**CODES:**

module SPI\_Protocol (

input wire clk, //System Clock

input wire reset, //Asynchronous system reset

input wire [15:0] datain, //Binary input vector

output wire spi\_cs\_l, //SPI Active-Low chip select

output wire spi\_sclk,

output wire spi\_data,

output [4:0]counter

);

//reg dclk;

reg [15:0] MOSI; //SPI shift register

reg [4:0] count; //Control Counter

reg cs\_l; //SPI chip select (active-low)

reg sclk;

reg [2:0]state;

always@(posedge clk or posedge reset)

if(reset) begin

MOSI <= 16'b0;

count <= 5'd16;

cs\_l <= 1'b1;

sclk <= 1'b0;

end

else begin

case(state)

0:begin

sclk <= 1'b0;

cs\_l <=1'b1;

state<= 1;

end

1:begin

sclk <= 1'b0;

cs\_l <= 1'b0;

MOSI <= datain[count-1];

count<= count-1;

state<=2;

end

2:begin

sclk <= 1'b1;

if(count >0)

state <=1;

else begin

count<=16;

state<=0;

end

end

default:state<=0;

endcase

end

assign spi\_cs\_l = cs\_l;

assign spi\_sclk = spi\_sclk;

assign spi\_data = MOSI; //MOSI

assign counter=count;

endmodule

**TestBench**

module TB();

//Inputs

reg clk;

reg reset;

reg [15:0] datain;

//Outputs

wire spi\_cs\_l;

wire spi\_sclk;

wire spi\_data;

wire [4:0] counter;

//Instantitate the UUT (Unit Under Test)

SPI\_Protocol dut(

.clk(clk),

.reset(reset),

.counter(counter),

.datain(datain),

.spi\_cs\_l(spi\_cs\_l),

.spi\_sclk(spi\_sclk),

.spi\_data(spi\_data)

);

initial begin

clk = 0;

reset = 1;

datain = 0;

end

//Clock Generation

always #5 clk=~clk;

initial begin

#10 reset=1'b0;

#10 datain=16'hA569;

#335 datain = 16'h2563;

#335 datain = 16'h9B63;

#335 datain=16'h6A61;

#335 datain=16'hA265;

#335 datain=16'h7564;

end