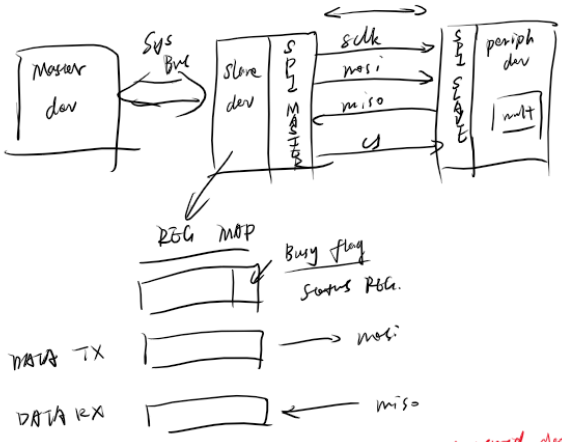
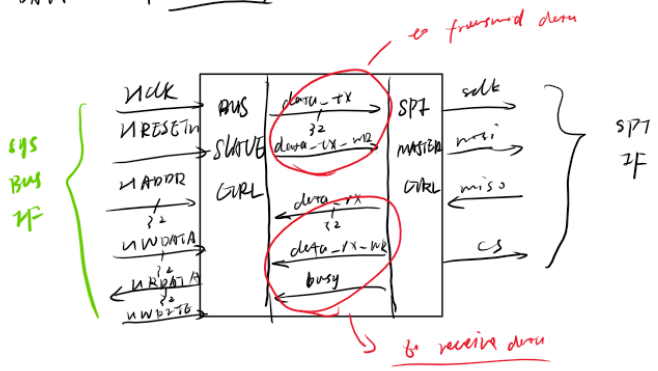
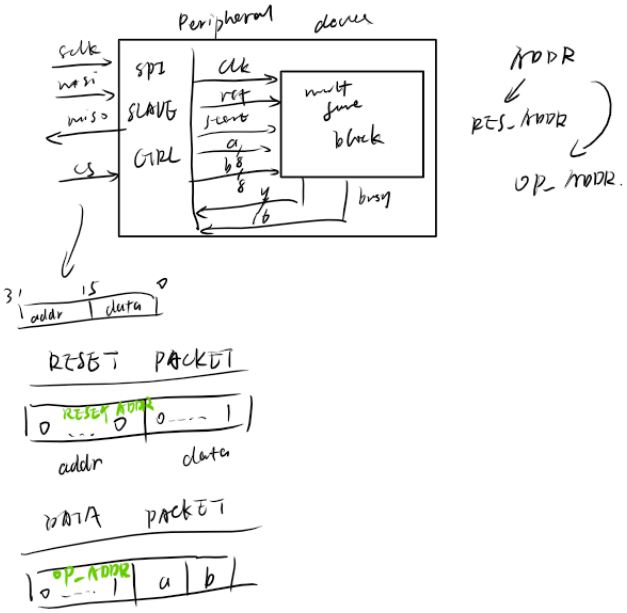
Functional Electronic Circuits Lab5

Student Name: CAO Xinyang
Student ID: 20321308

1. The picture of the system



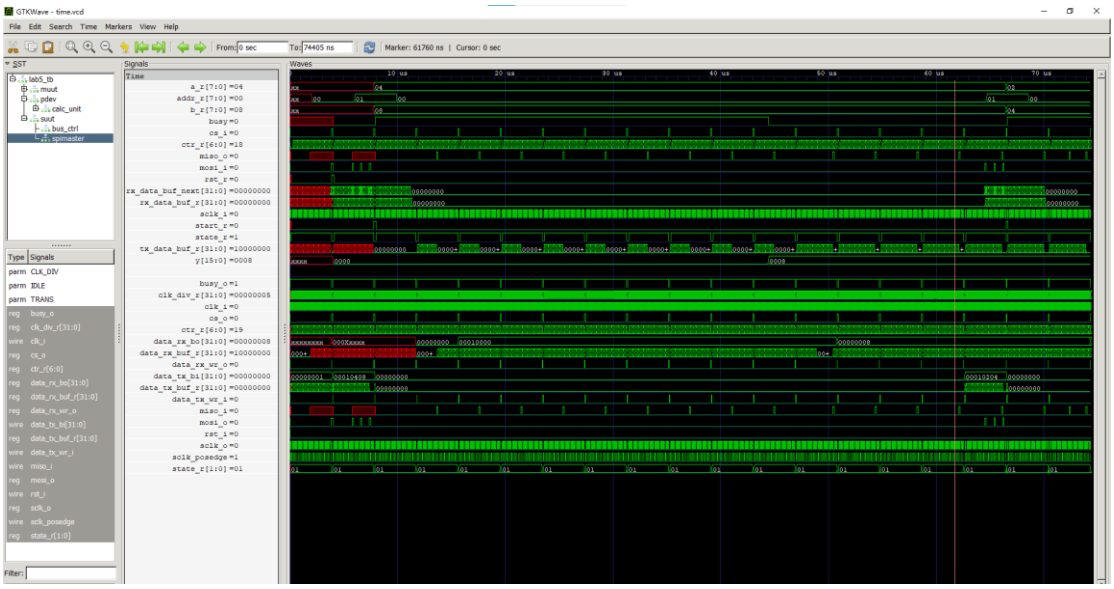




2. The timing diagram with simulation results

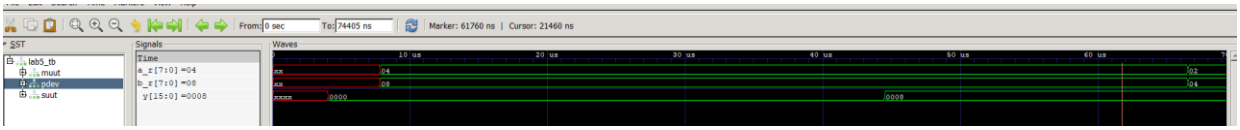
```
PS C:\Users\a\OneDrive\Learning Materials\2023.3-2023.7\Functional Electronic Circuits\Lab5\example(1)> gtkwave time.vcd
还原的历史记录

READ DATA | data: 00000008
READ DATA | data: 00010008
master.v:118: $finish called at 74405000 (1ps)
```



My Variant in Lab3：

8	$y = a \cdot \sqrt[3]{b}$	Number of adders: 2 Number of multipliers: 1



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;
    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
```