

Lab01-1

ALU、Regfiles设计

Ma De (马德)

made@zju.edu.cn

2022

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：数据通路与控制单元
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. Sword2.0/Sword4.0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

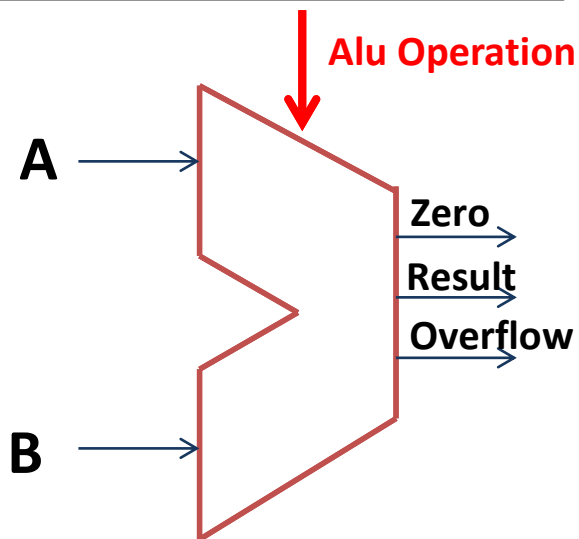
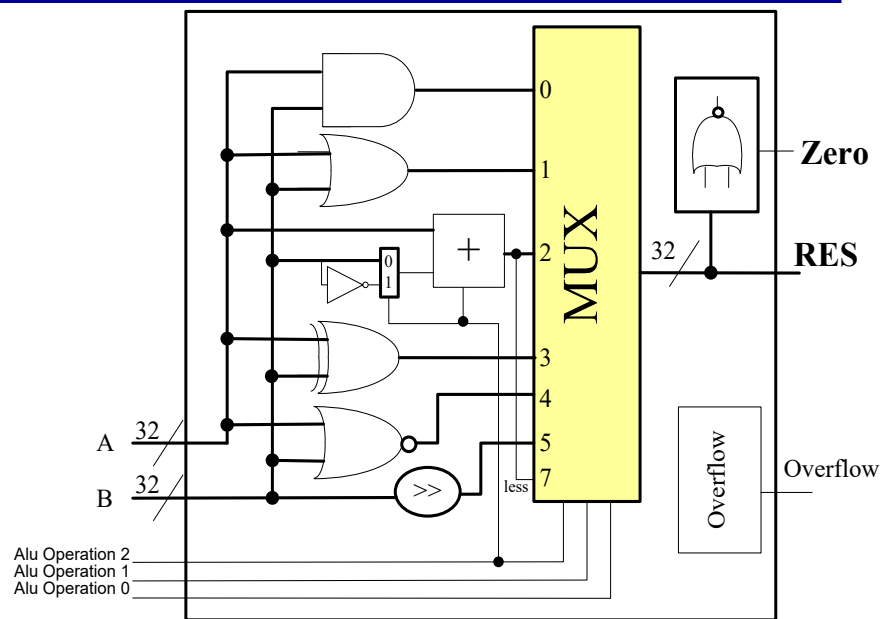
---方法二：根据原理图进行功能性描述

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

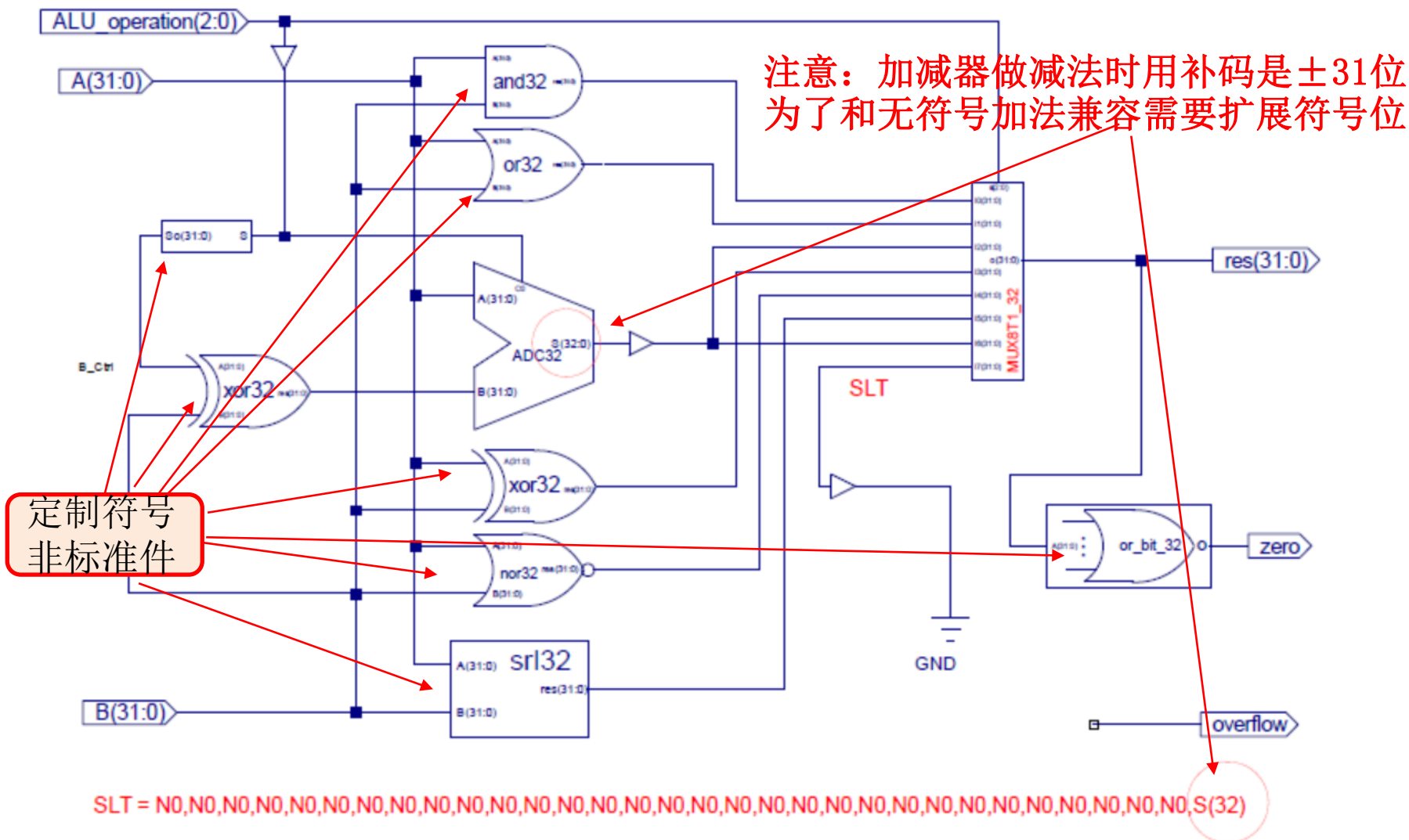


注：overflow功能暂未实现

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

?

< Back

Next >

Finish

Cancel

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>

?

OK

Cancel

结构化描述输入设计ALU

拷贝下列模块到ALU工程目录：

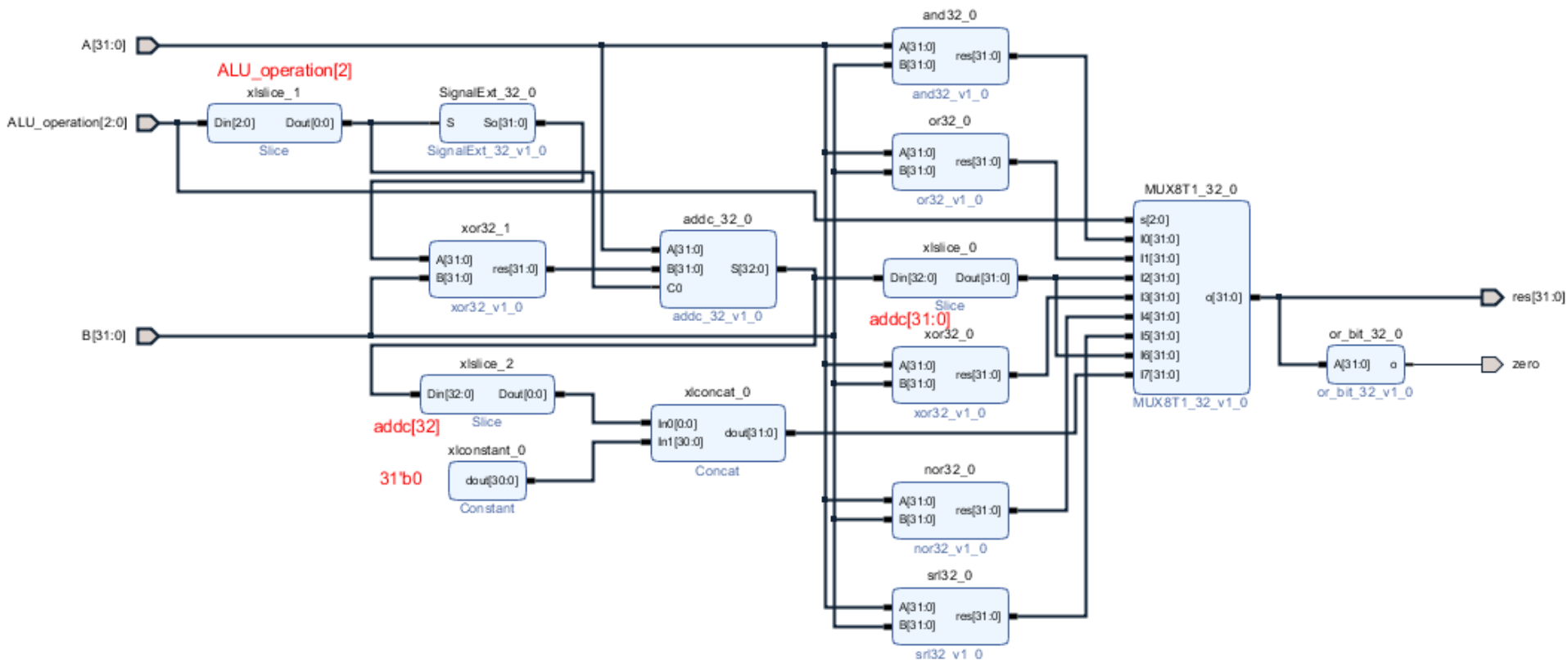
(Exp0设计)

**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)

Project Summary

ALU.v

ALU_tb.v

C:/Users/ASUS/Desktop/OExp01-ALU/OExp01-ALU.srscs/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  srl32_0 srl32_0
77      (.A(A),
78      .B(B),
79      .res(srl32_0_res));
80  xor32_0 xor32_0
81      (.A(A),
82      .B(B),
83      .res(xor32_0_res));
```

调用的
子模块

Hierarchy

IP Sources

Libraries

Compile Order

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;
```

```
.....  
.....  
.....
```

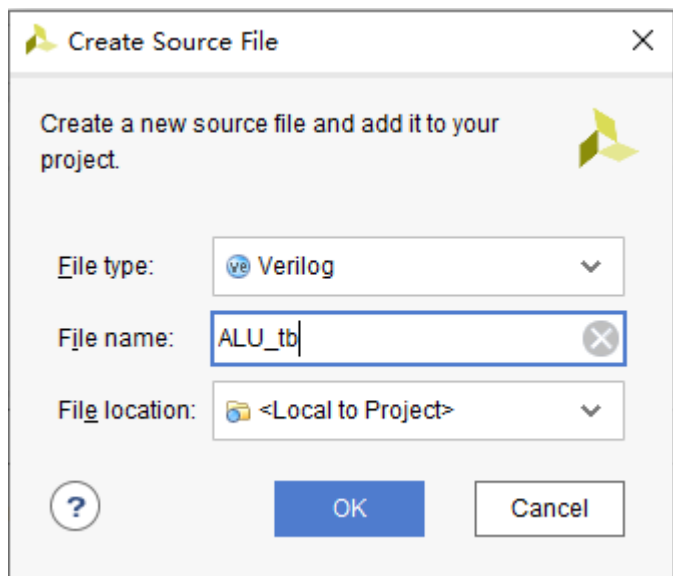
```
endcase
```

```
endmodule
```



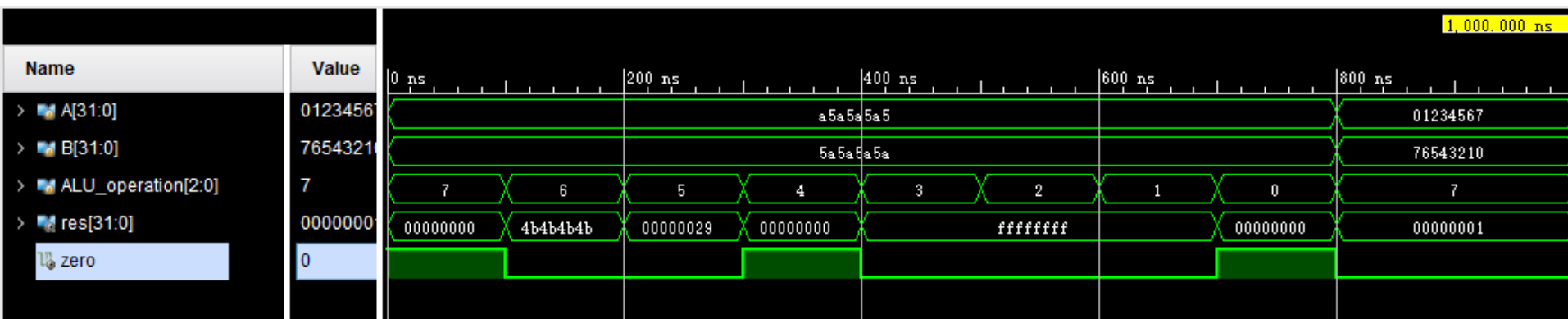
采用逻辑
表达式描
述功能

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



仿真正确之后，封装为IP

■ 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

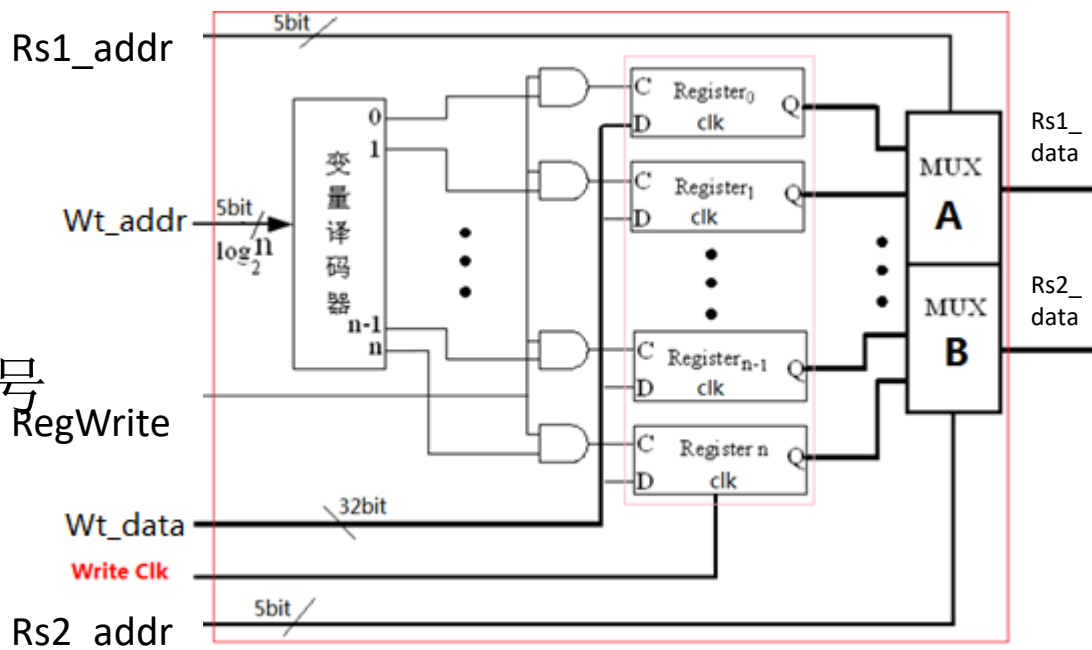
数字系统的功能部件之一：Register files

□ 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



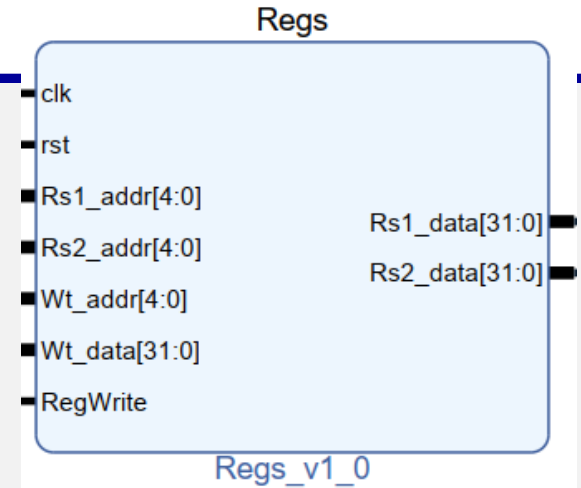
Regfile参考代码

```
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];
  assign rdata_B = (Rs2_addr== 0) ? 0 : register[Rs2_addr];

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

endmodule
```



// read
// read

// reset

// write

regfile仿真结果

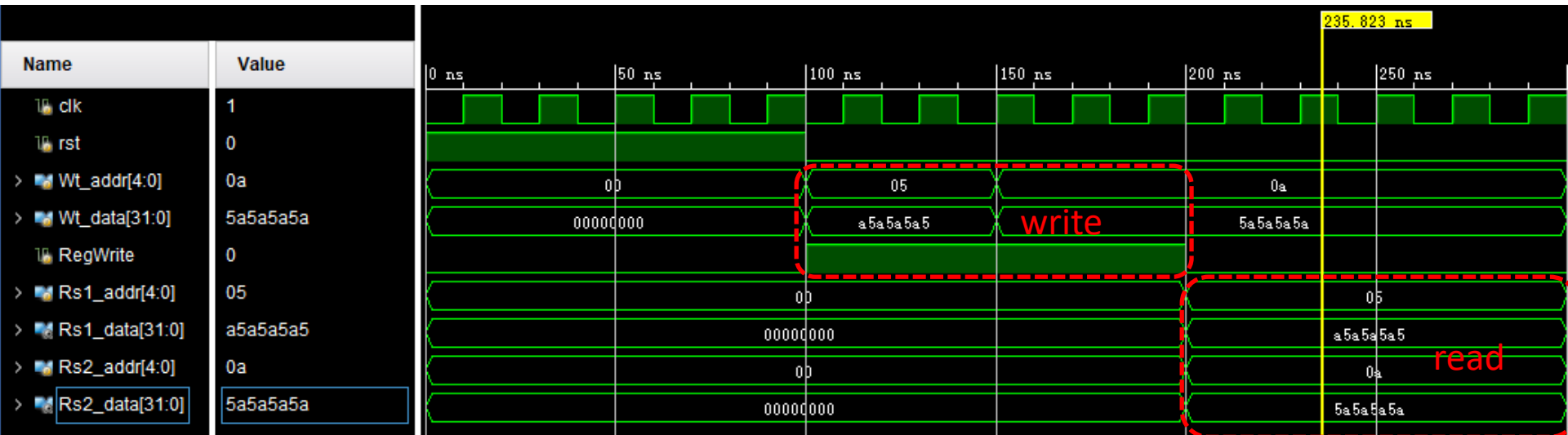
0ns-100ns regfile初始化复位，读写都为0；

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;Rs1_data[31:0]=5a5a5a5a;读地址0a



仿真正确之后，封装为IP

Regfile封装关键点

- ❑ 寄存器堆带有clk和rst；直接封装时会存在两个问题
 - 端口警告
 - 复位信号自动反向

The screenshot displays the Vivado IP packaging workflow. On the left, the 'Packaging Steps' sidebar shows 'Ports and Interfaces' as the current step, highlighted with a red dashed box. The main area is titled 'Review and Package' and shows a warning icon with the text '1 warning 2 info messages', also highlighted with a red dashed box. Below this, the 'Summary' section provides details for the IP: Display name: REG32_v1_0, Description: REG32_v1_0, and Root directory: c:/users/asus/desktop/oexp_riscv/oexp04/oexp04-datapath/cpu/oexp04-pc_reg32/oexp04-pc_reg32.srscs/sources_1/new. The 'After Packaging' section contains instructions on where the IP archive will be created and how it will be made available in the catalog. On the right, a block diagram of the 'Regs_v1_0' IP is shown, with its ports listed: clk, rst, Rs1_addr[4:0], Rs1_data[31:0], Rs2_addr[4:0], Rs2_data[31:0], Wt_addr[4:0], Wt_data[31:0], and RegWrite. The 'rst' port is highlighted with a red dashed box, indicating the warning mentioned in the interface.

Packaging Steps

- ✓ Identification
- ✓ Compatibility
- ✓ File Groups
- Customization Parameters
 - Ports and Interfaces**
- Addressing and Memory
- ✓ Customization GUI
- ✓ Review and Package

Review and Package

1 warning 2 info messages

Summary

Display name: REG32_v1_0

Description: REG32_v1_0

Root directory: c:/users/asus/desktop/oexp_riscv/oexp04/oexp04-datapath/cpu/oexp04-pc_reg32/oexp04-pc_reg32.srscs/sources_1/new

After Packaging

Create archive of IP - c:/users/asus/desktop/oexp_riscv/oexp04/oexp04-datapath/cpu/oexp04-pc_reg32/oexp04-pc_reg32.srscs/sources_1/new/x

[edit](#)

IP will be made available in the catalog using the repository -

c:/Users/ASUS/Desktop/OExp_RISCV/OExp04/OExp04-DataPath/CPU/OExp04-PC_REG32/OExp04-PC_REG32.srscs/sources_1/new

[Edit packaging settings](#)

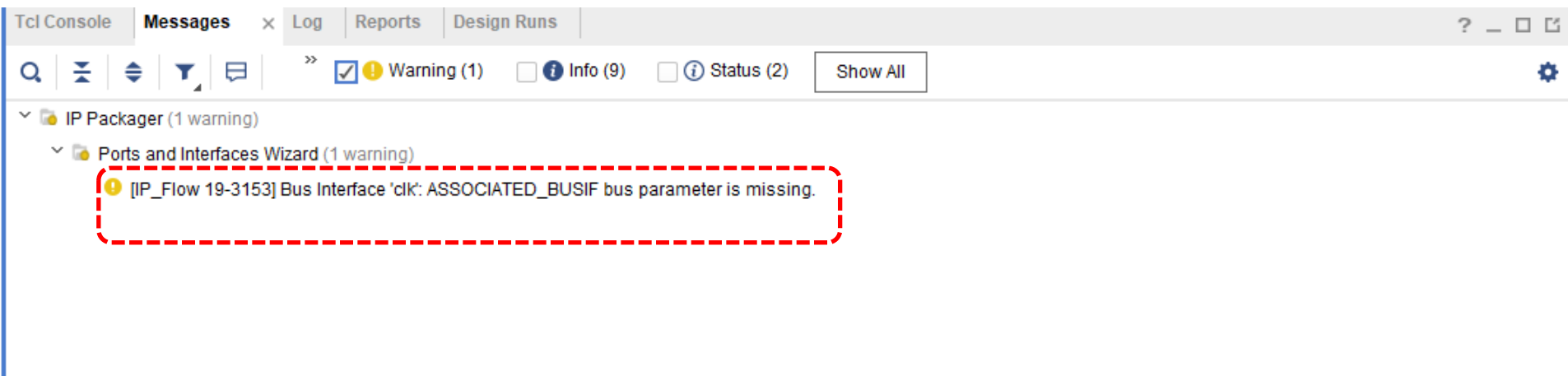
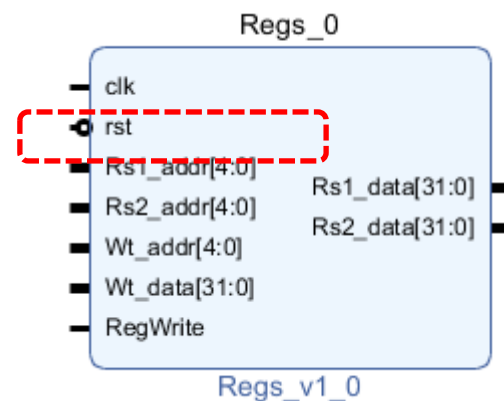
Regs_v1_0

- clk
- rst
- Rs1_addr[4:0]
- Rs1_data[31:0]
- Rs2_addr[4:0]
- Rs2_data[31:0]
- Wt_addr[4:0]
- Wt_data[31:0]
- RegWrite

Re-Package IP

Regfile封装关键点

- 寄存器堆带有clk和rst；直接封装时会存在两个问题
 - 端口警告原因是clk端口属性未知
 - 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



Regfile封装关键点--clk

□ 点击clk进入端口编辑界面,Parameters下添加

ASSOCIATED_BUSIF

The screenshot displays the 'Packaging Steps' on the left, with 'Ports and Interfaces' highlighted. The main area shows the 'Ports and Interfaces' table, where the 'clk' port is selected. The 'Edit Interface' dialog is open, showing the 'Parameters' tab. The 'Parameters' table lists 'ASSOCIATED_RESET'. The 'Add Parameter' dialog is also open, showing the 'Enter new Bus Parameter name:' field with 'ASSOCIATED_BUSIF' entered.

Packaging Steps

- Identification
- Compatibility
- File Groups
- Customization Parameters
- Ports and Interfaces**
- Addressing and Memory
- Customization GUI
- Review and Package

Ports and Interfaces ! 1

Name	Interface Mode	Enablement Dependency	Is Declaration	Direction	Driver Value	Size Left	Size Right	Size Left Dependency	Size Right Dependency
> Clock and Reset Signals			<input type="checkbox"/>						
> rst	slave		<input type="checkbox"/>						
> clk	slave		<input type="checkbox"/>						
CE									
D									
Q									

Edit Interface

Use the tabs and fields below to modify the Bus Interface on your IP.

Parameters

Name	Description	Display Name	Usage	Value	Parameter Types
ASSOCIATED_RESET			all	rst	

Add Parameter

Enter new Bus Parameter name:

ASSOCIATED_BUSIF

OK Cancel

Regfile封装关键点--clk

- 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

Edit Interface

Use the tabs and fields below to modify the Bus Interface on your IP.

General Port Mapping Parameters

Q + C

Name	Description	Display Name	Usage	Value	Parameter Types
ASSOCIATED_RESET			all	rst	
ASSOCIATED_BUSIF	List of bus interface names separated by colons. For example, m_axis_a:s_axis_b:s_axis_c		all	clk	

Regfile封装关键点--rst

- 点击rst进入端口编辑界面,Parameters下添加POLARITY

The screenshot displays the 'Regfile' packaging interface. On the left, the 'Packaging Steps' sidebar lists various stages, with 'Ports and Interfaces' highlighted. The main area shows the 'Ports and Interfaces' table, where the 'rst' port is selected. Below this, the 'Edit Interface' dialog is open, showing the 'Parameters' tab. A red dashed box highlights the 'Parameters' tab and the 'Add' button. At the bottom, the 'Add Parameter' dialog is shown with 'POLARITY' entered as the new bus parameter name.

Packaging Steps

- ✓ Identification
- ✓ Compatibility
- ✓ File Groups
- Customization Parameters
- Ports and Interfaces**
- Addressing and Memory
- ✓ Customization GUI
- ✓ Review and Package

Ports and Interfaces ! 1

Name	Interface Mode	Enablement Dependency	Is Declaration	Direction	Driver Value	Size Left	Size Right	Size Left Dependency	Size Right Dependency	T
✓ Clock and Reset Signals			<input type="checkbox"/>							
> rst	slave		<input type="checkbox"/>							
> clk										
CE										
D										
Q										

Edit Interface

Use the tabs and fields below to modify the Bus Interface on your IP.

General **Port Mapping** **Parameters**

Name	Description ^1	Display Name	Usage	Value	Parameter Types
------	----------------	--------------	-------	-------	-----------------

Add Parameter

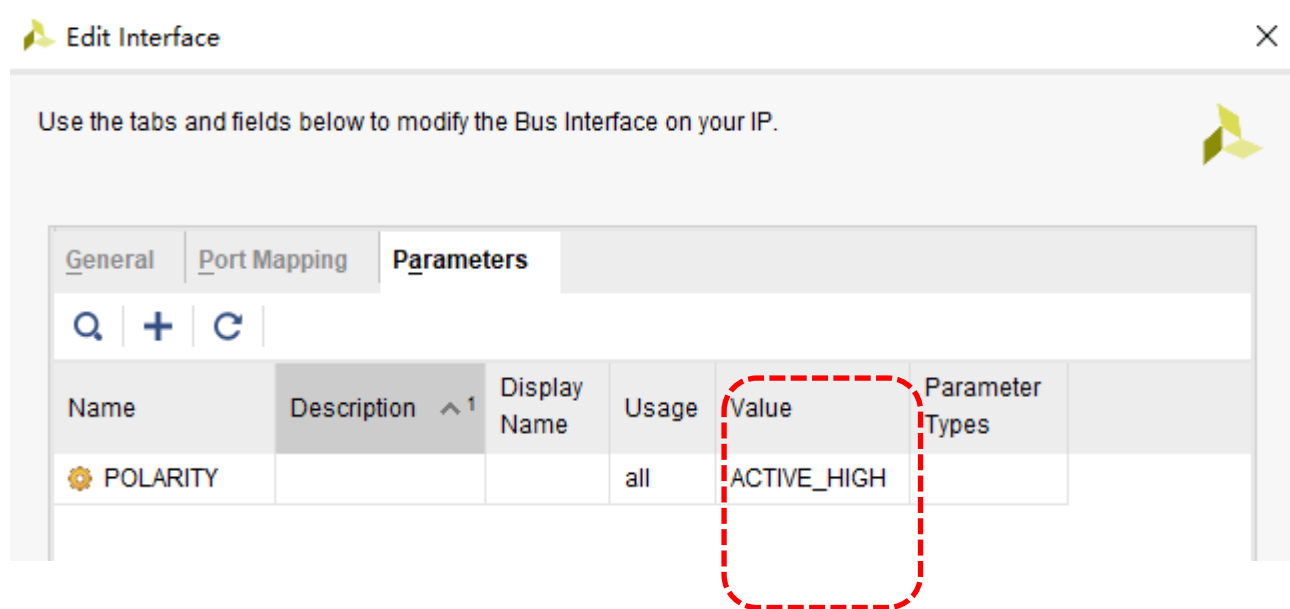
Enter new Bus Parameter name:

POLARITY

OK Cancel

Regfile封装关键点--rst

- 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

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，你认为会存在那些问题？

 **END**