , masterDatatoSend %b , slaveDataReceived
%b ",masterDatatoSend,slaveDataReceived);

if(masterDataReceived==slaveDataToSend)

$display("SUCCESS from slave to master , slaveDataToSend %b , masterDataReceived
%b ",slaveDataToSend, masterDataReceived);

else

$display("FAILURE from slave to master , slaveDataToSend %b , masterDataReceived
%b ",slaveDataToSend, masterDataReceived);

end

if(k==10&&index<5)begin

 k=0;

 index=index+1;


cs=1;

reset=1;

#5 cs=0;

reset=0;

end

end


//end
```

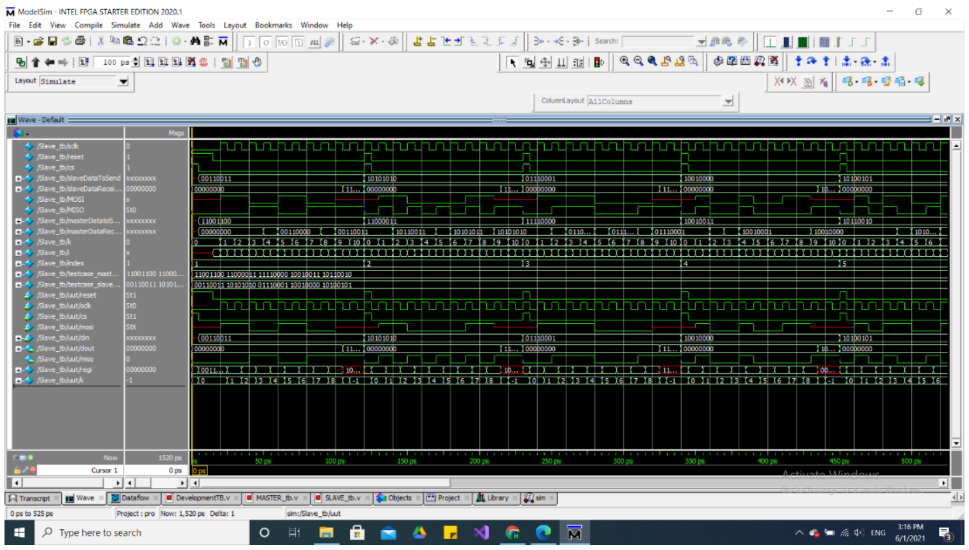
```
end

Endmodule
```

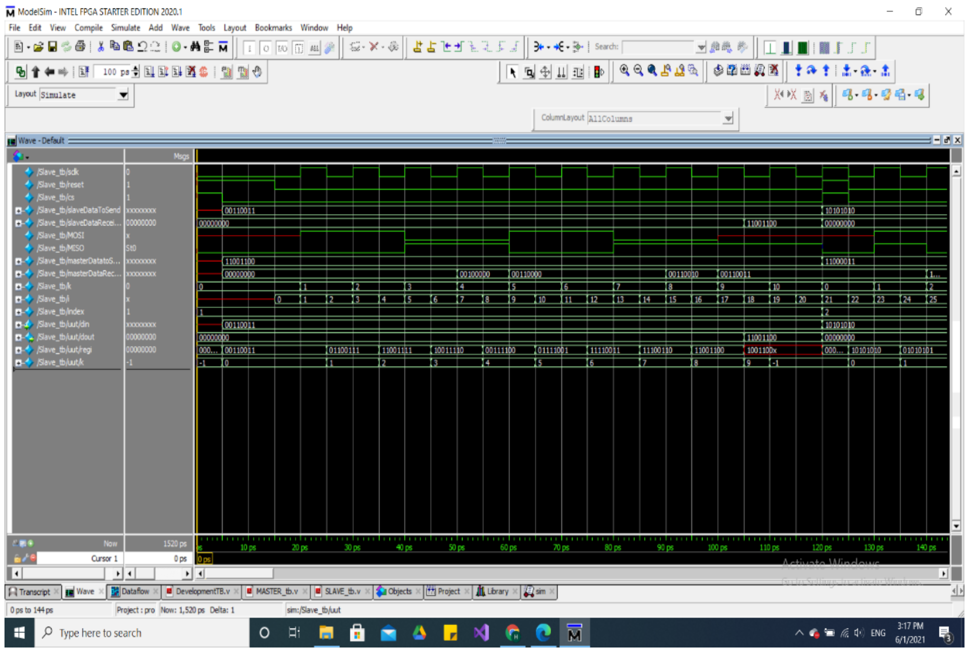
OUTPUT



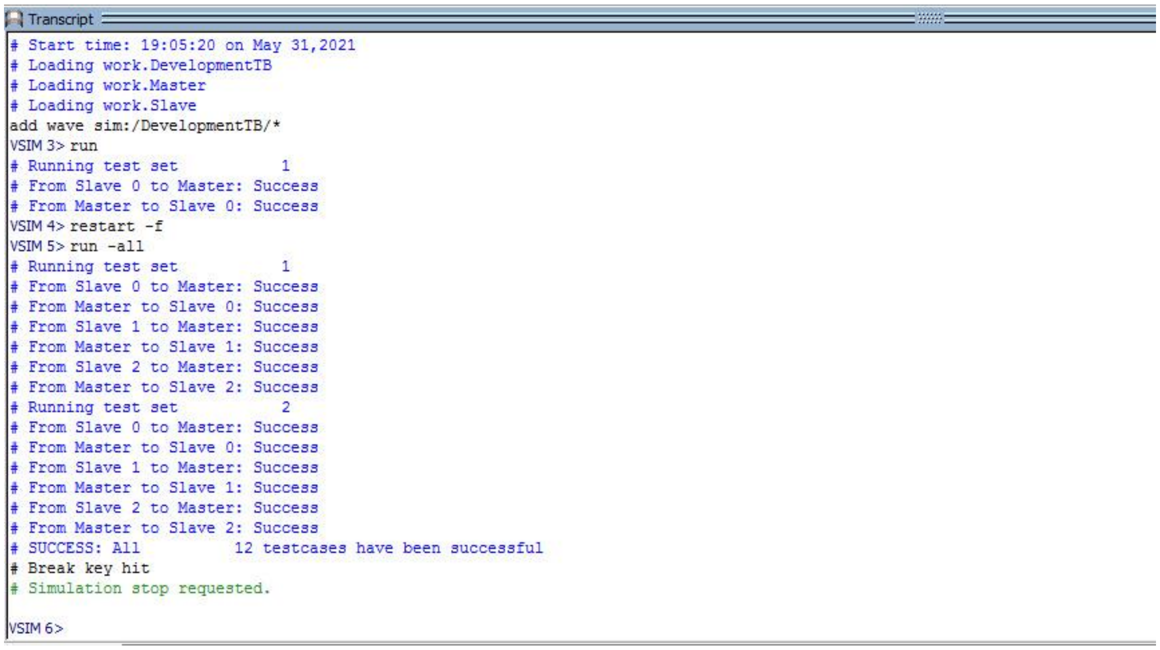
```
# SUCCESS from master to slave , masterDataToSend 10100101 , slaveDataReceived 10100101
# WARNING: No extended dataflow license exists
# Break key hit
# Simulation stop requested.
VSIM 10> quit -sim
# End time: 15:11:04 on Jun 01,2021, Elapsed time: 0:33:19
# Errors: 0, Warnings: 4
# Compile of DevelopmentTB.v was successful.
# Compile of MASTER.v was successful.
# Compile of MASTER_tb.v was successful.
# Compile of SLAVE.v was successful.
# Compile of SLAVE_tb.v was successful.
# 5 compiles, 0 failed with no errors.
ModelSim> vsim -gui work.Slave_tb
# vsim -gui work.Slave_tb
# Start time: 15:12:47 on Jun 01,2021
# Loading work.Slave_tb
# Loading work.Slave
add wave -position insertpoint sim:/Slave_tb/*
add wave -position insertpoint sim:/Slave_tb/uut/*
VSIM 14> run -all
#
#  SUCCESS from master to slave , masterDatatoSend 11001100 , slaveDataReceived 11001100
# SUCCESS from slave to master , slaveDataToSend 00110011 , masterDataReceived 00110011
#
#  SUCCESS from master to slave , masterDatatoSend 11000011 , slaveDataReceived 11000011
# SUCCESS from slave to master , slaveDataToSend 10101010 , masterDataReceived 10101010
#
#  SUCCESS from master to slave , masterDatatoSend 11110000 , slaveDataReceived 11110000
# SUCCESS from slave to master , slaveDataToSend 01110001 , masterDataReceived 01110001
#
#  SUCCESS from master to slave , masterDatatoSend 10010011 , slaveDataReceived 10010011
# SUCCESS from slave to master , slaveDataToSend 10010000 , masterDataReceived 10010000
#
#  SUCCESS from master to slave , masterDatatoSend 10110010 , slaveDataReceived 10110010
# SUCCESS from slave to master , slaveDataToSend 10100101 , masterDataReceived 10100101

VSIM 15>
```

WAVE



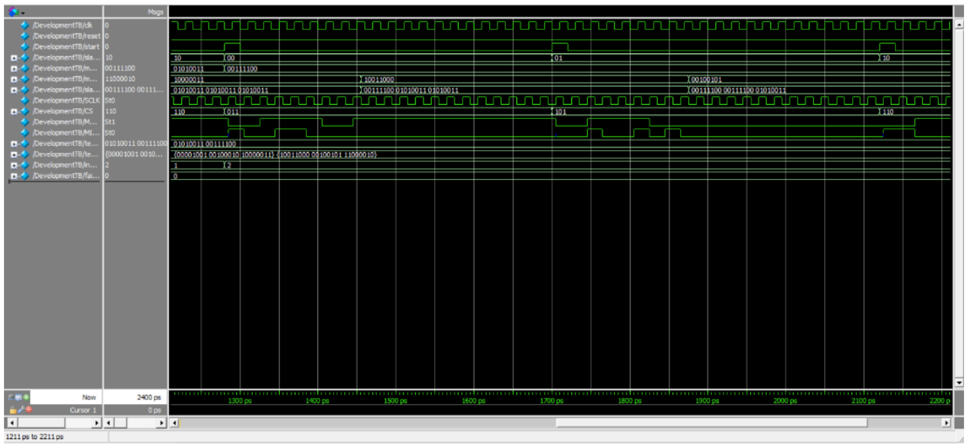


SIMULATION OUTPUT



```
Transcript
# Start time: 19:05:20 on May 31,2021
# Loading work.DevelopmentTB
# Loading work.Master
# Loading work.Slave
add wave sim:/DevelopmentTB/*
VSIM 3> run
# Running test set           1
# From Slave 0 to Master: Success
# From Master to Slave 0: Success
VSIM 4> restart -f
VSIM 5> run -all
# 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
# Break key hit
# Simulation stop requested.

VSIM 6>
```

SIMULATION WAVE


Msgs
/DevelopmentTB/clk 0
/DevelopmentTB/reset 0
/DevelopmentTB/start 0
/DevelopmentTB/sla... 10
/DevelopmentTB/m... 00111100
/DevelopmentTB/m... 11000010
/DevelopmentTB/sla... 00111100 00111...
/DevelopmentTB/SCLK St0
/DevelopmentTB/CS 110
/DevelopmentTB/M... St1
/DevelopmentTB/MI... St0
/DevelopmentTB/te... 01010011 00111100
/DevelopmentTB/te... {00001001 0010...
/DevelopmentTB/in... 2
/DevelopmentTB/fai... 0
Now 2400 ps
Cursor 1 0 ps
1300 ps
1400 ps
1500 ps
1600 ps
1700 ps
1800 ps
1900 ps
2000 ps
2100 ps
2200 ps
1211 ps to 2211 ps