浙江大学

计算机组成实验报告

实验名称：	ALU、Regfiles 以及有限状态机设计
姓　　名：	蒋奕
学　　号：	3210103803
专　　业：	计算机科学与技术
课程名称：	计算机组成
实验地点：	东 4-509
指导老师：	赵莎

2023　　年　3　月　9　日

实验 1—ALU、Regfiles 及有限状态机设计

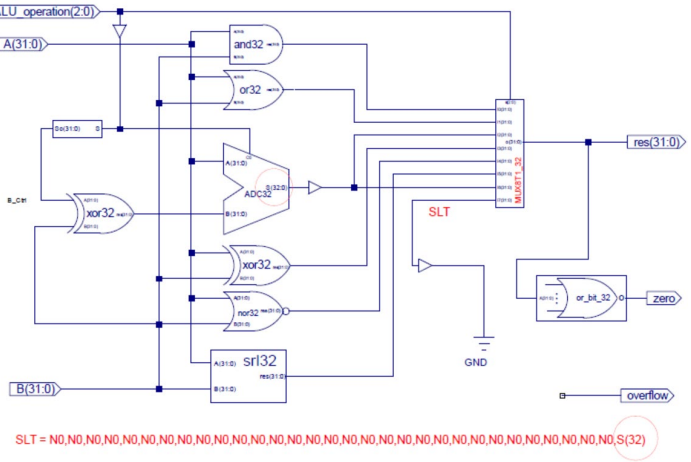
一、 实验目的和要求

1. 复习寄存器传输控制技术与有限状态机的基本概念
2. 掌握 CPU 的核心组成：数据通路与控制器，以及有限状态机的两种模型
3. 设计数据通路的功能部件，设计有限状态机解决实际问题
4. 进一步了解计算机系统的基本结构
5. 熟练掌握 IP 核的使用方法

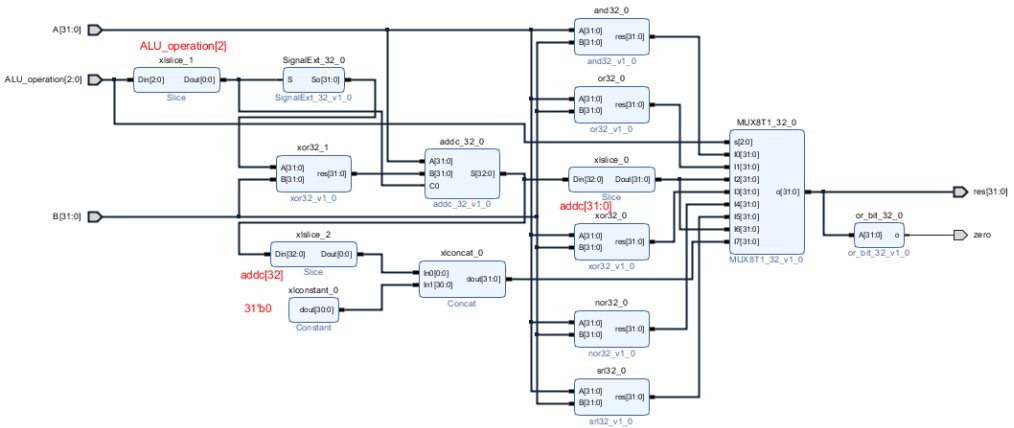
二、 实验内容和原理

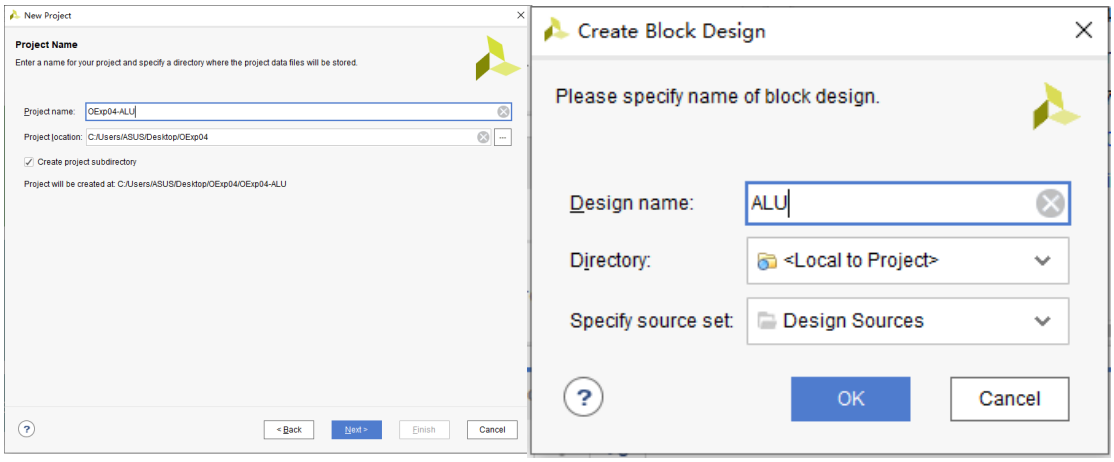
2.1 实验任务

1. 设计实现数据通路部件 ALU：采用原理图的设计方法

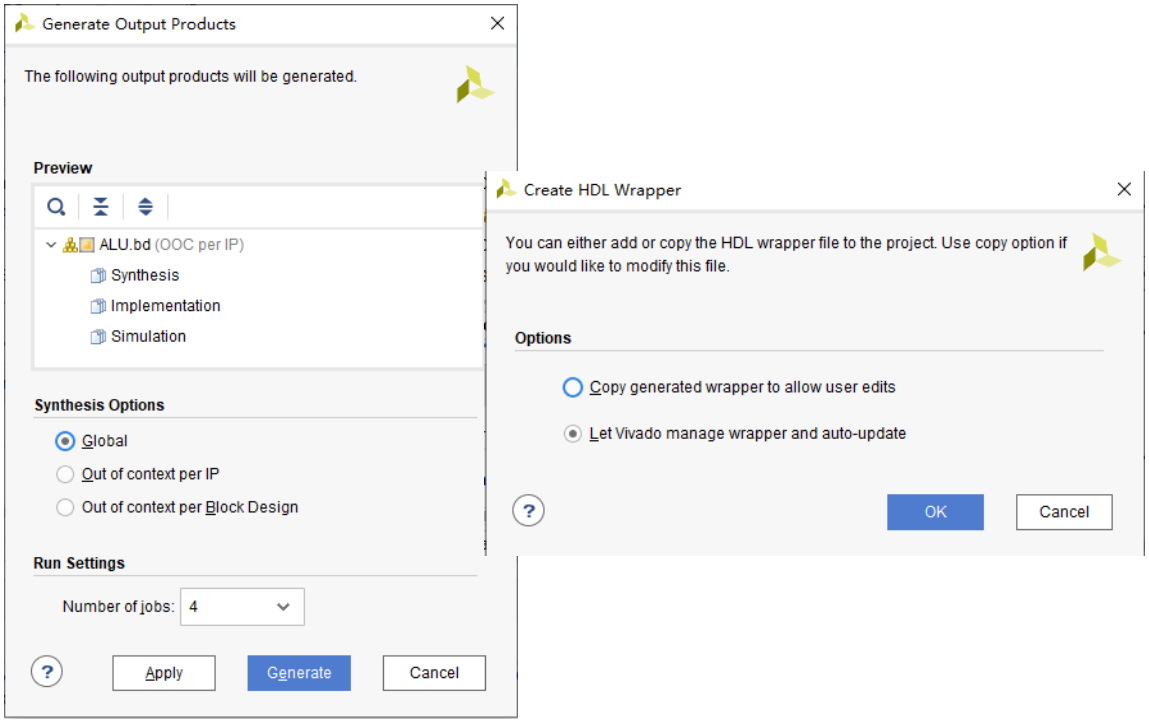


拷贝下列模块到 ALU 工程目录：（Exp0 提供）
and32、or32、ADC32、xor32、nor32、srl32、SignalExt_32、mux8to1_32、or_bit_32
添加模块路径到 ALU 工程目录：

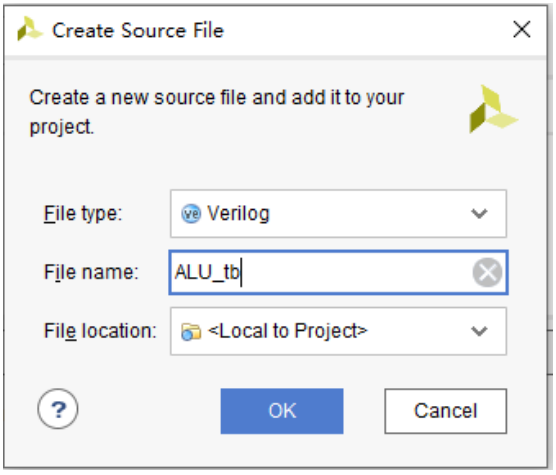




生成 IP 并且封装：



建立仿真文件仿真：



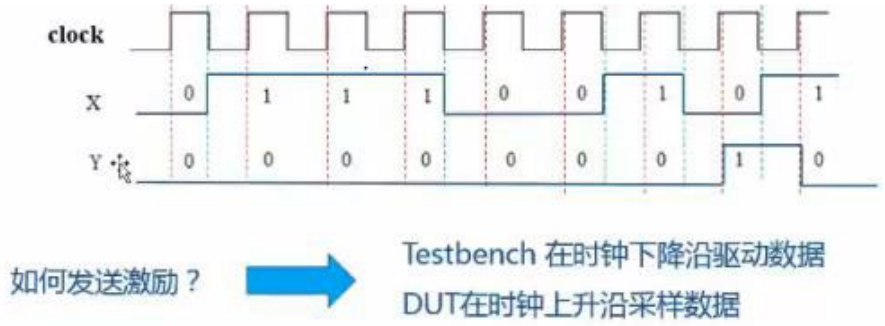
2.设计实现数据通路部件 Register Files：采用硬件描述语言的设计方法

实现 32 × 32bit 寄存器组
优化逻辑实验 Regs，行为描述并仿真结果
Regfile 封装关键点：
①端口警告原因是 clk 端口属性未知
点击 clk 进入端口编辑界面,Parameters 下添加 ASSOCIATED_BUSIF,然后在新建的 ASSOCIATED_BUSIF 这个参数后面的 value 列输入定义的时钟信号的名字，此处为 clk

②复位信号自动反向原因是系统默认是低电平而实验设计时高电平,需要进行属性约束
点击 rst 进入端口编辑界面,Parameters 下添加 POLARITY。然后在新建的 POLARITY 这个参数后面的 value 列输入属性 ACTIVE_HIGH。
注意：后续封装的含时钟和复位信号的 IP，建议均作此类属性限制以免设计时产生问题

3.设计有限状态机完成序列检测器并测试

状态机设计方法：
一段式描述（即状态跳转与输出信号都在同一个 always 块里面进行描述）
二段式描述（即将输出信号与状态跳转分开描述，便于设计代码管理）
三段式描述（即将输出信号与状态跳转分开描述，并且状态跳转用组合逻辑来控制）
设计要求：
用状态机设计序列检测器（1110010------序列检测器）
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：

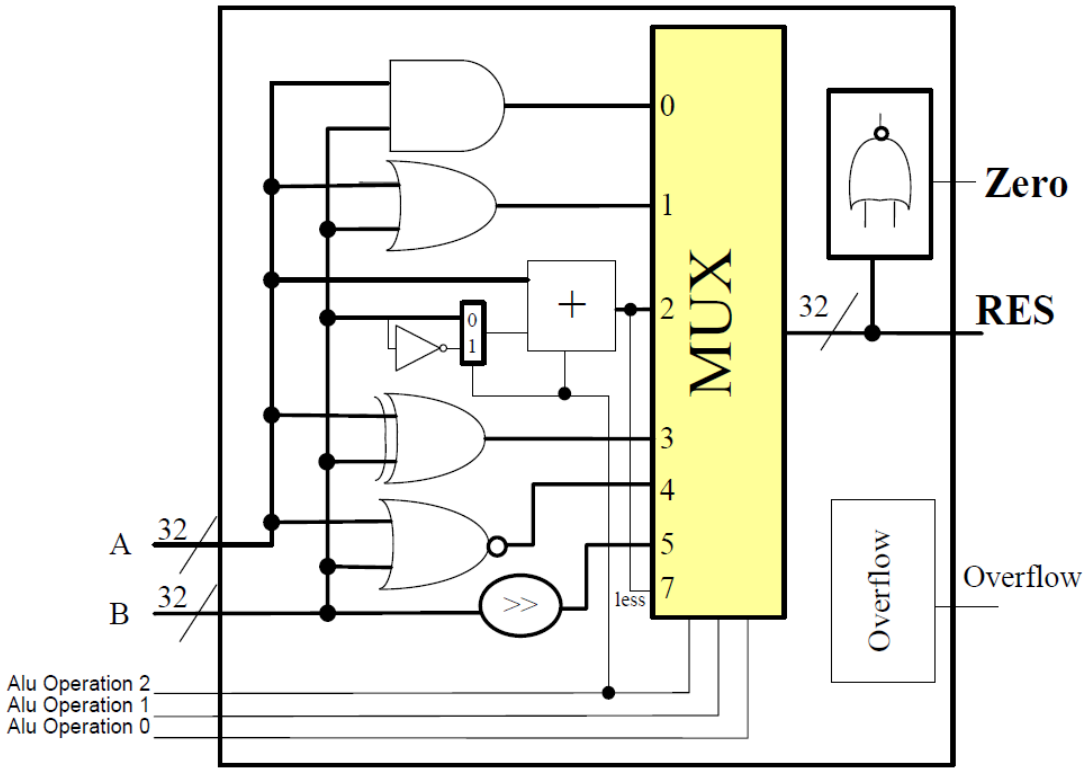


2.2 实验原理

1. ALU 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

原理图：



2.Reg file 端口

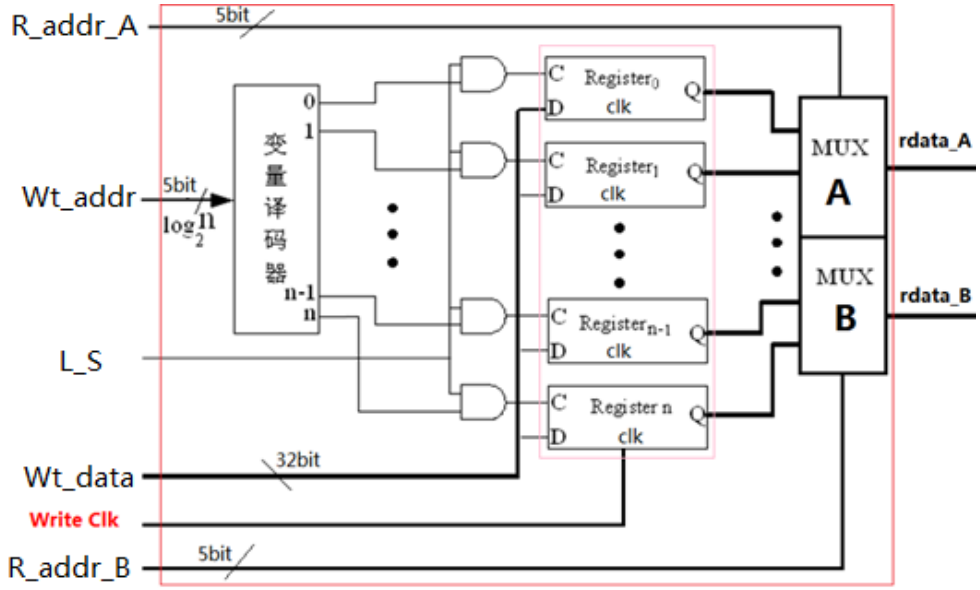
二个读端口：

Rs1_addr;Rs1_data

Rs2_addr;Rs2_data

一个写端口，带写信号
Wt_addr;Wt_data
RegWrite

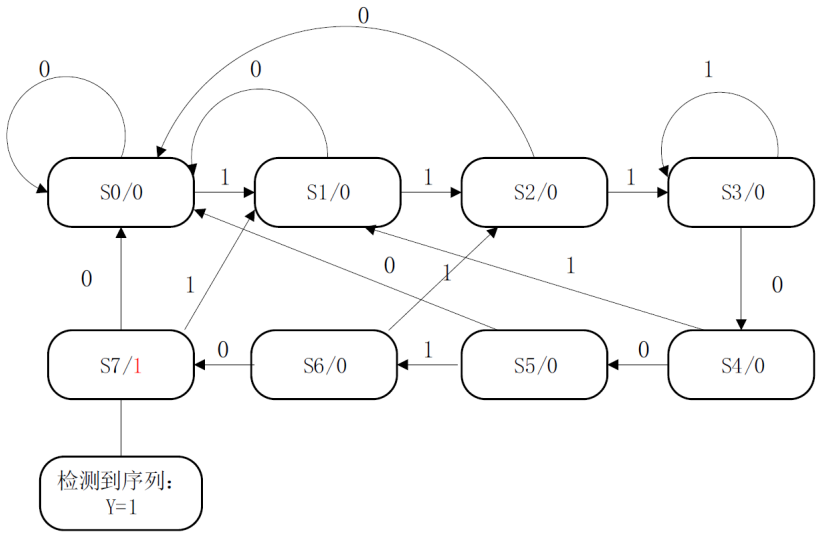
示意图：



3.状态机
系统架构和接口定义

接口	接口定义
clk	系统时钟
rst_n	系统复位
X	序列输入
Y	检测输出

状态转换图：

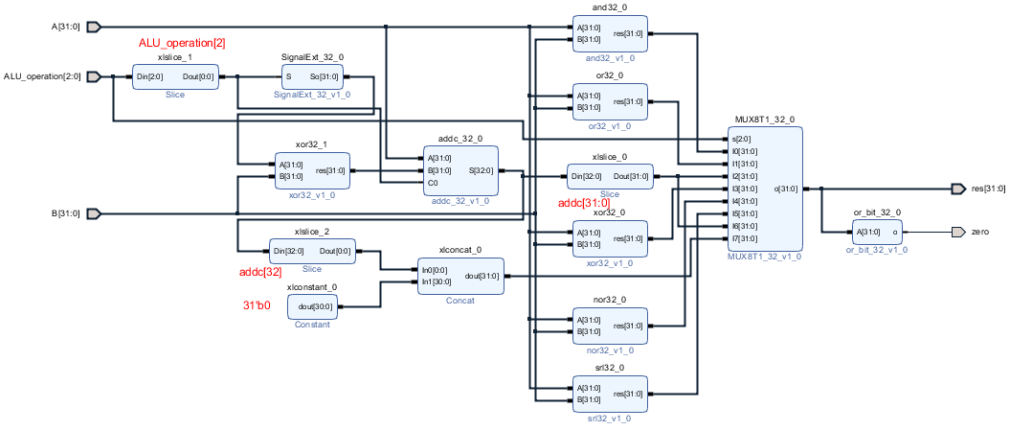


三、 实验设备和环境

1.计算机（Intel Core i5 以上，4GB 内存以上）系统
2. Sword2.0/Sword4.0 开发板
3. Xilinx VIVADO 2017.4 及以上开发工具

四、 实验实现方法、步骤与调试

1.ALU



2. Reg file 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  Rs1_data = (Rs1_addr== 0) ? 0 : register[Rs1_addr];//
read rdata_A
    assign  Rs2_data = (Rs2_addr== 0) ? 0 : register[Rs2_addr]; //
read rdata_B
    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
```

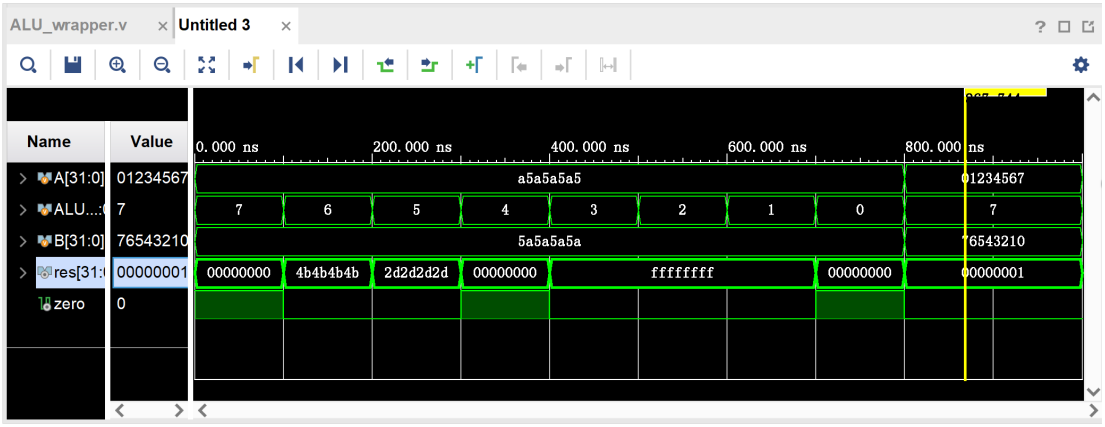
3.有限状态机代码

```
module seq_moore(
    input clk,input reset,input in,output out
);
 //define state
 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; //internal variable
always @(posedge clk or negedge reset) //first segment:state transfer
    begin
        if(!reset)// reset == 0
            curr_state <= S0;
        else
            curr_state <= next_state;
    end
 always @(curr_state or in) //second segment:transfer condition
    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
 //three segment: state output
 //moore type fsm
   assign out = (curr_state == S7) ? 1 : 0;
endmodule
```

五、 实验结果与分析

1. ALU

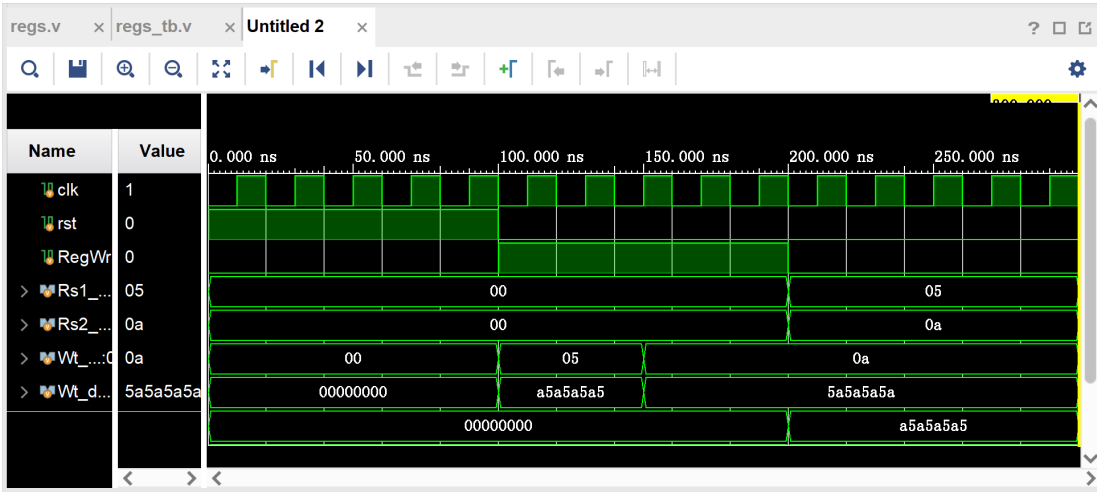


仿真代码：

```
module ALU_tb;
  reg [31:0] A;
  reg [2:0] ALU_operation;
  reg [31:0] B;
  wire [31:0] res;
  wire zero;
  ALU_wrapper ALU_wrapper_U(
        .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
```

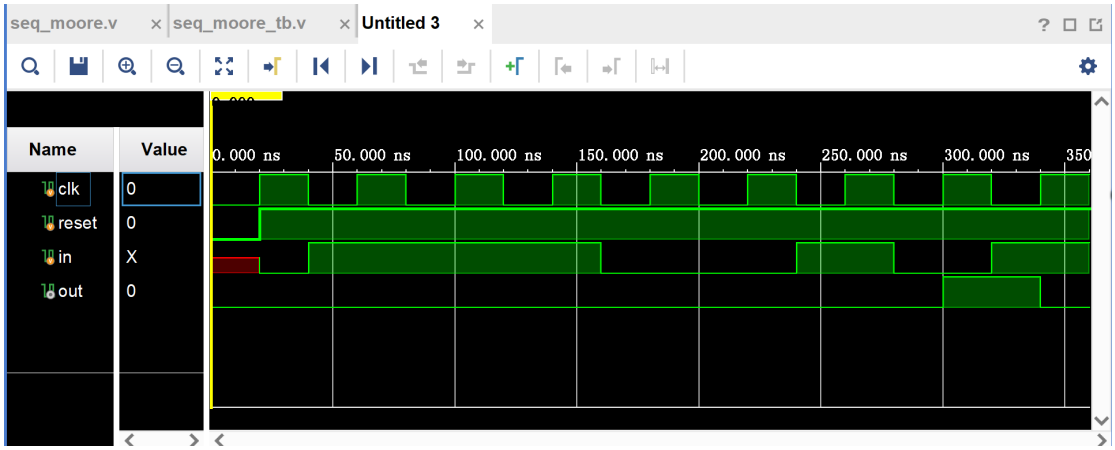
2.register file



仿真代码：

```
module regs_tb;
    reg clk , rst , RegWrite;
    reg [4:0] Rs1_addr, Rs2_addr,Wt_addr;
    reg [31:0] Wt_data;
    wire [31:0] Rs1_data, Rs2_data;
    regs regs_tb_U(
        .clk(clk),
        .rst(rst),
        .RegWrite(RegWrite),
        .Rs1_addr(Rs1_addr),
        .Rs2_addr(Rs2_addr),
        .Wt_addr(Wt_addr),
        .Wt_data(Wt_data),
        .Rs1_data(Rs1_data),
        .Rs2_data(Rs2_data)
    );
    always #10 clk = ~clk;
    initial begin
        clk = 0;
        rst = 1; RegWrite = 0;
        Rs1_addr = 0; Rs2_addr = 0; Wt_addr = 0;
        Wt_data = 0;
        #100;//initial
        rst = 0;RegWrite = 1;
        Wt_addr = 5'b00101;
        Wt_data = 32'ha5a5a5a5;
        #50;
        Wt_addr = 5'b01010;
        Wt_data = 32'h5a5a5a5a;
        #50;
        RegWrite = 0;
        Rs1_addr = 5'b00101;
        Rs2_addr = 5'b01010;
        #100;
        $finish;
    end
endmodule
```

3. 有限状态机



仿真代码：

```
module seq_moore_tb;
     reg clk,reset,in;
     wire out;
always #20 clk = ~clk;
initial begin
         clk = 0; reset = 0;#20
         reset = 1; in = 0; #20
         in = 1;#40
         in = 1; #40
         in = 1; #40
         in = 0;
         #40
         in = 0;
         #40
         in = 1;
         #40
         in = 0;
         #40
         in = 1;
         #40
         $finish;
         end
 seq_moore seq_moore_U(
     .clk(clk),
     .reset(reset),
     .in(in),
     .out(out)
    );
endmodule
```

六、 实验讨论、心得

这次实验让我充分体会到了 ALU 的原理和有限状态机的实现