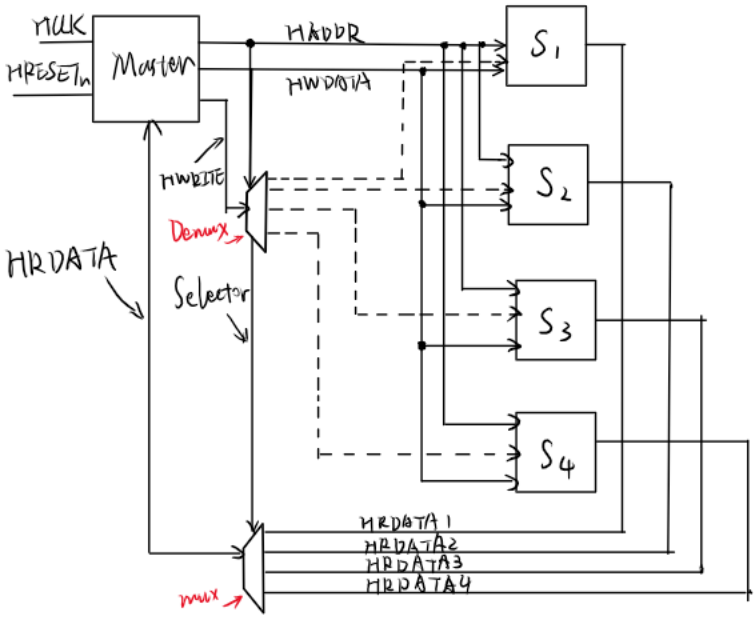
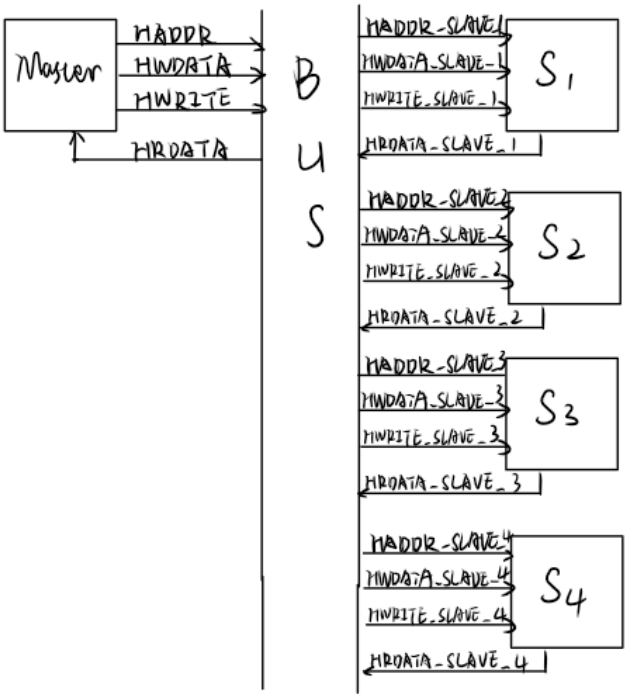
Functional Electronic Circuits Lab4

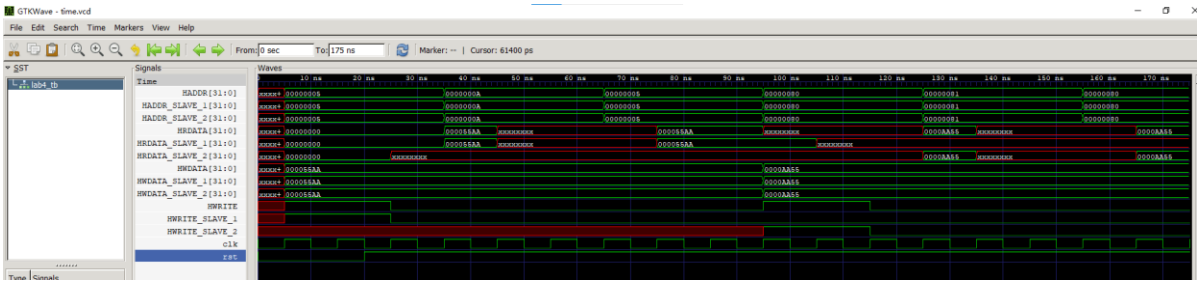
Student Name: CAO Xinyang
Student ID: 20321308

1. The picture of the system





2. The timing diagram with simulation results



3. Code of the testbench and the device

busDevice.v

```
module bus(
    input [31:0] HADDR_MASTER_bi,
    input [31:0] HWDATA_MASTER_bi,
    input HWRITE_MASTER_i,
    output reg [31:0] HRDATA_MASTER_bo,

    output [31:0] HADDR_SLAVE_1_bo,
    output [31:0] HWDATA_SLAVE_1_bo,
    output reg HWRITE_SLAVE_1_o,
    input [31:0] HRDATA_SLAVE_1_bi,
    output [31:0] HADDR_SLAVE_2_bo,
    output [31:0] HWDATA_SLAVE_2_bo,
    output reg HWRITE_SLAVE_2_o,
    input [31:0] HRDATA_SLAVE_2_bi
);
assign HADDR_SLAVE_1_bo = HADDR_MASTER_bi;
assign HADDR_SLAVE_2_bo = HADDR_MASTER_bi;
assign HWDATA_SLAVE_1_bo = HWDATA_MASTER_bi;
assign HWDATA_SLAVE_2_bo = HWDATA_MASTER_bi;
always@*
    if (HADDR_MASTER_bi < 32'h80) begin
        HWRITE_SLAVE_1_o <= HWRITE_MASTER_i;
        HRDATA_MASTER_bo <= HRDATA_SLAVE_1_bi;
    end else begin
        HWRITE_SLAVE_2_o <= HWRITE_MASTER_i;
        HRDATA_MASTER_bo <= HRDATA_SLAVE_2_bi;
    end
endmodule
```

masterDevice.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

);
task write(
    input [31:0] addr,
    input [31:0] data
);
    begin
        @(posedge HCLK_i);
        HADDR_bo <= addr;
        HWRITE_o <= 1;
        HWDATA_bo <= data;
        @(posedge HCLK_i);
        @(posedge HCLK_i);
        HWRITE_o <= 0;
    end
endtask
task read(
    input [31:0] addr
);
    begin
        @(posedge HCLK_i);
        HADDR_bo <= addr;
        HWRITE_o <= 0;
        @(posedge HCLK_i);
        @(posedge HCLK_i);
        $display("READ DATA | addr: %h, data: %h", addr, HRDATA_bi);
    end
endtask
initial begin
    write(32'h5, 32'h55AA);
    read(32'hA);
    read(32'h5);
    write(32'h80, 32'hAA55);
    read(32'h81);
    read(32'h80);
    $finish;
end
```

```
endmodule
```

slaveDevice.v

```
module 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
);
reg [31:0] mem [255:0];
always@(posedge HCLK_i)
    if (HRESETn_i == 0)
        HRDATA_bo <= 0;
    else begin
        if (HWRITE_i)
            mem[HADDR_bi] <= HWDATA_bi;
        else
            HRDATA_bo <= mem[HADDR_bi];
    end
endmodule
```

testBench.v

```
`timescale 1ns/1ps

module lab4_tb;
reg clk, rst;
wire [31:0] HRDATA, HADDR, HWDATA;
wire HWRITE;
wire [31:0] HRDATA_SLAVE_1, HADDR_SLAVE_1, HWDATA_SLAVE_1;
wire HWRITE_SLAVE_1;
wire [31:0] HRDATA_SLAVE_2, HADDR_SLAVE_2, HWDATA_SLAVE_2;
wire HWRITE_SLAVE_2;
bus buut (
    .HADDR_MASTER_bi(HADDR),
    .HWDATA_MASTER_bi(HWDATA),
    .HRDATA_MASTER_bo(HRDATA),
    .HWRITE_MASTER_i(HWRITE),
    .HADDR_SLAVE_1_bo(HADDR_SLAVE_1),
    .HWDATA_SLAVE_1_bo(HWDATA_SLAVE_1),
```

```
    .HRDATA_SLAVE_1_bi(HRDATA_SLAVE_1),
    .HWRITE_SLAVE_1_o(HWRITE_SLAVE_1),
    .HADDR_SLAVE_2_bo(HADDR_SLAVE_2),
    .HWDATA_SLAVE_2_bo(HWDATA_SLAVE_2),
    .HRDATA_SLAVE_2_bi(HRDATA_SLAVE_2),
    .HWRITE_SLAVE_2_o(HWRITE_SLAVE_2)
);
master muut(
    .HCLK_i(clk),
    .HRESETn_i(rst),
    .HRDATA_bi(HRDATA),
    .HADDR_bo(HADDR),
    .HWDATA_bo(HWDATA),
    .HWRITE_o(HWRITE)
);
slave suut1(
    .HCLK_i(clk),
    .HRESETn_i(rst),
    .HRDATA_bo(HRDATA_SLAVE_1),
    .HADDR_bi(HADDR_SLAVE_1),
    .HWDATA_bi(HWDATA_SLAVE_1),
    .HWRITE_i(HWRITE_SLAVE_1)
);
slave suut2(
    .HCLK_i(clk),
    .HRESETn_i(rst),
    .HRDATA_bo(HRDATA_SLAVE_2),
    .HADDR_bi(HADDR_SLAVE_2),
    .HWDATA_bi(HWDATA_SLAVE_2),
    .HWRITE_i(HWRITE_SLAVE_2)
);
always #5 clk = ~clk;
initial begin
    $dumpfile("time.vcd");
    $dumpvars(1, lab4_tb);
    clk = 0;
    rst = 0;
    #20
    rst = 1;

end
endmodule
```