

# 集成电路设计仿真与验证技术 实验报告

题    目：           sram 与 cpu 接口仿真验证          

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          

姓名学号：           赵祖康（05219038）          

指导教师：           闫  宵

## 一、 实验目的

通过本次实验，验证 **sram** 与 **cpu** 接口的正确性

实验要求：

硬件平台：vivado 仿真器

## 二、 实验原理

通过实现 **cpu** 外设外接 **sram** 存储器，实现 **add** ， **load** ，  
**store** 三种自定义指令并验证相关接口时序的正确性

## 三、 实验过程

实现 **cpu** 的解码模块，译码出寄存器读端口 **sram** 读写使能等相关信号

```
Desktop > sramCpu > sramCpu.src > sources_1 > imports > cpusramIO > sv > decoder.sv
1  `include "interface.sv"
2
3  module decoder (arb_if.decoder io);
4      logic [1:0] opcode2 ;
5      assign opcode2 = io.ins[31:30];
6
7      assign io.add    = ~opcode2[0] & ~opcode2[1];
8      assign io.load   = opcode2[0] & ~opcode2[1];
9      assign io.store  = ~opcode2[0] & opcode2[1];
10
11     logic Imodel ;
12     assign Imodel = io.ins[29];
13     logic [4:0]test;
14     assign test = io.ins[29:25];
15     assign io.rs1 = {5{io.add }} & io.ins[29:25] |
16                 {5{io.store}} & io.ins[20:16] & {5{~Imodel}};
17     assign io.rs2 = {5{io.add }} & io.ins[24:20];
18     assign io.rd  = ({5{io.add }} & io.ins[19:15]) |
19                 ({5{io.load }} & io.ins[20:16]) |
20                 ({5{io.store }} & 5'b00000);
21     assign io.AWsram = {32{io.store }} & io.ins[28:21];
22     assign io.ARsram = {32{io.load }} & {32{~Imodel}} & io.ins[28:21];
23
24     assign io.imodel = Imodel;
25     assign io.PrfWen = io.load | io.add;
26 endmodule
27
```

实现 **monior** 模块，在标准的处理器验证中，应该是当指令顺序提交退休后才进行 **monitor** 的监控以及 **check**，在此处采用便捷方案（利用计数器控制每条指令存在的生命周期），在当条指令存在的周期内进行检查

```

module monitor(arb_if.monitor io);

    reg [9:0] cnt =0;

    always_ff @(posedge io.clk) begin : counnter
        cnt <= cnt + 'b1;
    end

    always_ff @(posedge io.clk) begin : MonitorCheck
        if(cnt>='d6 && cnt<='d12)begin
            if(io.rs1=='d1 && io.rs2 == 'd2 && io.rd == 'd3)begin
                $display("add pass");
            end
            else assert (0);
        end

        if(cnt>'d12 && cnt<='d15)begin
            if(io.sram_raddr == 'd15)begin
                $display("loadR pass");
            end
            else assert (0);
        end

        if(cnt>'d17 && cnt<='d20)begin
            if(io.rd == 'd5)begin
                $display("loadI pass");
            end
            else assert (0);
        end

        if(cnt>'d21 && cnt<='d26)begin
            if(io.sram_waddr == 'd10 && io.sram_wr && io.rs1 == 'd5)begin
                $display("storeR pass");
            end
            else assert (0);
        end

        if(cnt>'d27 && cnt<='d70)begin
            if(io.sram_waddr == 'd11 && io.sram_wr_data == 'd10 && io.sram_wr)begin
                $display("storeI pass");
            end
            else assert (0);
        end
    end
end
endmodule

```

CPU 顶层模块，主要例化了 monitor 和 prf 两个模块，以及一个加法器构成这个简单的自定义 cpu

```

4 module Cpu(
5     output [31:0] sramRaddr, sramWaddr,
6     output sramWen,
7     output [31:0] sramWdata,
8     input [31:0] sramRdata,
9     input clk,
10    input rstn,
11    input [31:0] ins
12 );
13
14 logic [31:0] WPrfData;
15 logic [31:0] rdata_1, rdata_2;
16 logic [31:0] SRAM_Rdata;
17 logic [31:0] SRAM_Wdata;
18
19 arb_if arb_if_inst(clk);
20
21 assign arb_if_inst.ins = ins;
22 assign arb_if_inst.rstn = rstn;
23 assign arb_if_inst.sramRdata = sramRdata;
24
25 decoder decoder_U(
26     .io(arb_if_inst)
27 );
28
29 regfile u_regfile(
30     .clk ( clk ),
31     .rstn ( rstn ),
32     .Wadd ( arb_if_inst.rd ),
33     .Wdata ( WPrfData ),
34     .isWreg ( arb_if_inst.PrfWen ),
35     .Radd1 ( arb_if_inst.rs1 ),
36     .Rdata1 ( rdata_1 ),
37     .Radd2 ( arb_if_inst.rs2 ),
38     .Rdata2 ( rdata_2 )
39 );
40
41 logic [31:0] aADDb ;
42 assign aADDb = rdata_1 + rdata_2;
43
44 assign WPrfData = ({32{arb_if_inst.imodel}} & {{24{1'b0}}, ins[28:21]} & {32{arb_if_inst.load}}) |
45                 ({32{~arb_if_inst.imodel}} & sramRdata & {32{arb_if_inst.load}}) |
46                 (aADDb & {32{arb_if_inst.add}});
47
48 assign SRAM_Wdata = {32{arb_if_inst.imodel}} & {32{arb_if_inst.store}} & {{24{1'b0}}, ins[20:13]};
49
50 assign sramRaddr = arb_if_inst.ARsram;
51 assign sramWaddr = arb_if_inst.AWsram;
52 assign sramWen = arb_if_inst.store;
53 assign sramWdata = SRAM_Wdata;
54
55
56 arb_if monitor_io(clk);
57 assign monitor_io.sram_wr_data = sramWdata;
58 assign monitor_io.sram_raddr = sramRaddr;
59 assign monitor_io.sram_waddr = sramWaddr;
60 assign monitor_io.sram_wr = sramWen;
61 assign monitor_io.rs1 = arb_if_inst.rs1;
62 assign monitor_io.rs2 = arb_if_inst.rs2;
63 assign monitor_io.rd = arb_if_inst.rd;
64
65
66 monitor monitor_U(
67     .io(monitor_io)
68 );
69 endmodule
70

```

顶层测试模块，通过 **task** 封装好五条自定义指令，并利用“人”充当 **ref** 进行正确性计算并将结果置于 **monitor** 中进行监测

```

21 );
22
23
24 task inst_add(output [31:0] sramRdata_t, output [31:0] ins_t);
25 begin
26     sramRdata_t = 32'b1;
27     ins_t       = 32'b00_00001_00010_00011_0000_0000_0000_000; // add x3 x2 x1
28 end
29 endtask
30
31 task inst_load_R(output [31:0] sramRdata_t, output [31:0] ins_t);
32 begin
33     sramRdata_t = 'b1000;
34     ins_t       = 32'b01_0_00001111_00100_0000_0000_0000_0000; // load x4 (Addr15) data='d8;
35 end
36 endtask
37
38 task inst_load_I(output [31:0] sramRdata_t, output [31:0] ins_t);
39 begin
40     sramRdata_t = 'b1000;
41     ins_t       = 32'b01_1_00001111_00101_0000_0000_0000_0000; // load x5 data='d15;
42 end
43 endtask
44
45 task inst_store_R(output [31:0] sramRdata_t, output [31:0] ins_t);
46 begin
47     ins_t       = 32'b10_0_00001010_00101_0000_0000_0000_0000; // store x5 (addr = 'd10);
48 end
49 endtask
50
51 task inst_store_I(output [31:0] sramRdata_t, output [31:0] ins_t);
52 begin
53     ins_t       = 32'b10_1_00001011_0000_1010_0_0000_0000_0000; // load 10 (addr = 'd11);
54 end
55 endtask
56
57 initial begin
58     clk = 0;
59     rstn = 0;
60     #10;
61     rstn = 1;
62     #50;
63     inst_add(sramRdata, ins);
64     #50;
65     inst_load_R(sramRdata, ins);
66     #50;
67     inst_load_I(sramRdata, ins);
68     #50;
69     inst_store_R(sramRdata, ins);
70     #50;
71     inst_store_I(sramRdata, ins);
72     #20;
73 end
74

```

最后的结果如图所示，全部 pass

