

**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： /

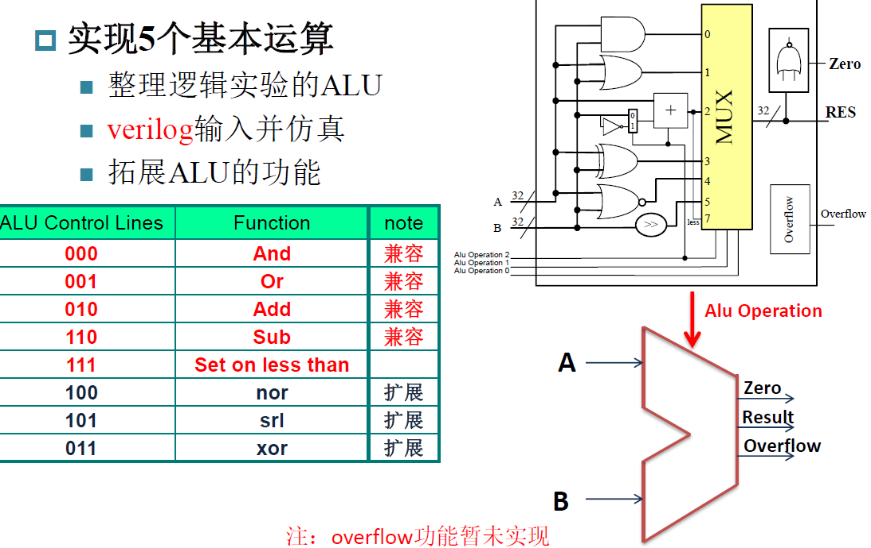
实验地点： 紫金港东四509室 实验日期： 2024 年 3 月 6 日

# 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：



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;

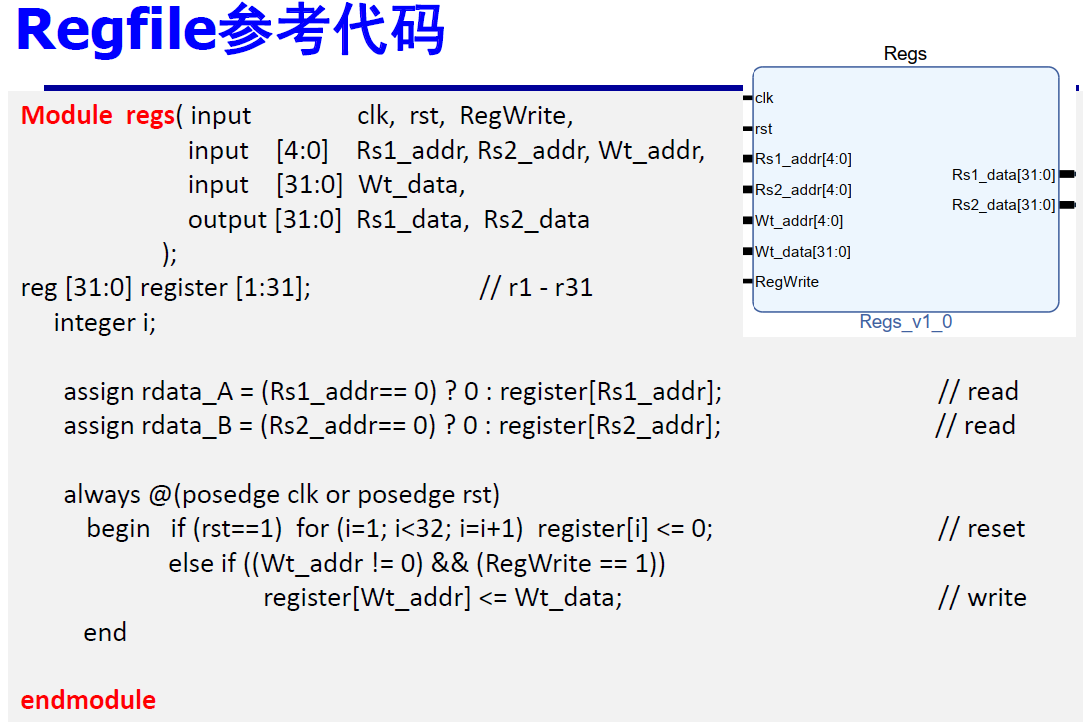
ALU\_operation =3'b111;

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：



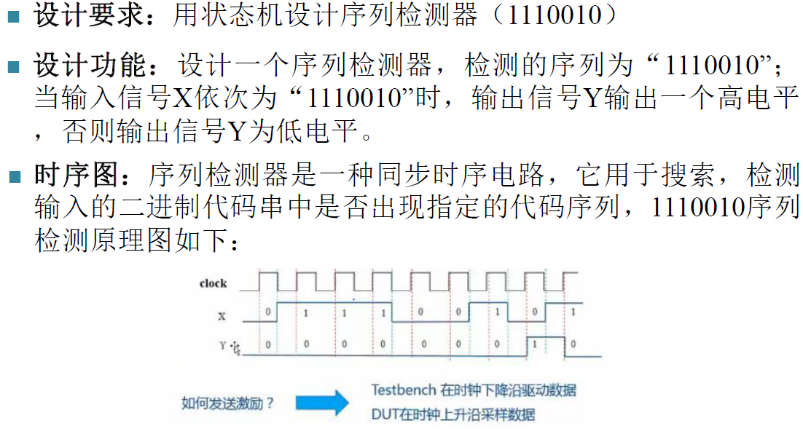
（需要注意的是，参考代码中assign语句后的rdata\_A应改为Rs1\_data，rdata\_B同理改动）

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）



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;

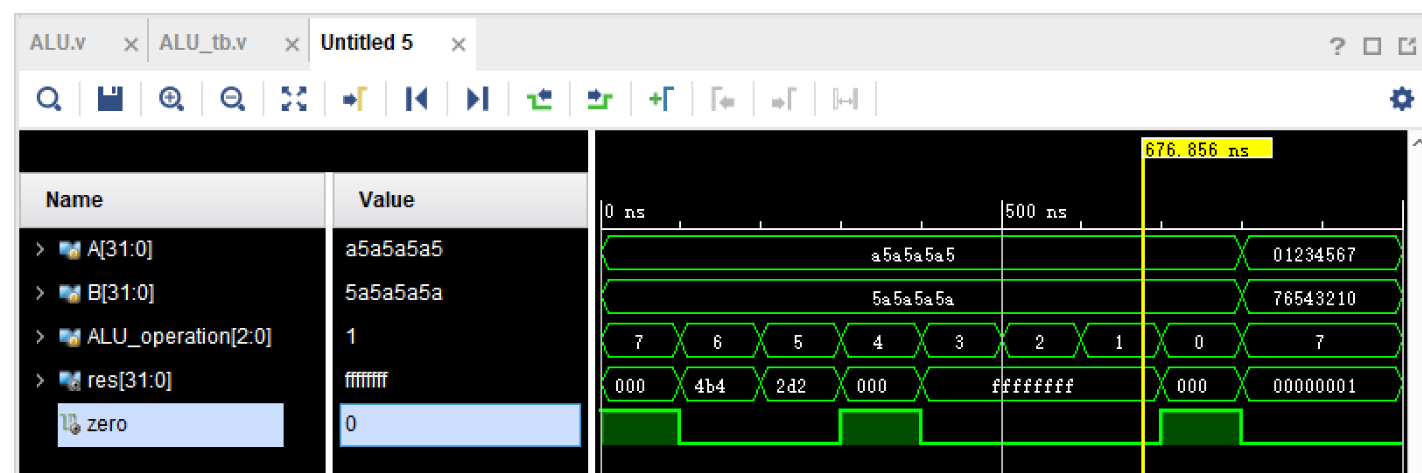
endmodule

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

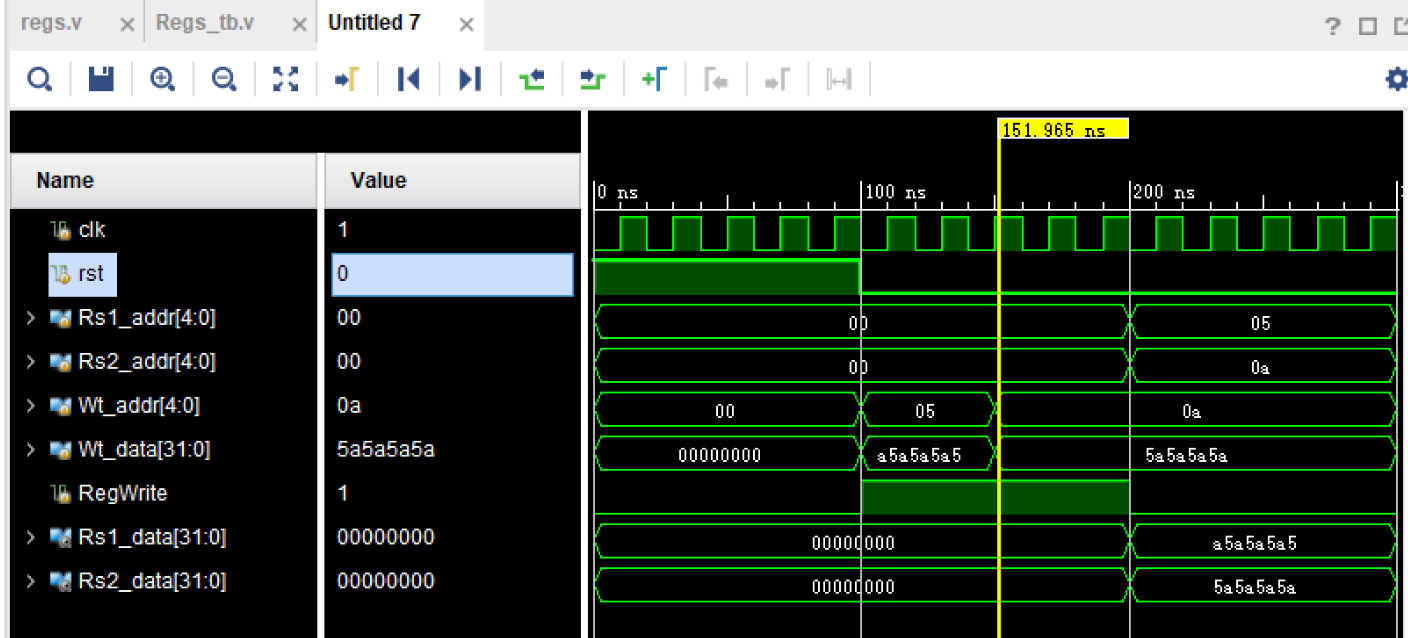
Lab01-1

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**

