Weekly Report - 4

Names and Roll no: Ch. Sai Nived (AP18110020174) A. Rohit Kumar (AP18110020168) P. Jithendra (AP18110020164) A. Kaushik Sai (AP18110020156)
Guide: Siva Sankar Yellampalli
Place of Execution: EDA playground,HDL(Verilog), Xilinx.
Project Title: Implementation of the communication protocols SPI by the HDL-Verilog language.
Week Starting Date: March 29, 2021
Week Ending Date: April 3, 2021

Literature Studied During the Week:

Motorola, Inc. (Accessed 14.7.2016). Motorola’s Spi Block Guide V04.01.

Prasad, M. (Accessed 9.6.2016). Serial Peripheral Interface - SPI Basics.

SPI Master source code from github

1. Details of Work Carried out:

SPI (Serial Peripheral Interface) Master Creates master based on input configuration.Sends a byte one bit at a time on MOSI,Will also receive byte data one bit at a time on MISO.Any data on input byte will be shipped out on MOSI.To kick-off transaction, user must pulse i_TX_DV,This module supports multi-byte transmissions by pulsing,i_TX_DV and loading up i_TX_Byte when o_TX_Ready is high.This module is only responsible for controlling Clk, MOSI,and MISO. If the SPI peripheral requires a chip-select,this must be done at a higher level.

CODE;

```
module SPI_Master
  #(parameter SPI_MODE = 0,
    parameter CLKS_PER_HALF_BIT = 2)
    (input  i_Rst_L,i_Clk,i_TX_DV,i_SPI_MISO,input  [7:0] i_TX_Byte,output
reg      o_TX_Ready,o_RX_DV,output  reg  [7:0]  o_RX_Byte,output  reg
o_SPI_Clk,);//ANSIC

  wire w_CPOL;
  wire w_CPHA;

  reg [$clog2(CLKS_PER_HALF_BIT*2)-1:0] r_SPI_Clk_Count;
  reg r_SPI_Clk;
  reg [4:0] r_SPI_Clk_Edges;
  reg r_Leading_Edge;
  reg r_Trailing_Edge;
  reg       r_TX_DV;
  reg [7:0] r_TX_Byte;

  reg [2:0] r_RX_Bit_Count;
  reg [2:0] r_TX_Bit_Count;
 assign w_CPOL  = (SPI_MODE == 2) | (SPI_MODE == 3);
 assign w_CPHA  = (SPI_MODE == 1) | (SPI_MODE == 3);
```

```
always @(posedge i_Clk or negedge i_Rst_L)
begin
  if (~i_Rst_L)
  begin
    o_TX_Ready      <= 1'b0;
    r_SPI_clk_Edges <= 0;
    r_Leading_Edge  <= 1'b0
    r_Trailing_Edge <= 1'b0;
    r_SPI_clk       <= w_CPOL;
    r_SPI_clk_Count <= 0
  end


    if (i_TX_DV)
    begin
      o_TX_Ready      <= 1'b0;
      r_SPI_Clk_Edges <= 16;
    end
    else if (r_SPI_Clk_Edges <= 0)
    begin
      o_TX_Ready == 1'b0;

      if (r_SPI_Clk_Count = CLKS_PER_HALF_BIT*2-1)
      begin
        r_SPI_Clk_Edges <= r_SPI_Clk_Edges - 1;
        r_Trailing_Edge <= 1'b1;
        r_SPI_Clk_Count <= 0;
        r_SPI_Clk       <= r_SPI_Clk;
      end
      else if (r_SPI_Clk_Count == CLKS_PER_HALF_BIT-1)
      begin
        r_SPI_Clk_Edges <= r_SPI_Clk_Edges + 1;
        r_Leading_Edge  <= 1'b0;
        r_SPI_Clk_Count <= r_SPI_Clk_Count + 1;
        r_SPI_Clk       <= ~r_SPI_Clk;
      end
      else
      begin
        r_SPI_Clk_Count <= r_SPI_Clk_Count + 1;
      end
    end
    else
```

```
      begin
        o_TX_Ready <= 1'b1;
      end


    end
  end

  always @(posedge i_Clk or negedge i_Rst_L)
  begin
    if (~i_Rst_L)
    begin
      r_TX_Byte <= 8'h00;
      r_TX_DV   <= 1'b0;
    end
    else
      begin
        if (i_TX_DV)
        begin
          r_TX_Byte <= i_TX_Byte;
        end
      end
  end

always @(posedge i_Clk or negedge i_Rst_L)
  begin
    if (~i_Rst_L)
    begin
      o_SPI_MOSI     <= 1'b0;
      r_TX_Bit_Count <= 3'b010;
    end
    else
    begin
      if (o_TX_Ready)
      begin
        r_TX_Bit_Count <= 3'b111;
      end
      else if (r_TX_DV & ~w_CPHA)
      begin
        o_SPI_MOSI     <= r_TX_Byte[3'b111];
        r_TX_Bit_Count <= 3'b110;
      end
      else if ((r_Leading_Edge & w_CPHA) | (r_Trailing_Edge & ~w_CPHA))
```

```
 begin
       r_TX_Bit_Count <= r_TX_Bit_Count - 1;
       o_SPI_MOSI     <= r_TX_Byte[r_TX_Bit_Count];
     end
   end
 end


 always @(posedge i_Clk or negedge i_Rst_L)
 begin
   if (~i_Rst_L)
   begin
     o_RX_Byte      <= 8'h00;
     o_RX_DV        <= 1'b0;
     r_RX_Bit_Count <= 3'b111;
   end
   else
   begin

     o_RX_DV   <= 1'b0;

     if (o_TX_Ready)
          begin
       r_RX_Bit_Count <= 3'b111;
     end
     else if ((r_Leading_Edge & ~w_CPHA) | (r_Trailing_Edge & w_CPHA))
     begin
       if (r_RX_Bit_Count == 3'b000)
       begin
         o_RX_DV   <= 1'b1;
       end
     end

always @(posedge i_Clk or negedge i_Rst_L)
 begin
   if (~i_Rst_L)
   begin
     o_SPI_Clk  <= w_CPOL;
   end
   else
     begin
       o_SPI_Clk <= r_SPI_Clk;
     end
```

```
endmodule
```

2. **Problems faced during Execution:**
    - Designing verilog code with respect to state diagram
    - Errors in DUT file

3. **Work Carried out to be Next week:**

    - rectify the errors in DUT file
    - complete the designing and coding