集成电路设计仿真与验证技术
实验报告

题　　目：单端口 SRAM 仿真验证

学　　院：电子工程学院

系　　部：微电子学系

专　　业：集成电路设计与集成系统

班　　级：电路 2101 赵祖康

姓名学号：05219038

指导教师：

一、 实验目的（重点说明本实验要学习的知识点以及要达到的目标）。

通过本次实验，要求学生掌握 Interface 机制的基本原理和具体实现方法。

二、 实验要求：对单端口 SDRAM 进行仿真验证其正确性

三、 硬件平台：HP 服务器，软件环境：VCS

四、 实验原理

实现一个 dut 和 ref（在这里确保这个 ref 是完全正确的），然后利用 dut 和 ref 做 diff，验证 dut 的正确性

五、 实验过程

先实现一个 interface

```
interface arb_if(input bit clk);
    logic           en          ;
    logic           wr          ;
    logic  [7:0]    wr_data     ;
    logic  [7:0]    rd_data     ;
    logic  [31:0]    addr        ;
    // logic [7:0] dutRdata;
    modport DUT
                    (input clk ,en ,wr ,wr_data , addr ,
                        output rd_data);
    modport testbench
                    (input clk , rd_data ,
                    output en ,wr ,wr_data , addr);
    modport monitor_
                    (
                        input clk,en,wr,wr_data,addr,
                        output rd_data
                    );
endinterface
```

然后实现 dut，ref

```
module dut(arb_if.DUT arbif);

    parameter WIDTH=8;
    parameter DEPTH =32;
    logic [WIDTH-1:0]mem[DEPTH-1:0];


    always@(negedge arbif.clk)begin
        if(arbif.en & arbif.wr)
            mem[arbif.addr]<= arbif.wr_data;
        end

    assign arbif.rd_data=((!arbif.wr)& arbif.en)? mem[arbif.addr] : 8'd0;
```

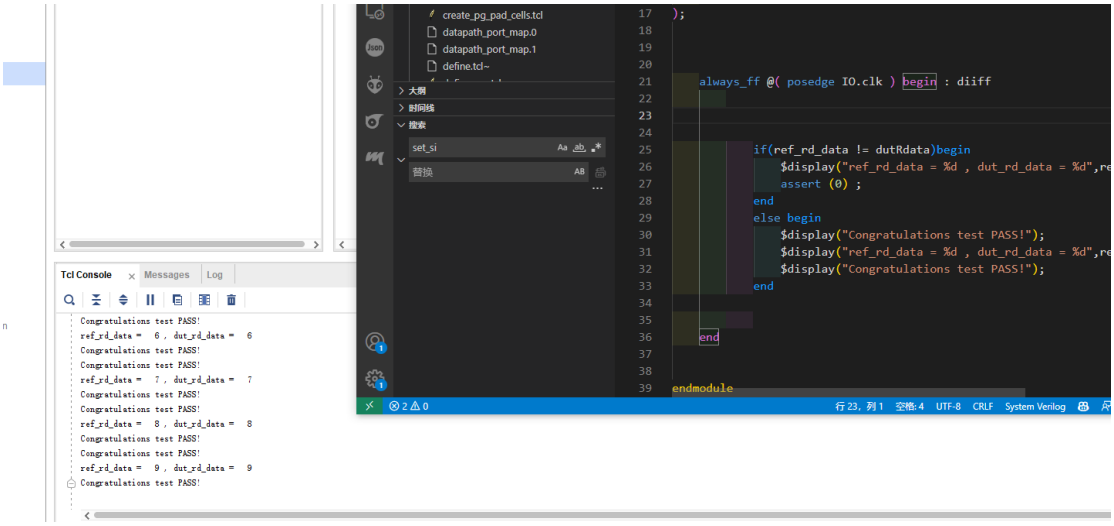
```
module ref_1 #(
    parameter WIDTH=8,
    parameter DEPTH =32
) (
    input bit              clk    ,
    input logic            en           ,
    input logic            wr           ,
    input logic  [WIDTH-'d1:0]     wr_data      ,
    input logic  [DEPTH-'d1:0]     addr         ,
    output logic  [WIDTH-'d1:0]     rd_data

);
logic [WIDTH-1:0]mem[DEPTH-1:0];

    //initial begin
    //    for (int i = 0; i < DEPTH; i++) begin
    //        mem[i] = '0;
    //    end
    //end

     always_ff @(negedge clk) begin
        if (en && wr) begin
            mem[addr] <= wr_data;
        end
    end

always_comb begin
    rd_data = {8{((!wr) && en)}} & mem[addr];

end

endmodule
```

```
module monitor(input [7:0]dutRdata ,arb_if.monitor_ IO);


wire [7:0] ref_rd_data;
ref_1#(
    .WIDTH        ( 8 ),
    .DEPTH        ( 32 )
)u_ref_1(
    . clk       (  IO.clk         ),
    . en        (  IO.en          ),
    . wr        (  IO.wr          ),
    .  wr_data  (   IO.wr_data    ),
    .  addr     (   IO.addr       ),
    .  rd_data  (   ref_rd_data   )
);



    always_ff @( posedge IO.clk ) begin : diiff
        $monitor ("en=%d wr=%d wr_data=%d addr=%d  ",IO.en , IO.wr ,IO.wr_data ,IO.addr );


            if(ref_rd_data != dutRdata)begin
                $display("ref_rd_data = %d , dut_rd_data = %d",ref_rd_data ,dutRdata);
                assert (0) ;
            end
            else begin
                $display("Congratulations test PASS!");
                $display("ref_rd_data = %d , dut_rd_data = %d",ref_rd_data ,dutRdata);
                $display("Congratulations test PASS!");
            end


    end
```

在 mointor 中监视读写行为，并 dut 和 ref 读数据作对比并打印

```
module tb_top();
    arb_if tb();
    dut dut_top(.arbif(tb.DUT));
    monitor monitor_top(.dutRdata(tb.rd_data), .IO(tb.monitor_));

    task Data_from( output logic en, output logic wr, output logic [31:0] addr, output logic
        en = $urandom%2;
        wr = $urandom%2;
        addr = $urandom_range(0, 31);
        wr_data = $urandom_range(0, 63);
    endtask

    initial begin
        tb.clk = 0;
        forever #5 tb.clk = ~tb.clk;
    end

    initial begin
      forever begin
          #10 Data_from(tb.en, tb.wr, tb.addr, tb.wr_data);
      end
  end


    initial begin

        #1000 $finish;
    end
endmodule
```

顶层测试模块，对读写使能，写地址/数据进行随机赋值，对 dut 和 ref 采用同样的激励进行 diff



最后在 vivado 中打印测试结果

六、 实验总结

第一次使用 sv 进行设计与验证，使用端口时候比 verilog 简洁太多，验证时利用搭建的平台进行系统验证，学习到了验证方法学的知识。