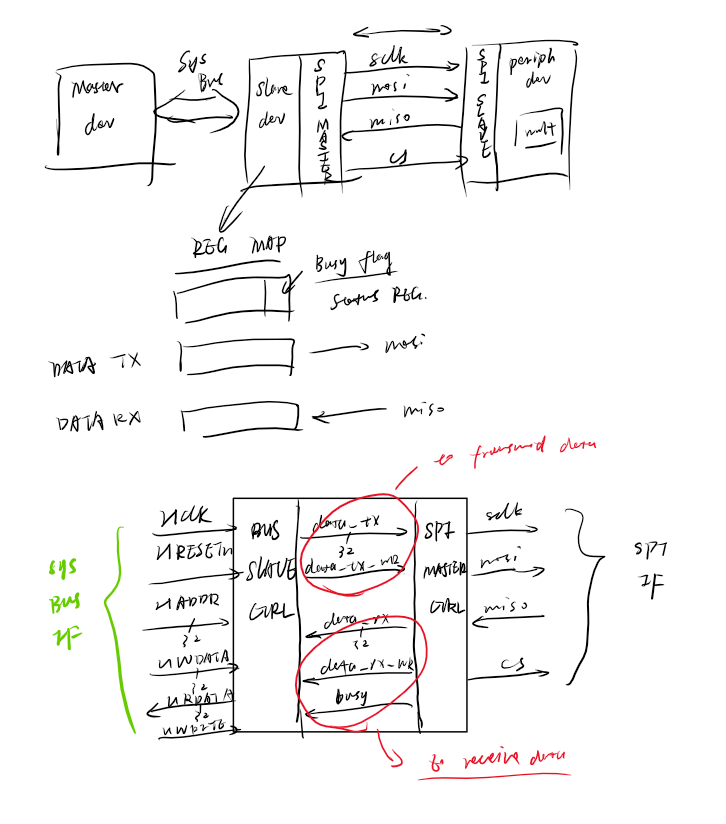
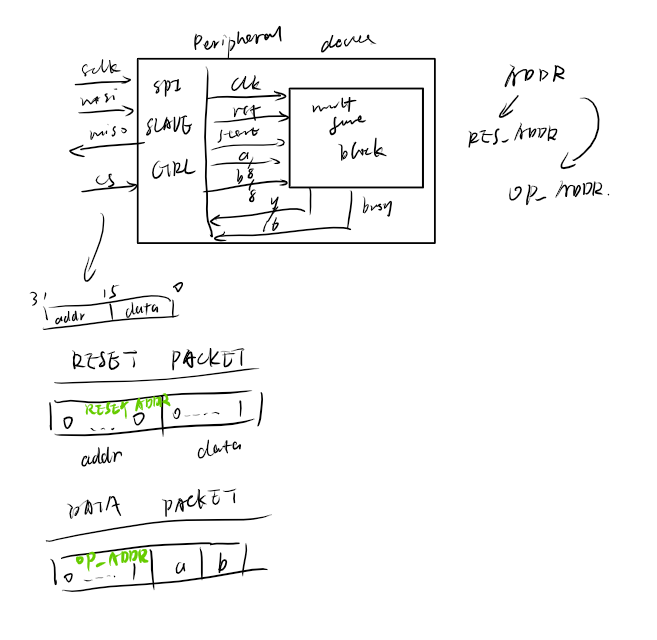
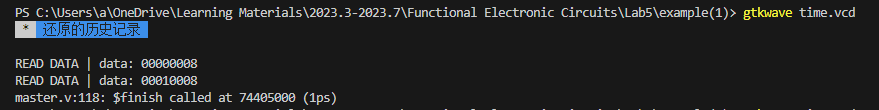
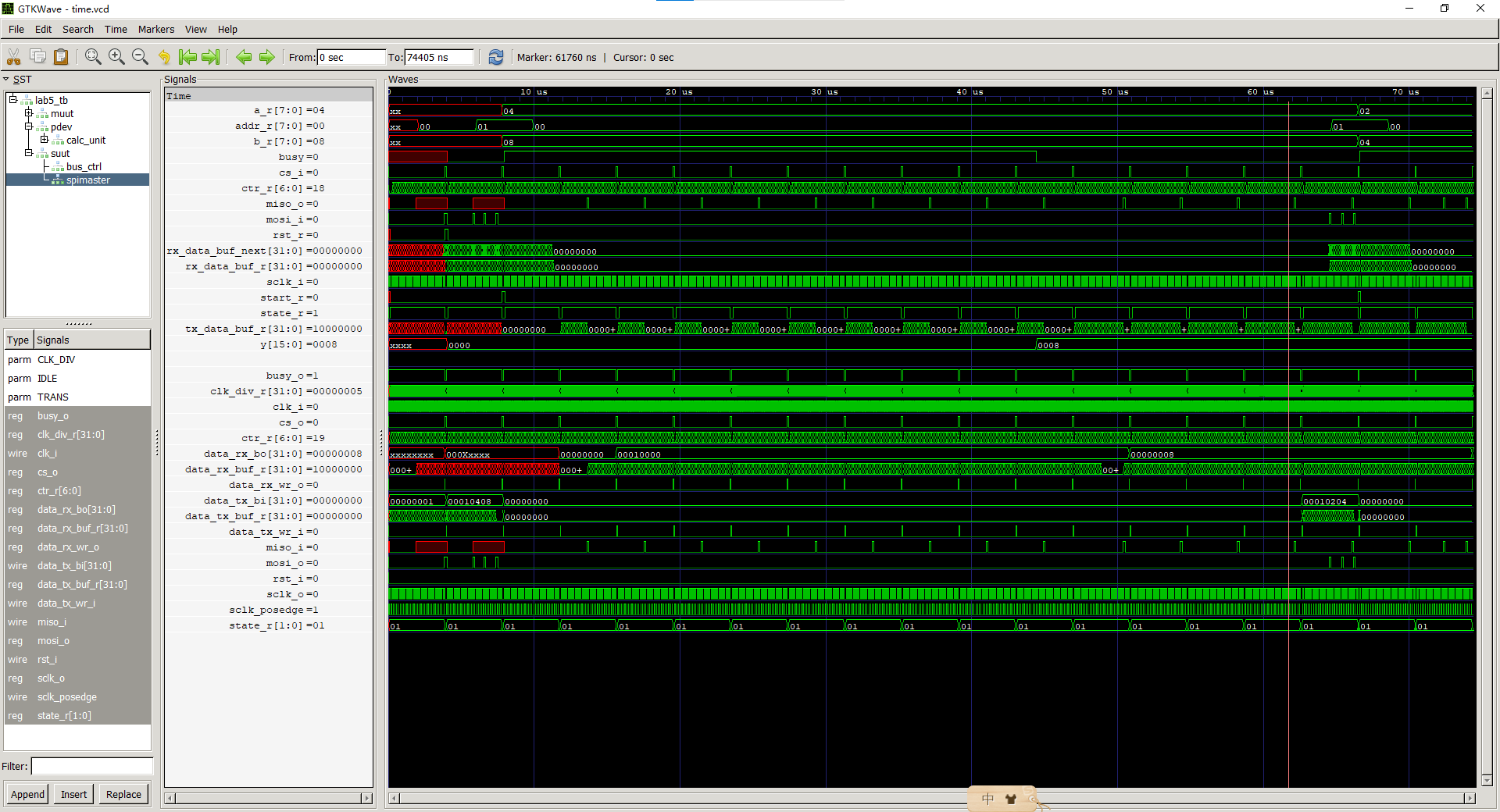
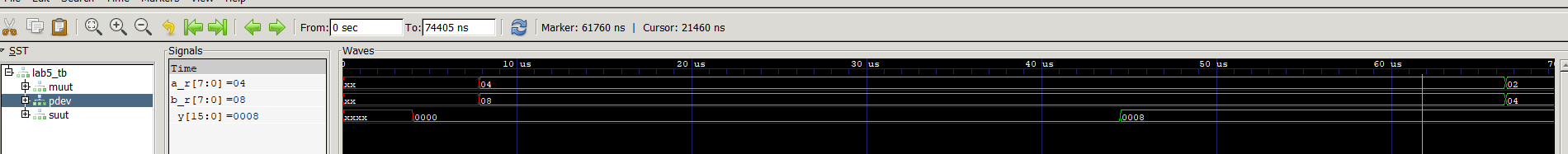
Functional Electronic Circuits Lab5

Student Name: CAO Xinyang

Student ID: 20321308

1. **The picture of the system  
     
   **
2. **The timing diagram with simulation results  
     
     
     
   My Variant in Lab3：  
     
     
   a = 4, b = 8, y = 8, it is correct**
3. **Code of the testbench and the device.**

***dev.v***

module dev (

    input clk\_i,

    input rst\_i,

    input [7:0] x\_bi,

    input [7:0] y\_bi,

    input start\_i,

    output reg [15:0] y\_bo,

    output reg busy\_bo

);

reg signed [31:0] a1\_op1, a1\_op2;

reg signed [31:0] a2\_op1, a2\_op2;

wire signed [31:0] a1\_res, a2\_res;

adder a1(

    .a\_bi(a1\_op1),

    .b\_bi(a1\_op2),

    .c\_bo(a1\_res)

);

adder a2(

    .a\_bi(a2\_op1),

    .b\_bi(a2\_op2),

    .c\_bo(a2\_res)

);

reg [31:0] m1\_op1, m1\_op2, y1;

reg m1\_start;

wire [15:0] m1\_res16;

wire [31:0] m1\_res = {16'h0, m1\_res16};

mult m1 (

    .clk\_i(clk\_i),

    .rst\_i(rst\_i),

    .a\_bi(m1\_op1[7:0]),

    .b\_bi(m1\_op2[7:0]),

    .start\_i(m1\_start),

    .busy\_o(m1\_busy),

    .y\_bo(m1\_res16)

);

// FSM control unit code

localparam N = 32;

localparam IDLE               = 0;

localparam CMP\_S              = 1;

localparam CALC\_MULT\_S1\_START = 2;

localparam CALC\_MULT\_S1       = 3;

localparam CALC\_MULT\_S2\_START = 4;

localparam CALC\_MULT\_S2       = 5;

localparam CALC\_B             = 6;

localparam CMP\_B              = 7;

localparam CALC\_S             = 8;

localparam CALC\_MULT\_S3       = 9;

localparam FINISH\_START       = 10;

localparam FINISH             = 11;

reg [3:0] state\_r;

reg [31:0] x\_r;

reg signed [31:0] s\_r;

reg [31:0] y\_r;

reg [31:0] b\_r;

always@(posedge clk\_i)

    if(rst\_i) begin

        x\_r <= 0;

        m1\_op1 <= 0;

        m1\_op2 <= 0;

        m1\_start <= 0;

        a1\_op1 <= 0;

        a1\_op2 <= 0;

        a2\_op1 <= 0;

        a2\_op2 <= 0;

        y\_bo   <= 0;

        y\_r    <= 0;

        busy\_bo  <= 0;

        s\_r    <= 0;

        y1     <= 0;

        state\_r <= IDLE;

    end else begin

        case(state\_r)

            IDLE:

                if(start\_i) begin

                    x\_r <= x\_bi;

                    y\_r <= 0;

                    busy\_bo <= 1;

                    b\_r <= 0;

                    s\_r <= 0;

                    // s = N-2

                    a1\_op1 <= N;

                    a1\_op2 <= -2;

                    state\_r <= CMP\_S;

                end

            CMP\_S:

                begin

                    s\_r <= a1\_res;

                    if(a1\_res > -3) begin

                        y\_r = y\_r << 1;

                        // 3\*y

                        m1\_op1 <= 3;

                        m1\_op2 <= y\_r;

                        m1\_start <= 1;

                        // y+1

                        a1\_op1 <= y\_r;

                        a1\_op2 <= 1;

                        state\_r <= CALC\_MULT\_S1\_START;

                    end else begin

                        y1 <= y\_r;

                        m1\_start <= 0;

                        state\_r <= CALC\_MULT\_S3;

                    end

                end

            CALC\_MULT\_S3:

                if(m1\_busy == 0) begin

                    m1\_op1 <= y1;

                    m1\_op2 <= y\_bi;

                    m1\_start <= 1;

                    state\_r <= FINISH\_START;

                end

            FINISH\_START:

                begin

                    m1\_start <= 0;

                    state\_r <= FINISH;

                end

            FINISH:

                if(m1\_busy == 0) begin

                    y\_bo <= m1\_res;

                    busy\_bo <= 0;

                    state\_r <= IDLE;

                end

            CALC\_MULT\_S1\_START: //waiting 1 clk for starting of the mult module

                begin

                    m1\_start <= 0;

                    state\_r <= CALC\_MULT\_S1;

                end

            CALC\_MULT\_S1:

                if(m1\_busy == 0) begin

                    // m1\_res = 3\*y

                    // a1\_res = y+1

                    // m1\_res \* a1\_res = 3\*y\*(y+1)

                    m1\_op1 <= m1\_res;

                    m1\_op2 <= a1\_res;

                    m1\_start <= 1;

                    state\_r <= CALC\_MULT\_S2\_START;

                end

            CALC\_MULT\_S2\_START: //waiting 1 clk for starting of the mult module

                begin

                    m1\_start <= 0;

                    state\_r <= CALC\_MULT\_S2;

                end

            CALC\_MULT\_S2:

                if(m1\_busy == 0) begin

                    // 3\*y(y+1)+1

                    a1\_op1 <= m1\_res;

                    a1\_op2 <= 1;

                    state\_r <= CALC\_B;

                end

            CALC\_B:

                begin

                    b\_r <= a1\_res << s\_r;

                    state\_r <= CMP\_B;

                end

            CMP\_B:

                if(x\_r >= b\_r) begin

                    // x-b

                    a1\_op1 <= x\_r;

                    a1\_op2 <= -b\_r;

                    // y+1

                    a2\_op1 <= y\_r;

                    a2\_op2 <= 1;

                    state\_r <= CALC\_S;

                end else begin

                    state\_r <= CALC\_S;

                end

            CALC\_S:

                begin

                    if(x\_r >= b\_r) begin

                        //x = x-b

                        x\_r <= a1\_res;

                        //y = y+1

                        y\_r <= a2\_res;

                    end

                    // s-3

                    a1\_op1 <= s\_r;

                    a1\_op2 <= -3;

                    state\_r <= CMP\_S;

                end

        endcase

    end

endmodule

***periph\_dev.v***

module periph\_dev (

    input  sclk\_i,

    input  mosi\_i,

    output reg miso\_o,

    input  cs\_i

);

localparam RESET\_ADDR    = 8'h0;

localparam OP\_ADDR  = 8'h1;

reg [7:0]  addr\_r, cmd\_r;

reg [31:0] rx\_data\_buf\_r, tx\_data\_buf\_r;

reg start\_r;

reg [7:0] a\_r, b\_r;

wire [15:0] y;

wire busy;

reg rst\_r;

dev calc\_unit(

    .clk\_i(sclk\_i),

    .rst\_i(rst\_r),

    .start\_i(start\_r),

    .x\_bi(b\_r),

    .y\_bi(a\_r),

    .y\_bo(y),

    .busy\_bo(busy)

);

localparam STATE0 = 0;

localparam STATE1 = 1;

reg state\_r = STATE0;

reg [6:0] ctr\_r = 0;

wire [31:0] rx\_data\_buf\_next = {rx\_data\_buf\_r[30:0], mosi\_i};

localparam ADDR\_CTR\_VAL = 15;

localparam DATA\_CTR\_VAL  = 30;

always@(posedge sclk\_i) begin

    miso\_o        <= tx\_data\_buf\_r[31];

    if(state\_r == STATE0) begin

        tx\_data\_buf\_r <= {14'h0, busy, y, 1'b0};

        miso\_o        <= 0;

    end else

        tx\_data\_buf\_r <= tx\_data\_buf\_r << 1;

end

always@(negedge sclk\_i)

        if(cs\_i == 0) begin

            case(state\_r)

                STATE0:

                    begin

                        start\_r       <= 0;

                        ctr\_r         <= 0;

                        rst\_r         <= 0;

                        state\_r       <= STATE1;

                        ctr\_r         <= 0;

                        rx\_data\_buf\_r <= rx\_data\_buf\_next;

                    end

                STATE1:

                    begin

                        ctr\_r         <= ctr\_r + 1;

                        rx\_data\_buf\_r <= rx\_data\_buf\_next;

                        case(ctr\_r)

                                ADDR\_CTR\_VAL:

                                    begin

                                        addr\_r <= rx\_data\_buf\_r[7:0];

                                    end

                                DATA\_CTR\_VAL:

                                    begin

                                        state\_r <= STATE0;

                                        case(addr\_r)

                                            OP\_ADDR:

                                                if(busy == 0) begin

                                                    a\_r <= rx\_data\_buf\_next[15:8];

                                                    b\_r <= rx\_data\_buf\_next[7:0];

                                                    start\_r <= 1;

                                                end

                                            RESET\_ADDR:

                                                rst\_r <= rx\_data\_buf\_next[0];

                                        endcase

                                    end

                            endcase

                    end

            endcase

        end

endmodule

***master.v***

module master(

    input HCLK\_i,

    input HRESETn\_i,

    input [31:0] HRDATA\_bi,

    output reg [31:0] HADDR\_bo,

    output reg [31:0] HWDATA\_bo,

    output reg HWRITE\_o

);

localparam STATUS\_ADDR = 8'h0;

localparam OP\_ADDR     = 8'h1;

task write(

    input [31:0] addr,

    input [31:0] data

);

    begin

        @(posedge HCLK\_i);

        HADDR\_bo <= addr;

        HWRITE\_o <= 1;

        @(posedge HCLK\_i);

        HWDATA\_bo <= data;

        HWRITE\_o <= 1;

        @(posedge HCLK\_i);

        HWRITE\_o <= 0;

    end

endtask

task read(

    input  [31:0] addr,

    output [31:0] data

);

    begin

        @(posedge HCLK\_i);

        HADDR\_bo <= addr;

        HWRITE\_o <= 0;

        @(posedge HCLK\_i);

        @(posedge HCLK\_i);

        data <= HRDATA\_bi;

        @(posedge HCLK\_i);

    end

endtask

task wait\_ready();

begin

    begin : wait\_busy

        while(1) begin

            read(32'h0, data);

            if(data == 1)

                disable wait\_busy;

        end

    end

    begin : wait\_ready

        while(1) begin

            read(32'h0, data);

            if(data == 0)

                disable wait\_ready;

        end

    end

end

endtask

reg [31:0] data = 1;

initial begin

    write(32'h1, {8'h0, STATUS\_ADDR, 16'h1});

    wait\_ready;

    write(32'h1, {8'h0, OP\_ADDR, 8'h4, 8'h8});

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    read(32'h2, data);

    $display("READ DATA | data: %h", data);

    write(32'h1, {8'h0, OP\_ADDR, 8'h2, 8'h4});

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    write(32'h1, 0);

    wait\_ready;

    read(32'h2, data);

    $display("READ DATA | data: %h", data);

    $finish;

end

endmodule

***bus\_slave.v***

module bus\_slave(

    input HCLK\_i,

    input HRESETn\_i,

    output reg [31:0] HRDATA\_bo,

    input [31:0] HADDR\_bi,

    input [31:0] HWDATA\_bi,

    input HWRITE\_i,

    input  [31:0] data\_rx\_bi,

    input  data\_rx\_wr\_i,

    input  busy\_i,

    output reg [31:0] data\_tx\_bo,

    output reg data\_tx\_wr\_o

);

parameter BASE\_ADDR        = 0;

localparam STATUS\_REG\_ADDR = BASE\_ADDR + 0;

localparam DATA\_TX\_ADDR    = BASE\_ADDR + 1;

localparam DATA\_RX\_ADDR    = BASE\_ADDR + 2;

reg [7:0]  status\_r;

reg [31:0] data\_rx\_r;

always@(posedge HCLK\_i)

    if(HRESETn\_i == 0) begin

        HRDATA\_bo     <= 0;

        data\_tx\_bo    <= 0;

        data\_tx\_wr\_o  <= 0;

    end else begin

        data\_tx\_wr\_o <= 0;

        if(HWRITE\_i)

            case(HADDR\_bi)

                DATA\_TX\_ADDR:

                    begin

                        data\_tx\_bo   <= HWDATA\_bi;

                        data\_tx\_wr\_o <= 1;

                    end

            endcase

        else begin

            case(HADDR\_bi)

                STATUS\_REG\_ADDR:

                    HRDATA\_bo <= {24'h0, status\_r};

                DATA\_TX\_ADDR:

                    HRDATA\_bo <= data\_tx\_bo;

                DATA\_RX\_ADDR:

                    HRDATA\_bo <= data\_rx\_r;

                default:

                    HRDATA\_bo <= 0;

            endcase

        end

    end

always@(posedge HCLK\_i)

    if(HRESETn\_i == 0) begin

        status\_r  <= 0;

        data\_rx\_r <= 0;

    end else begin

        status\_r <= {7'h0, busy\_i};

        if(data\_rx\_wr\_i)

            data\_rx\_r <= data\_rx\_bi;

    end

endmodule

***slave.v***

module slave(

    input HCLK\_i,

    input HRESETn\_i,

    output [31:0] HRDATA\_bo,

    input [31:0] HADDR\_bi,

    input [31:0] HWDATA\_bi,

    input HWRITE\_i,

    output sclk\_o,

    output mosi\_o,

    input  miso\_i,

    output cs\_o

);

parameter BASE\_ADDR = 0;

parameter CLK\_DIV   = 10;

wire [31:0] data\_rx, data\_tx;

wire data\_rx\_wr, data\_tx\_wr;

wire busy;

spi\_master  #(.CLK\_DIV(CLK\_DIV))

spimaster (

    .clk\_i(HCLK\_i),

    .rst\_i(!HRESETn\_i),

    .sclk\_o(sclk\_o),

    .mosi\_o(mosi\_o),

    .miso\_i(miso\_i),

    .cs\_o(cs\_o),

    .data\_rx\_bo(data\_rx),

    .data\_rx\_wr\_o(data\_rx\_wr),

    .busy\_o(busy),

    .data\_tx\_bi(data\_tx),

    .data\_tx\_wr\_i(data\_tx\_wr)

);

bus\_slave #(.BASE\_ADDR(BASE\_ADDR))

bus\_ctrl (

    .HCLK\_i(HCLK\_i),

    .HRESETn\_i(HRESETn\_i),

    .HRDATA\_bo(HRDATA\_bo),

    .HADDR\_bi(HADDR\_bi),

    .HWDATA\_bi(HWDATA\_bi),

    .HWRITE\_i(HWRITE\_i),

    .data\_rx\_bi(data\_rx),

    .data\_rx\_wr\_i(data\_rx\_wr),

    .busy\_i(busy),

    .data\_tx\_bo(data\_tx),

    .data\_tx\_wr\_o(data\_tx\_wr)

);

endmodule

***spi\_master.v***

module spi\_master (

    input clk\_i,

    input rst\_i,

    output reg sclk\_o,

    output reg mosi\_o,

    input  miso\_i,

    output reg cs\_o,

    output reg [31:0] data\_rx\_bo,

    output reg data\_rx\_wr\_o,

    output reg busy\_o,

    input [31:0] data\_tx\_bi,

    input data\_tx\_wr\_i

);

parameter CLK\_DIV = 10;

localparam IDLE  = 0;

localparam TRANS = 1;

reg [6:0]  ctr\_r;

reg [31:0] clk\_div\_r;

reg [1:0]  state\_r;

reg [31:0] data\_rx\_buf\_r;

reg [31:0] data\_tx\_buf\_r;

wire sclk\_posedge = (clk\_div\_r == CLK\_DIV) & (sclk\_o == 0);

always@(posedge clk\_i)

    if(rst\_i) begin

        clk\_div\_r     <= 0;

        ctr\_r         <= 0;

        data\_rx\_buf\_r <= 0;

        sclk\_o        <= 0;

    end else begin

        if(busy\_o) begin

            clk\_div\_r <= clk\_div\_r + 1;

            if(clk\_div\_r == CLK\_DIV) begin

                sclk\_o <= ~sclk\_o;

                clk\_div\_r <= 0;

                if(sclk\_o) begin

                    ctr\_r         <= ctr\_r + 1;

                    data\_rx\_buf\_r <= {data\_rx\_buf\_r[30:0], miso\_i};

                end

            end

        end

    end

always@(posedge clk\_i)

    if(rst\_i) begin

        data\_tx\_buf\_r <= 0;

        mosi\_o <= 0;

    end else begin

        if(data\_tx\_wr\_i)

            data\_tx\_buf\_r <= data\_tx\_bi;

        if(sclk\_posedge) begin

            mosi\_o <= data\_tx\_buf\_r[31];

            data\_tx\_buf\_r <= data\_tx\_buf\_r << 1;

        end

    end

always@(posedge clk\_i)

    if(rst\_i) begin

        state\_r <= 0;

        busy\_o  <= 0;

        cs\_o    <= 1;

        data\_rx\_wr\_o  <= 0;

    end else begin

        case(state\_r)

            IDLE:

                begin

                    data\_rx\_wr\_o  <= 0;

                    if(data\_tx\_wr\_i) begin

                        busy\_o <= 1;

                        state\_r <= TRANS;

                        cs\_o  <= 0;

                        ctr\_r <= 0;

                    end

                end

            TRANS:

                begin

                    if(ctr\_r == 6'h20) begin

                        cs\_o    <= 1;

                        busy\_o  <= 0;

                        ctr\_r   <= 0;

                        state\_r <= IDLE;

                        data\_rx\_wr\_o  <= 1;

                        data\_rx\_bo <= data\_rx\_buf\_r;

                    end

                end

            default:

                state\_r <= IDLE;

        endcase

    end

endmodule

***lab5\_tb.v***

`timescale 1ns/1ps

module lab5\_tb;

reg clk, rst;

wire [31:0] HRDATA;

wire [31:0] HADDR;

wire [31:0] HWDATA;

wire HWRITE;

parameter BASE\_ADDR = 0;

parameter CLK\_DIV   = 10;

wire sclk, mosi, cs;

wire miso;

master muut(

    .HCLK\_i(clk),

    .HRESETn\_i(rst),

    .HRDATA\_bi(HRDATA),

    .HADDR\_bo(HADDR),

    .HWDATA\_bo(HWDATA),

    .HWRITE\_o(HWRITE)

);

slave #(.BASE\_ADDR(0), .CLK\_DIV(5))

suut(

    .HCLK\_i(clk),

    .HRESETn\_i(rst),

    .HRDATA\_bo(HRDATA),

    .HADDR\_bi(HADDR),

    .HWDATA\_bi(HWDATA),

    .HWRITE\_i(HWRITE),

    .sclk\_o(sclk),

    .mosi\_o(mosi),

    .miso\_i(miso),

    .cs\_o(cs)

);

periph\_dev pdev (

    .sclk\_i(sclk),

    .mosi\_i(mosi),

    .miso\_o(miso),

    .cs\_i(cs)

);

always #5 clk = ~clk;

initial begin

    $dumpfile("time.vcd");

    $dumpvars(0, lab5\_tb);

    clk = 0;

    rst = 0;

    #20

    rst = 1;

end

endmodule