
Computer Organization & Design 实验与课程设计

Lab01-1

ALU、Regfiles设计

Ma De (马德)

made@zju.edu.cn

2025

College of Computer Science, Zhejiang University

Course Outline

- 一、实验目的
- 二、实验环境
- 三、实验目标及任务

实验目的

1. 复习寄存器传输控制技术
2. 掌握CPU的核心组成：数据通路与控制器
3. 设计数据通路的功能部件
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

实验环境

□ 实验设备

1. 计算机（Intel Core i5以上， 4GB内存以上）系统
2. NEXYS A7开发板
3. Xilinx VIVADO2017.4及以上开发工具

□ 材料

无

实验目标及任务

- **目标：**熟悉SOC系统的原理，掌握IP核集成设计CPU的方法，了解数据通路结构并实现ALU和Register Files

- **任务一：**设计实现数据通路部件ALU
 - 采用硬件描述语言的设计方法
- **任务二：**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

■ 任务一：设计实现数据通路部件ALU

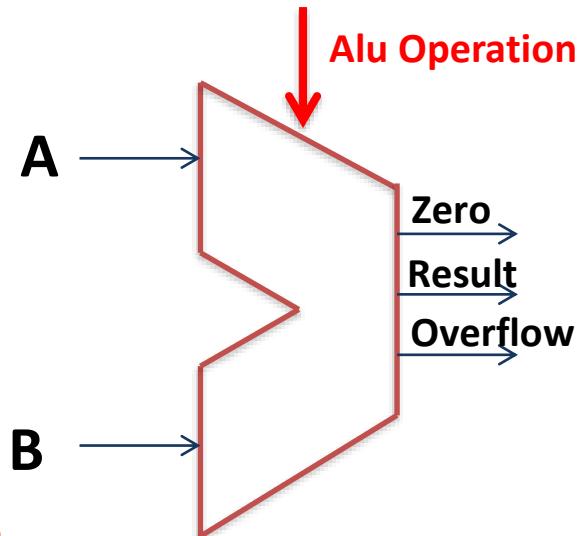
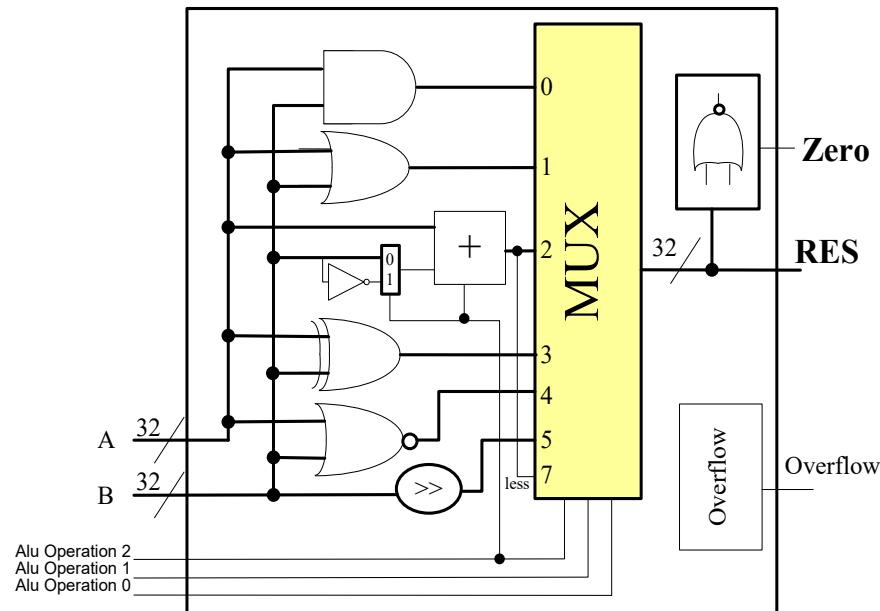
- 方法一：根据原理图进行结构化描述
- 方法二：根据原理图进行功能性描述

数据通路的功能部件之一：ALU

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

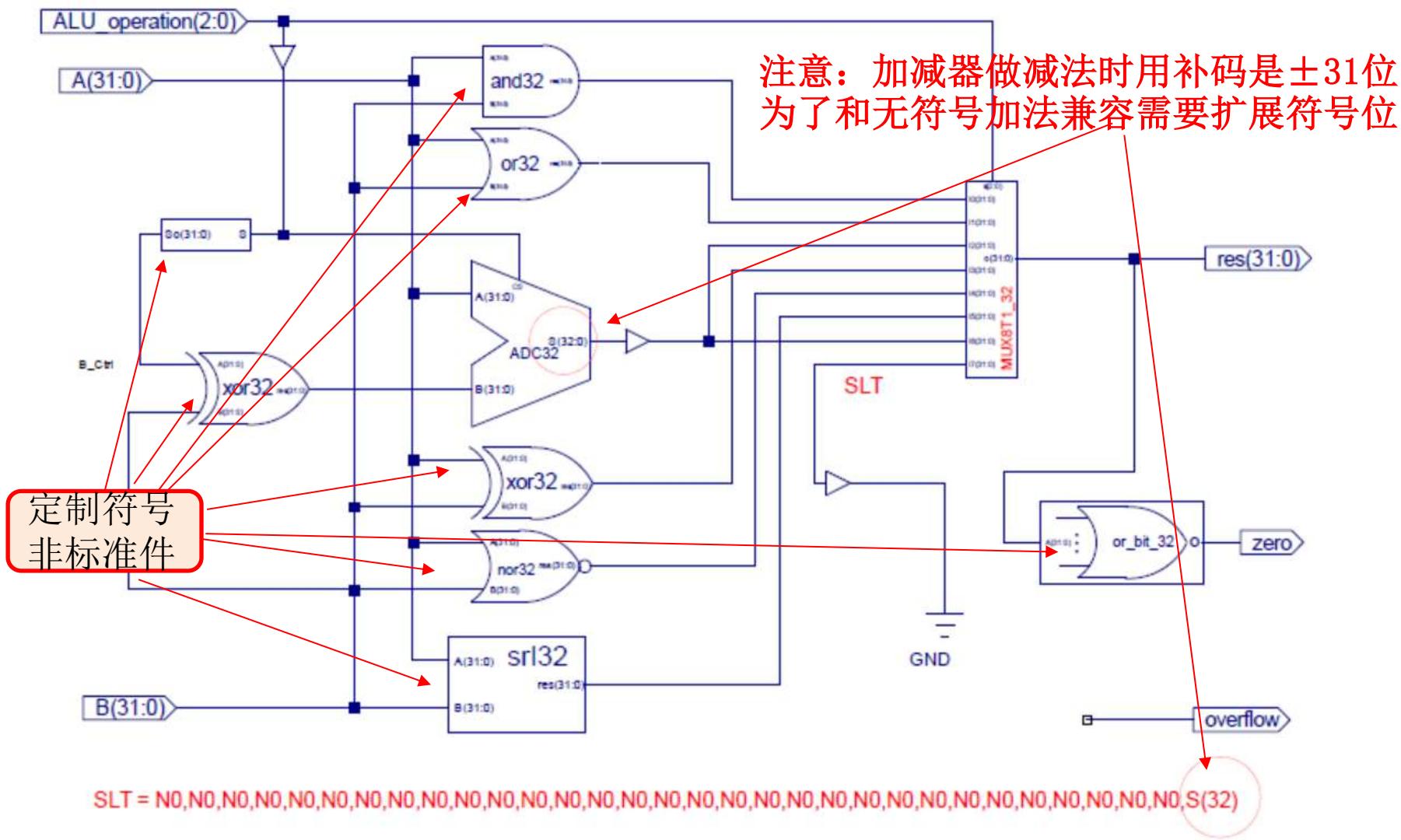


注：overflow功能暂未实现

根据原理图进行结构化描述设计

ALU

ALU逻辑原理图



结构化描述输入设计ALU

New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: OExp01-ALU

Project location: C:/Users/ASUS/Desktop

Create project subdirectory

Project will be created at: C:/Users/ASUS/Desktop/OExp01-ALU

Create Source File

Create a new source file and add it to your project.

File type: Verilog

File name: ALU

File location: <Local to Project>

?

< Back

Next >

Finish

Cancel

OK

Cancel

结构化描述输入设计ALU

拷贝下列模块到ALU工程目录：

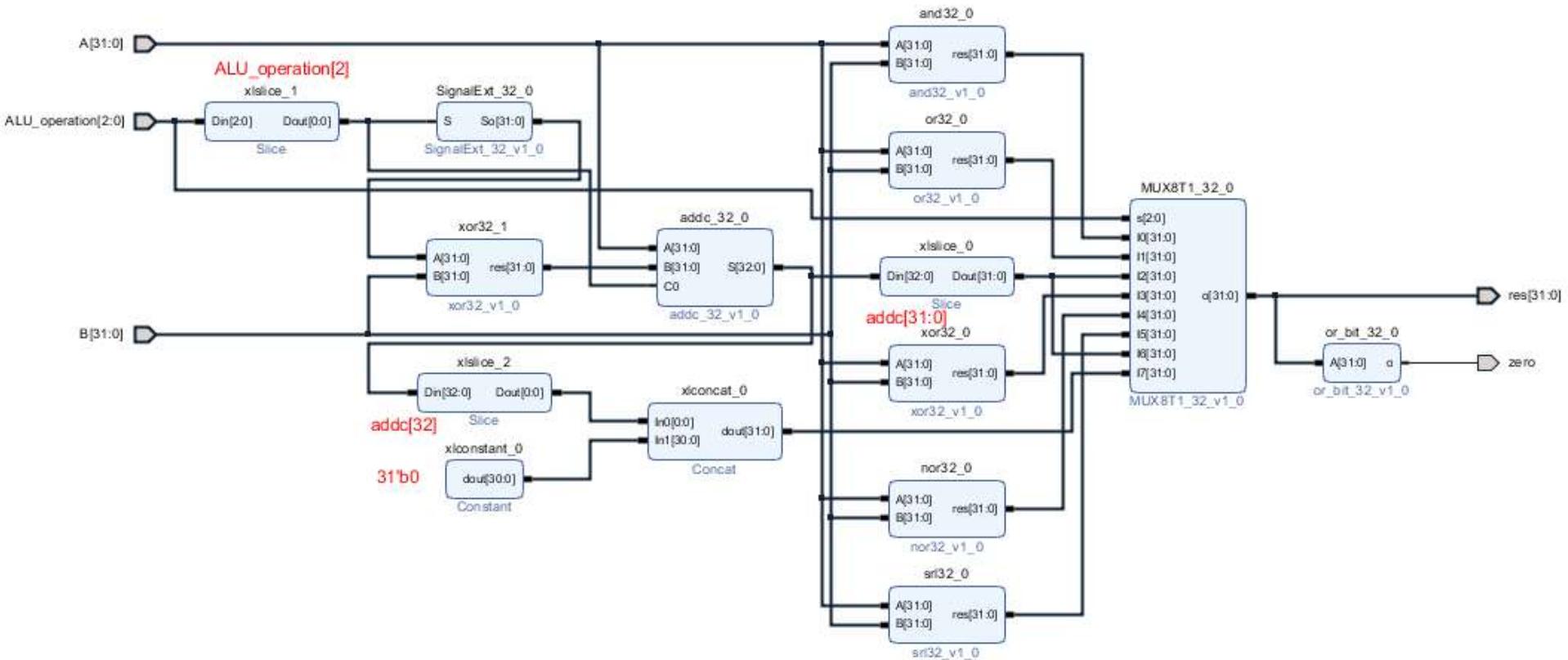
(Exp0设计)

**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：

ALU TOP逻辑原理图

顶层逻辑连接如图（详细可参照pdf文档），根据连接图可完成顶层文件的编写



ALU TOP文件编写

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

```
MUX8T1_32_0 MUX8T1_32_0  
    (.I0(and32_0_res),  
     .I1(or32_0_res),  
     .I2(addc_32_0_S[31:0]),  
     .I3(xor32_0_res),  
     .I4(nor32_0_res),  
     .I5(srl32_0_res),  
     .I6(addc_32_0_S[31:0]),  
     .I7({31'b0,addc_32_0_S[32]}),  
     .o(MUX8T1_32_0_o),  
     .s(ALU_operation));
```

```
.....  
.....  
.....  
xor32_0 xor32_1  
    (.A(SignalExt_32_0_So),  
     .B(B),  
     .res(xor32_1_res));  
endmodule
```

ALU TOP文件编写

PROJECT MANAGER - OExp01-ALU

Sources

Design Sources (1)

- ALU (ALU.v) (10)
- MUX8T1_32_0 : MUX8T1_32_0 (MUX8T1_32_0.xci)
- SignalExt_32_0 : SignalExt_32_0 (SignalExt_32_0.xci)
- addc_32_0 : addc_32_0 (addc_32_0.xci)
- and32_0 : and32_0 (and32_0.xci)
- nor32_0 : nor32_0 (nor32_0.xci)
- or32_0 : or32_0 (or32_0.xci)
- or_bit_32_0 : or_bit_32_0 (or_bit_32_0.xci)
- srl32_0 : srl32_0 (srl32_0.xci)
- xor32_0 : xor32_0 (xor32_0.xci)
- xor32_1 : xor32_0 (xor32_0.xci)

Constraints

Simulation Sources (1)

- sim_1 (1)
- ALU_tb (ALU_tb.v) (1)

Bus Net Properties

Project Summary x ALU.v x ALU_tb.v x

C:/Users/ASUS/Desktop/OExp01-ALU/OExp01-ALU.srsc/sources_1/new/ALU.v

```
67 .B(B),  
68 .res(nor32_0_res));  
69 or32_0 or32_0  
70 (.A(A),  
71 .B(B),  
72 .res(or32_0_res));  
73 or_bit_32_0 or_bit_32_0  
74 (.A(MUX8T1_32_0_o),  
75 .o(zero));  
76 sr132_0 sr132_0  
77 (.A(A),  
78 .B(B),  
79 .res(sr132_0_res));  
80 xor32_0 xor32_0  
81 (.A(A),  
82 .B(B),  
83 .res(xor32_0_res));
```

调用的子模块

设计要点----子模块调用

- 通过顶层文件调用子模块的方式有两种：
- 一种是子模块以IP形式封装的，具体操作步骤参见Lab0调用封装模块方法
- 一种是子模块以.v形式添加的，具体操作方式参见Lab0添加源文件的方法

ALU (ALU.v) (10)

```
> MUX8T1_32_0 : MUX8T1_32_0 (MUX8T1_32_0.xci)
> SignalExt_32_0 : SignalExt_32_0 (SignalExt_32_0.xci)
> addc_32_0 : addc_32_0 (addc_32_0.xci)
> and32_0 : and32_0 (and32_0.xci)
> nor32_0 : nor32_0 (nor32_0.xci)
> or32_0 : or32_0 (or32_0.xci)
> or_bit_32_0 : or_bit_32_0 (or_bit_32_0.xci)
> srl32_0 : srl32_0 (srl32_0.xci)
> xor32_0 : xor32_0 (xor32_0.xci)
> xor32_1 : xor32_0 (xor32_0.xci)
```

子模块以IP
形式被调用

ALU (ALU.v) (10)

```
MUX8T1_32_0 : MUX8T1_32_0 (MUX8T1_32_0.v)
SignalExt_32_0 : SignalExt_32_0 (SignalExt_32_0.v)
addc_32_0 : addc_32_0 (addc_32_0.v)
and32_0 : and32_0 (and32_0.v)
nor32_0 : nor32_0 (nor32_0.v)
or32_0 : or32_0 (or32_0.v)
or_bit_32_0 : or_bit_32_0 (or_bit_32_0.v)
srl32_0 : srl32_0 (srl32_0.v)
xor32_0 : xor32_0 (xor32_0.v)
xor32_1 : xor32_0 (xor32_0.v)
```

子模块以.v
形式被调用

根据原理图进行功能性描述设计

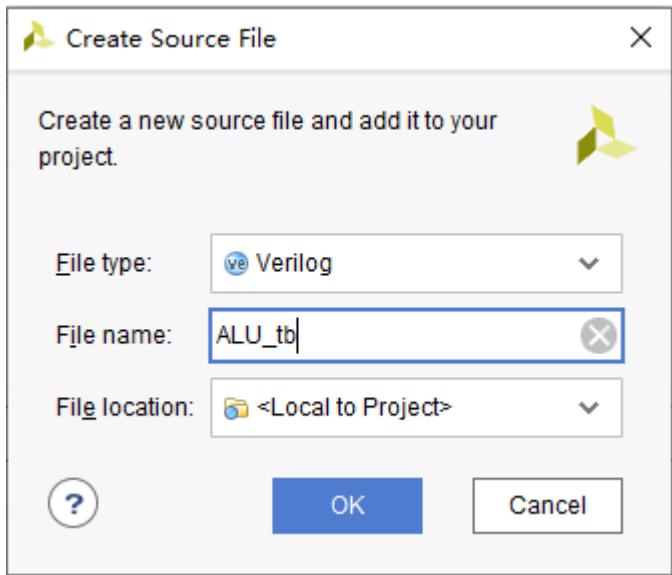
ALU

ALU 功能性描述

```
module ALU(  
    input [31:0] A,  
    input [2:0]  ALU_operation,  
    input [31:0] B,  
    output [31:0] res,  
    output      zero  
);  
.....  
always @ (*)  
    case (ALU_operation)  
        3'b000: res=A&B;  
.....  
.....  
.....  
endcase  
  
endmodule
```

采用逻辑表达式描述功能

ALU testbench文件



```
ALU ALU_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
```

ALU仿真波形图



仿真确保功能正确

■ 任务二：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

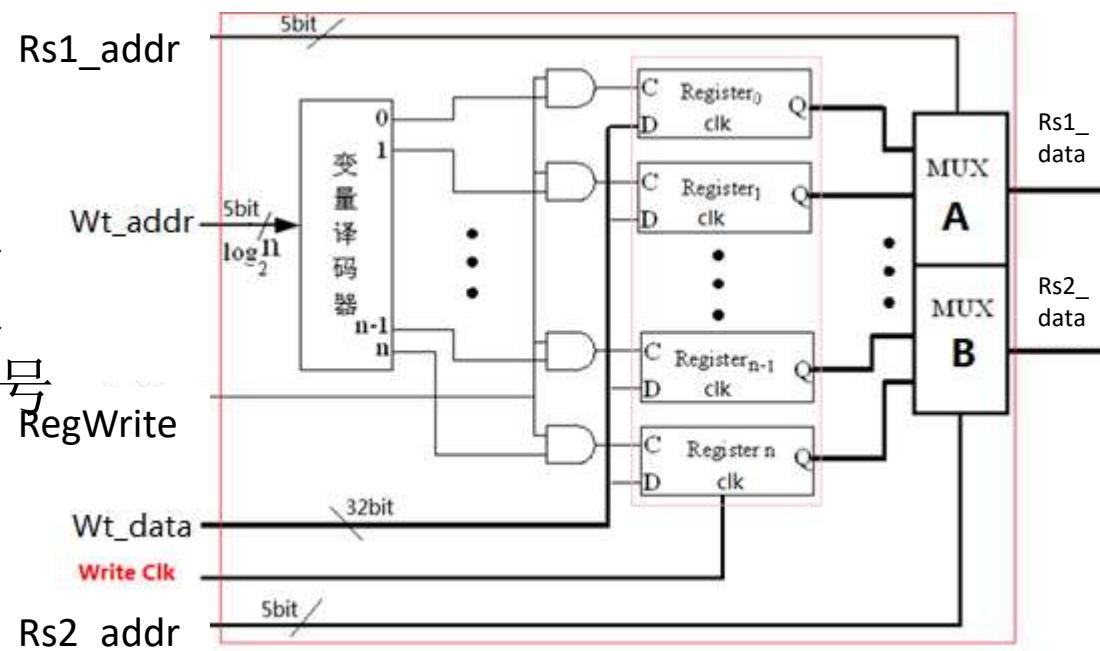
数字系统的功能部件之一： Register files

□ 实现 $32 \times 32\text{bit}$ 寄存器组

- 优化逻辑实验Regs
- 行为描述并仿真结果

□ 端口要求

- 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



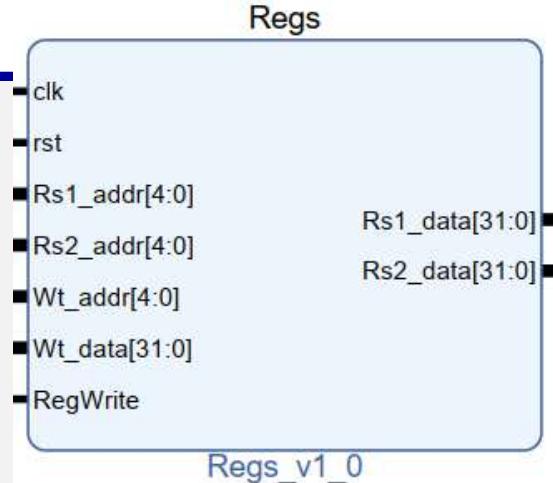
Regfile参考代码

```
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
```



regfile仿真结果

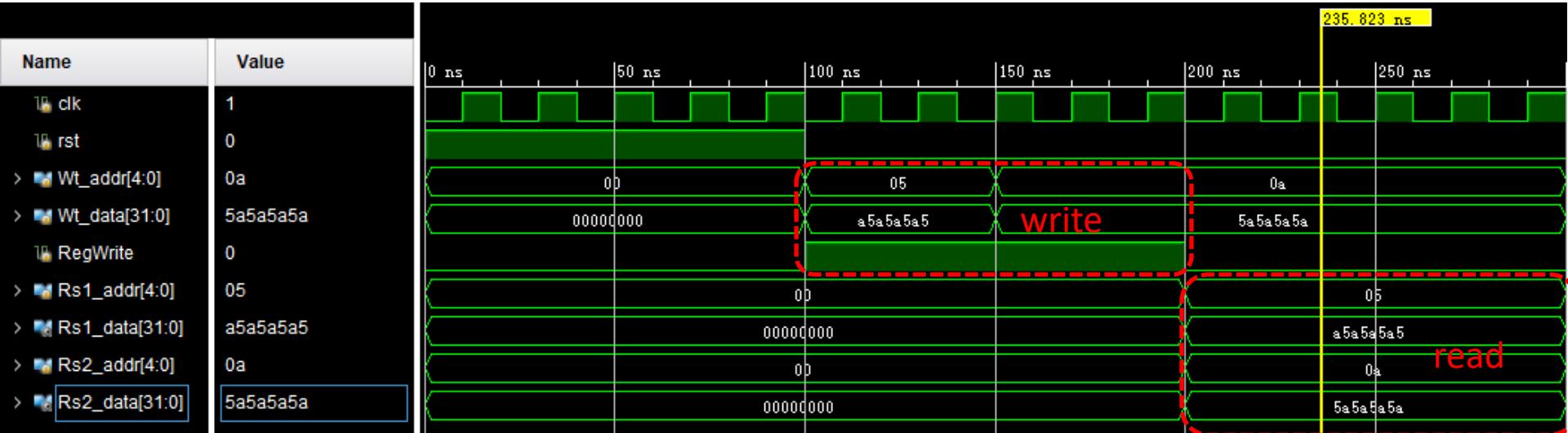
0ns-100ns regfile初始化复位，读写都为0；

100ns-150ns RegWrite=1;Wt_addr[4:0]=05;Wt_data[31:0]=a5a5a5a5;写地址05

150ns-200ns RegWrite=1;Wt_addr[4:0]=0a;Wt_data[31:0]=5a5a5a5a;写地址0a

200ns-300ns RegWrite=0;Rs1_addr[4:0]=05;Rs1_data[31:0]=a5a5a5a5;读地址05

200ns-300ns RegWrite=0;Rs2_addr[4:0]=0a;Rs2_data[31:0]=5a5a5a5a;读地址0a



仿真确保功能正确

思考题

□ 如何给ALU增加溢出功能

- 提示：分析运算结果的符号

□ 分析逻辑实验的Register Files设计

- 本实验你做了哪些优化？
- 逻辑实验的Register Files直接使用，你认为会存在哪些问题？

 END