浙江大学

本科实验报告

课程名称：	计算机组成
姓　　名：	姜雨童
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
邮　　箱：	3220103450@zju.edu.cn
QQ　　号：	1369218489
电　　话：	19550103468
指导教师：	马德
报告日期：	2024 年 3 月 9 日

浙江大学实验报告

课程名称：计算机组成　　实验类型：综合

实验项目名称：Lab01 Warm up（学生版）

学生姓名：姜雨童　学号：33220103450　同组学生姓名：/

实验地点：紫金港东四 509 室　实验日期：2024 年 3 月 6 日

一、操作方法与实验步骤

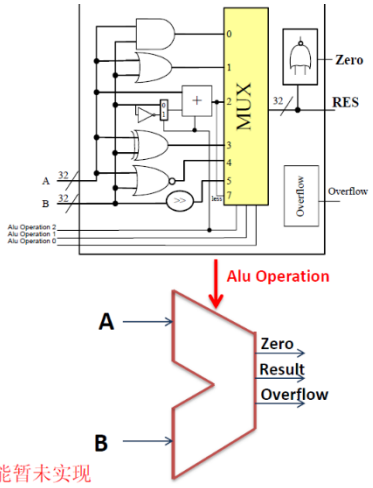
Lab01-1 ALU、Regfiles 设计

任务一：设计实现数据通路部件 ALU

这里采用根据原理图进行**功能性**描述设计的方式，设计要求如下：

- 实现5个基本运算
  - 整理逻辑实验的ALU
  - verilog输入并仿真
  - 拓展ALU的功能

ALU Control Lines	Function	note
000	And	兼容
001	Or	兼容
010	Add	兼容
110	Sub	兼容
111	Set on less than	
100	nor	扩展
101	srl	扩展
011	xor	扩展



注：overflow功能暂未实现

功能性描述代码：

```
module ALU(
    input [31:0] A,
    input [2:0] ALU_operation,
    input [31:0] B,
    output [31:0] res,
    output zero
    );
```

```
    reg [31:0] result;
    assign res = result;
    assign zero = ~(|result);

    always @(*) begin
        case(ALU_operation)
            3'b000: result = A & B;
            3'b001: result = A | B;
            3'b010: result = A + B;
            3'b011: result = A ^ B;
            3'b100: result = ~(A | B);
            3'b101: result = B >> 1;
            3'b110: result = A - B;
            3'b111: result = (A < B) ? 1 : 0;
            default;
        endcase
    end
endmodule
```

Testbench 代码：

```
module ALU_tb;
        reg [31:0] A, B;
                    reg[2:0] ALU_operation;
                    wire[31:0]   res;
                    wire zero;
   ALU ALU(
       .A(A),
       .B(B),
       .ALU_operation(ALU_operation),
       .res(res),
       .zero(zero)
   );
   initial begin
       A=32'hA5A5A5A5;
       B=32'h5A5A5A5A;
       ALU_operation =3'b111;
       #100;
       ALU_operation =3'b110;
       #100;
       ALU_operation =3'b101;
       #100;
       ALU_operation =3'b100;
       #100;
       ALU_operation =3'b011;
```

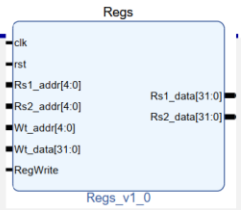
```
        #100;
        ALU_operation =3'b010;
        #100;
        ALU_operation =3'b001;
        #100;
        ALU_operation =3'b000;
        #100;
        A=32'h01234567;
        B=32'h76543210;
        ALU_operation =3'b111;
    end
endmodule
```

任务二：设计实现数据通路部件 Register Files

这里使用了给出的参考代码：

Regfile参考代码

```
Module regs( input          clk, rst, RegWrite,
             input  [4:0]   Rs1_addr, Rs2_addr, Wt_addr,
             input  [31:0]  Wt_data,
             output [31:0]  Rs1_data, Rs2_data
           );
reg [31:0] register [1:31];                  // r1 - r31
   integer i;

   assign rdata_A = (Rs1_addr== 0) ? 0 : register[Rs1_addr];          // read
   assign rdata_B = (Rs2_addr== 0) ? 0 : register[Rs2_addr];          // read

   always @(posedge clk or posedge rst)
     begin  if (rst==1)  for (i=1; i<32; i=i+1)  register[i] <= 0;    // reset
            else if ((Wt_addr != 0) && (RegWrite == 1))
                    register[Wt_addr] <= Wt_data;                     // write
     end

endmodule
```



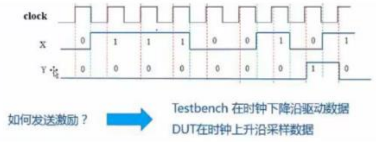
（需要注意的是，参考代码中 assign 语句后的 rdata_A 应改为 Rs1_data，rdata_B 同理改动）把给出的 testbench 文件添加进工程后（注意模块名称和自己命名的模块统一），进行仿真，仿真结果放在第二部分中。

（另：因为 testbench 已经给出，这里不做赘述。）

Lab01-2 有限状态机

任务：设计状态机解决序列检测问题（采用三段式）

- **设计要求**：用状态机设计序列检测器（1110010）
- **设计功能**：设计一个序列检测器，检测的序列为“1110010”；当输入信号X依次为“1110010”时，输出信号Y输出一个高电平，否则输出信号Y为低电平。
- **时序图**：序列检测器是一种同步时序电路，它用于搜索，检测输入的二进制代码串中是否出现指定的代码序列，1110010序列检测原理图如下：



这里使用了给出的参考代码：

```
module seq(
    input clk,
    input reset,
    input in,
    output out
    );

    parameter [2:0]  S0 = 3'b000,
                     S1 = 3'b001,
                     S2 = 3'b010,
                     S3 = 3'b011,
                     S4 = 3'b100,
                     S5 = 3'b101,
                     S6 = 3'b110,
                     S7 = 3'b111;
   reg [2:0] curr_state;
   reg [2:0] next_state;

   always@(posedge clk or negedge reset)
        begin
            if(!reset)
                curr_state <= S0;
            else
                curr_state <= next_state;
        end

    always @(curr_state or in)
        begin
            case(curr_state)
                S0:begin
                    if(in == 0) next_state = S0;
                    else        next_state = S1;
                end
                S1:begin
                    if(in == 0) next_state = S0;
                    else        next_state = S2;
                end
                S2:begin
                    if(in == 0) next_state = S0;
                    else        next_state = S3;
                end
                S3:begin
                    if(in == 0) next_state = S4;
```

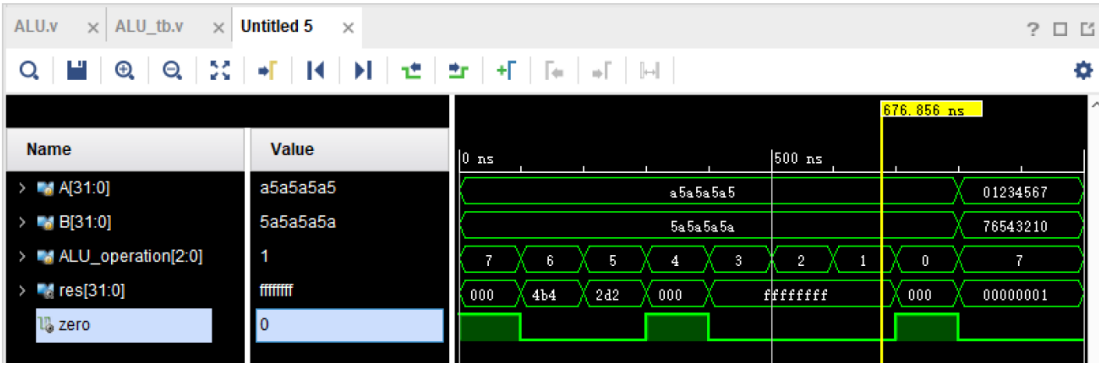
```
                else        next_state = S3;
            end
            S4:begin
                if(in == 0) next_state = S5;
                else        next_state = S1;
            end
            S5:begin
                if(in == 0) next_state = S0;
                else        next_state = S6;
            end
            S6:begin
                if(in == 0) next_state = S7;
                else        next_state = S2;
            end
            S7:begin
                if(in == 0) next_state = S0;
                else        next_state = S1;
            end
            default:        next_state = S0;
        endcase
    end
    assign out = (curr_state == S7) ? 1 : 0;

endmodule
```

把给出的 testbench 文件（不做赘述）添加进工程后，进行仿真。

二、实验结果与分析

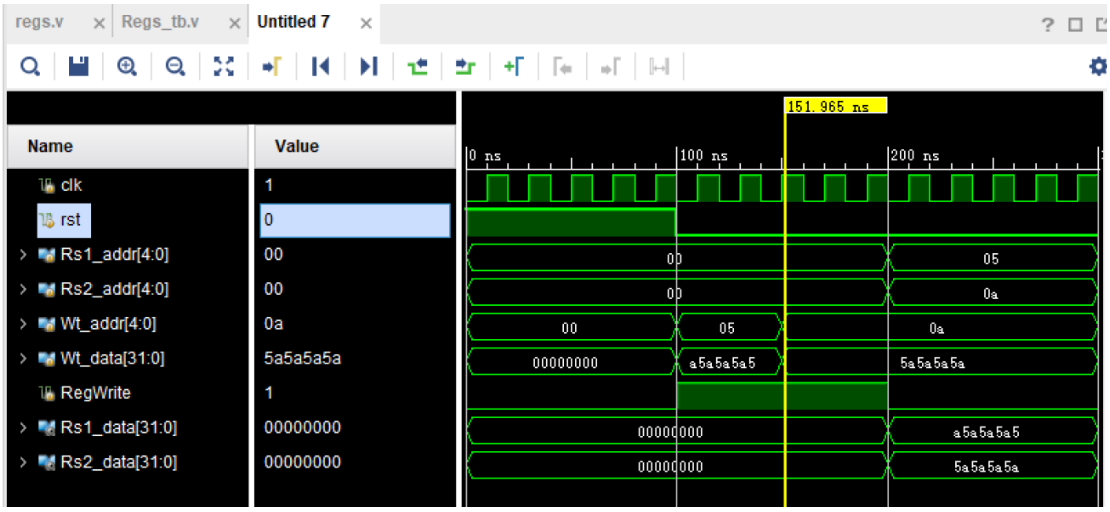
Lab01-1

任务一：ALU



根据设计要求，ALU_operation 为 0 至 7 时，分别进行 AND，OR，ADD，SUB，Set on less than，NOR，SRI，XOR 操作，分析后发现仿真结果完全符合预期。

任务二：Regfiles



在 100ns~150ns 阶段，Wt_data 的数据（xa5a5a5a5）被写入地址为 Wt_addr（x05）的寄存器中，而在 150ns~200ns 阶段，将新数据（x5a5a5a5a）写入新地址（x0a）。
200ns~300ns 阶段，Rs1_addr 和 Rs2_addr 均不为零，故读出存在该地址中的数据，表现在 Rs1_data 和 Rs2_data 中。
经判断，仿真结果完全符合预期。

Lab01-2 有限状态机

不难看出，输入出现“1110010”后，输出了一次 1。仿真结果符合预期。

三、讨论、心得

第一个实验的难度不是很大，而且很多代码都直接给出了，testbench 也可以直接添加进工程中省去了很多手动敲代码的时间，因此并没有碰到特别困难的地方，整体进度也非常快。
稍微需要考虑一下的是 ALU 中输出 zero 的部分需要对结果的每一位取或，这个操作我的印象不深，因此上网搜索了一下：

```
assign zero = ~(|result)
```

此外，vivado 中激励文件的编写方式和 ise 中的略有不同，需要稍微考虑一下。

四、个人生活照片