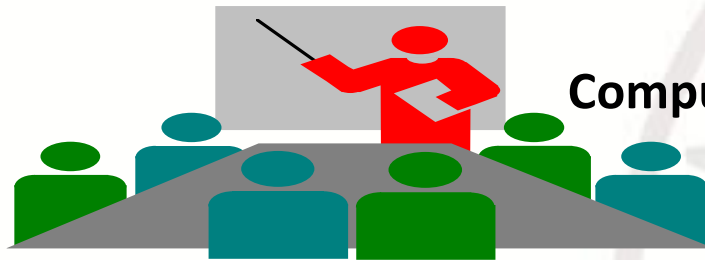




浙江大学
ZHEJIANG UNIVERSITY



Computer Organization & Design

Computer Organization & Design

实验与课程设计

实验五

CPU设计-数据通路

施青松

Asso. Prof. Shi Qingsong

College of Computer Science and Technology, Zhejiang University

zjsqs@zju.edu.cn

Course Outline





实验目的

1. 运用寄存器传输控制技术
2. 掌握CPU的核心：数据通路组成与原理
3. 设计数据通路
4. 学习测试方案的设计
5. 学习测试程序的设计



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. SWORD4.0开发板
3. Xilinx ISE14.7及以上开发工具

□ 材料

无

Course Outline



1. 设计9+条指令的数据通路

- 用逻辑原理图设计实现数据通路
 - ALU和Regs调用Exp04设计的模块
- 替换Exp04的数据通路核
- 此实验在Exp04的基础上完成

2. 设计数据通路测试方案:

- 部件测试: ALU、Register Files
- 通路测试: I-格式通路、R-格式通路

3. 设计数据通路测试程序

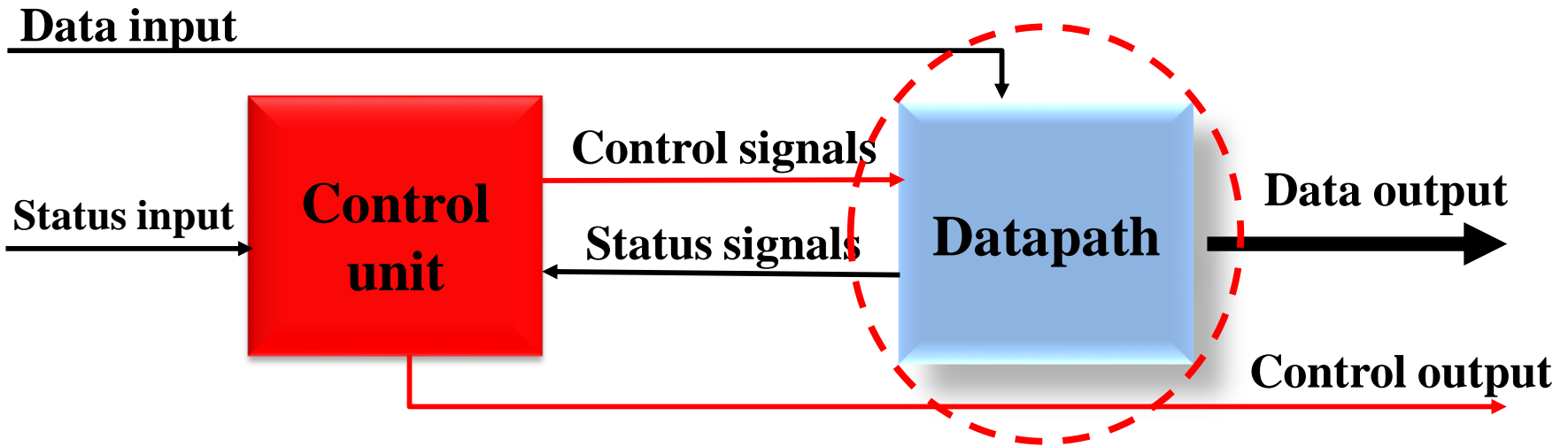
Course Outline



CPU organization

□ Digital circuit

- General circuits that controls logical event with logical gates -
-Hardware

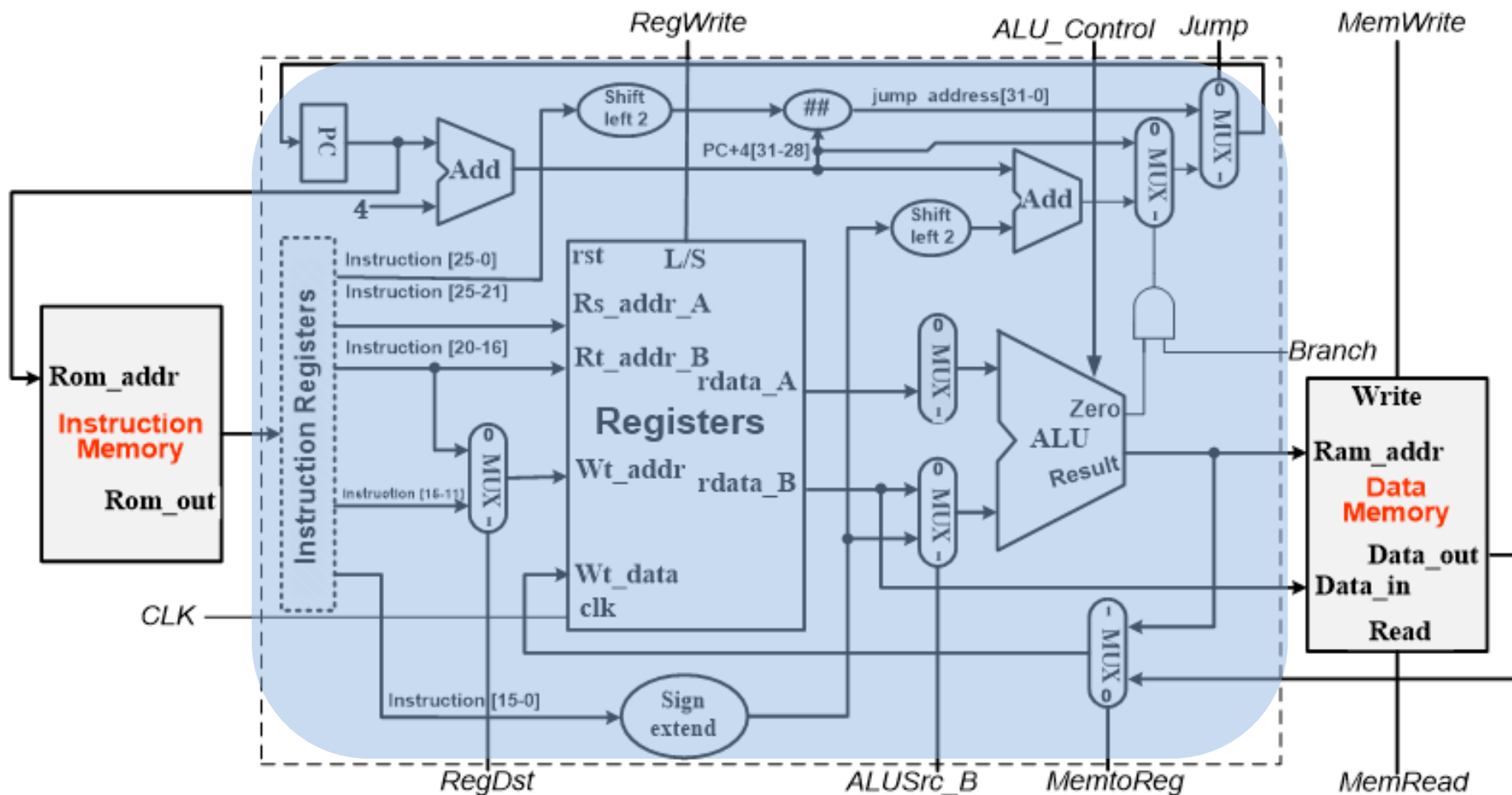


□ Computer organization

- Special circuits that processes logical action with instructions
-Software

单周期数据通路结构

找出九条指令的通路：5个MUX





控制信号定义

□ 通路与控制

信号	源数目	功能定义	赋值0时动作	赋值1时动作
ALUSrc_B	2	ALU端口B输入选择	选择寄存器B数据	选择32位立即数 (符号扩展后)
RegDst	2	寄存器写地址选择	选择指令rt域	选择指令rs域
MemtoReg	2	寄存器写入数据选择	选择存储器数据	选择ALU输出
Branch	2	Beq指令目标地址选择	选择PC+4地址	选择转移地址 (Zero=1)
Jump	2	J指令目标地址选择	选择J目标地址	由Branch决定输
RegWrite	-	寄存器写控制	禁止寄存器写	使能寄存器写
MemWrite	-	存储器写控制	禁止存储器写	使能存储器写
MemRead	-	存储器读控制	禁止存储器读	使能存储器读
ALU_Control	000-111	3位ALU操作控制	参考表 Lab4	Lab4

请填写对应操作

CPU部件之数据通路接口: **Data_path**



□ **Data_path**

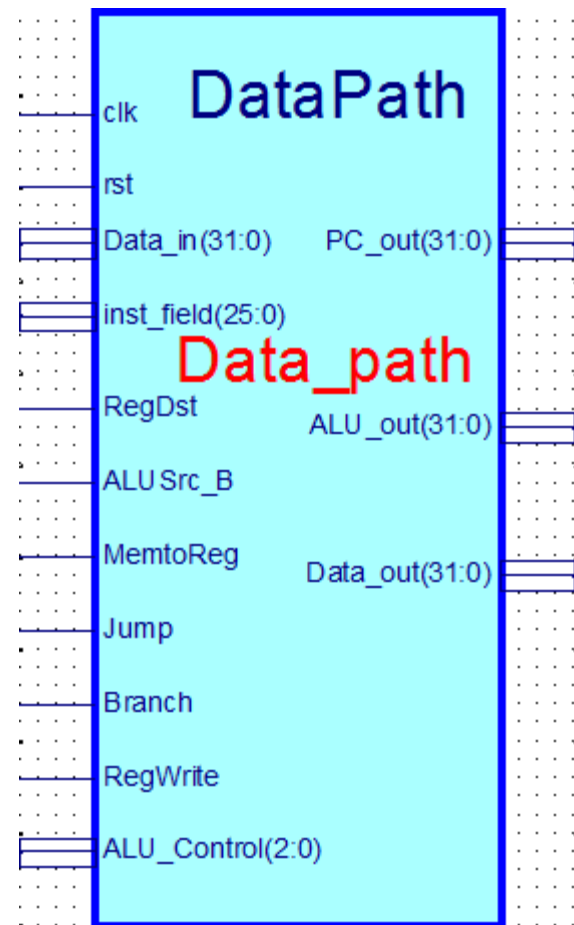
- CPU主要部件之一
- 寄存器传输控制对象: 通用数据通路

□ **基本功能**

- 具有通用计算功能的算术逻辑部件
- 具有通用目的寄存器
- 具有通用计数所需的尽可能的路径

□ **接口要求- **Data_path****

- 数据通路接口信号如右图
- 模块符号文档: Data_path.sym



数据通路接口信号标准- **Data_path.v**



```
module      Data_path( input clk,           //寄存器时钟
                      input rst,           //寄存器复位
                      input[25:0]inst_field, //指令数据域
                      input RegDst,        //Regs目的地址控制

                      input ALUSrc_B,      //ALU端口B输入选择
                      input MemtoReg,      //Regs写入数据源控制
                      input Jump,          //J指令
                      input Branch,        //Beq指令
                      input RegWrite,      //寄存器写信号
                      input[31:0]Data_in,  //      存储器输入
                      input[2:0]ALU_Control, //ALU操作控制

                      output[31:0]ALU_out, //ALU运算输出
                      output[31:0]Data_out, //CPU数据输出
                      output[31:0]PC_out   //PC指针输出
                      );
```

endmodule

Course Outline





CPU之数据通路设计

- 调用实验一设计的多路器
- 调用实验一的基本运算模块
- 调用实验四设计的ALU和Regs



设计工程：OExp05-Datapath

◎ 设计CPU之数据通路

- ☞ 根据理论课分析讨论设计9+条指令的数据通路
- ☞ 仿真测试DataPath模块

◎ 集成替换验证通过的数据通路模块

- ☞ 替换实验四(Exp04)中的Data_Path.ngc核
- ☞ 顶层模块沿用Exp04
 - 模块名：Top_OExp05_DataPath.sch

◎ 测试数据通路模块

- ☞ 设计测试程序(MIPS汇编)测试：
- ☞ ALU功能
- ☞ I-指令通路
- ☞ R-指令通路



设计要点

拷贝下列模块符号到当前工程目录：

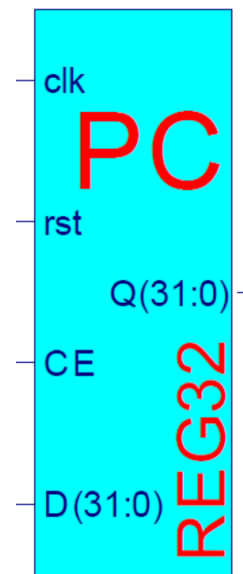
and32、or32、ADC32、xor32、nor32、srl32、
SignalExt_32、mux8to1_32、or_bit_32
add_32、mux2to1_32、mux2to1_5、
ALU、Regs、Ext_32、REG32

设计要点

□ 设计32位寄存器

- 用途：PC指针、数据、地址或指令锁存
- 参考逻辑实验Exp10，用行为描述实验
- 模块名：REG32
 - 上升沿触发：clk
 - 使能信号：CE
 - 同步复位：rst=1
 - 数据输入：D(31:0)
 - 数据输出：Q(31:0)
- 参考描述结构

```
module REG32(input clk,  
             .....  
             always @(posedge clk or posedge rst)  
                 if (rst==? ) Q <= ? ;  
                 else if (? ) Q <= ? ;  
endmodule
```

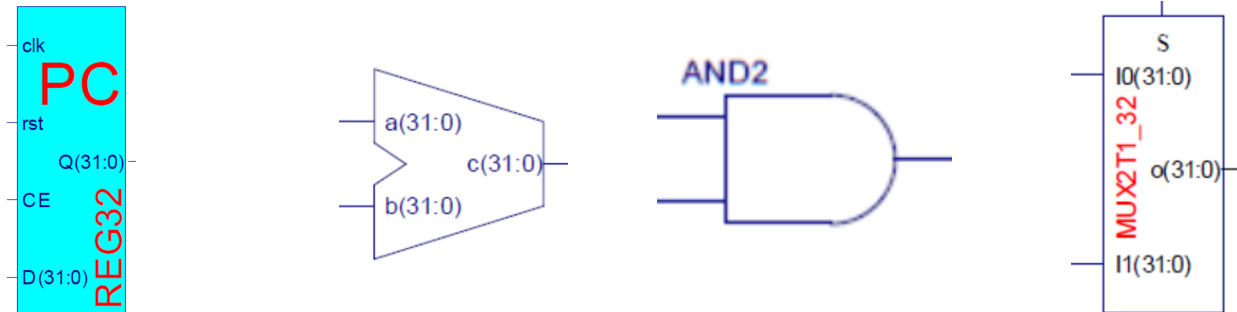


封装后的逻辑符号

建立DataPath原理图输入模板

设计PC通路

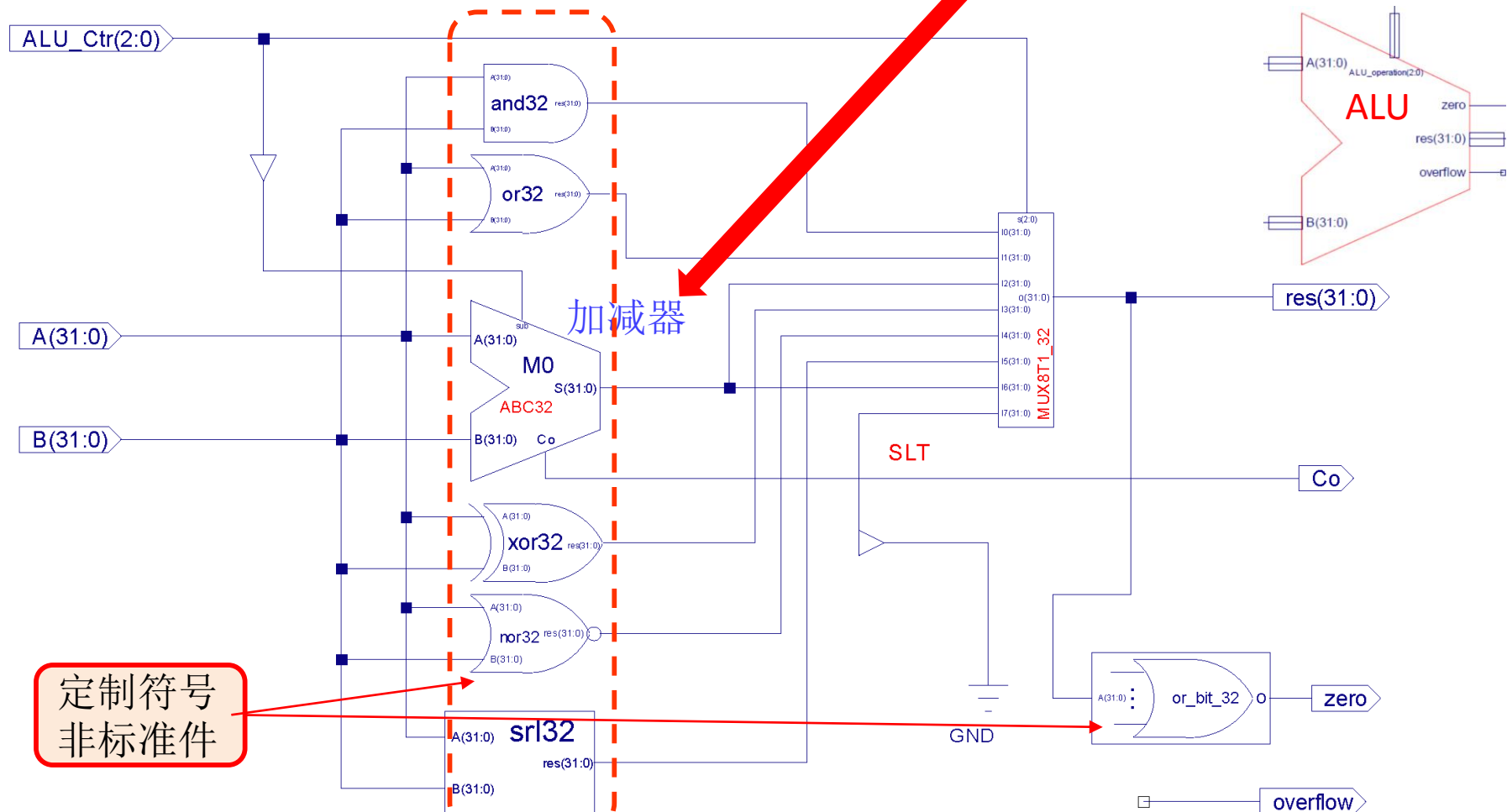
- 调用REG32、add_32、AND2(库)和MUX2T1_32模块



- 设计：
 - 顺序执行(PC+4)、Jump和Beq时的PC值
 - 计算和通路
- 注意常数的电路实现：
 - “4” = ? ? ? ?
 - Branch_offset = ? ? ? ?
 - Jump_addr = ? ? ? ?

Exp04的ALU模块

逻辑实验提供的ALU





调用Exp04的Regs模块


此代码留有BUG，请同学自行编写

```
Module regs(input clk, rst, L_S,
             input [4:0] R_addr_A, R_addr_B, Wt_addr,
             input [31:0] wt_data
             output [31:0] rdata_A, rdata_B
             );
    reg [31:0] register [1:31];
    integer i;

    assign rdata_A = (Rs_addr_A == 0) ? 0 : register[reg_Rd_addr_A]; // read
    assign rdata_B = (Rt_addr_B == 0) ? 0 : register[reg_Rt_addr_B]; // 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 ((Rd_addr != 0) && (we == 1))
              register[Wt_addr] <= wdata; // write
    end

endmodule
```



代码来自李亚民教授

□ 设计R-格式和I-格式数据通路

■ 根据理论课分析讨论的数据通路选择MUX

■ Regs数据通道设计

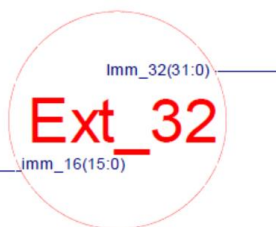
- R-格式源地址通道选择: rs、rt
- R-格式目的地址通道选择: rd
- R-格式目的数据通道选择: from ALU
- I-格式源地址通道选择: rs
- I-格式目的地址通道选择: rt
- I-格式目的数据通道选择: from Memory
- Beq的源地址通道是什么?

是什么地址?

控制信号是什么

■ ALU数据通路设计

- ALU输入端口A有通道选择吗?
- ALU输入端口B通道选择
 - 调用Ext_32模块
 - R-格式: from Regs B port
 - I-格式: Where from





不唯一





□ 数据通路仿真调试

■ DataPath模块仿真

□ 原理图检查没有Errors和warnings后仿真测试

□ 仿真激励代码设计要点

- 只做功能性测试，不做性能和完备性测试

- 通路功能测试

 - » 选择9条指令所有可能通路的代表指令

 - » 激励输入：

 - 计算出对应指令的输入控制信号和代表数据
 - clk、rst

- ALU功能测试

 - » 选择add、and、sub、or、nor、slt指令

 - 计算出对应指令的输入控制信号和代表数据

 - » 选择Beq比较、Load和Store测试地址计算

- Regs功能测试

 - » add指令代表作寄存器遍历测试



DataPath替换集成

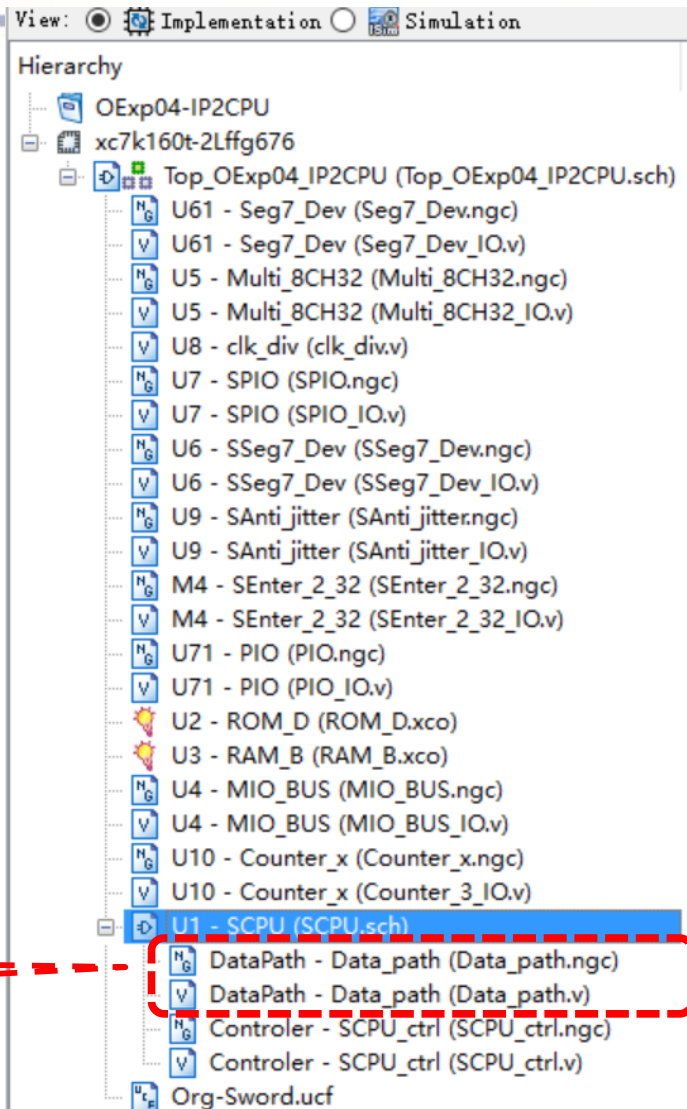
□ 集成替换

- 仿真正确后替换Exp04的数据通路IP核

□ 清理Exp04工程

- 移除工程中的数据通路核
 - Exp04工程中移除数据通路核关联
- 删除工程中CPU核文件
 - Data_path.ngc和 Data_path.v 文件
 - 在Project菜单中运行:
Cleanup Project Files ...
- 建议用Exp04资源重建工程
 - 除Data_path.ngc核

Exp04需要清理的核

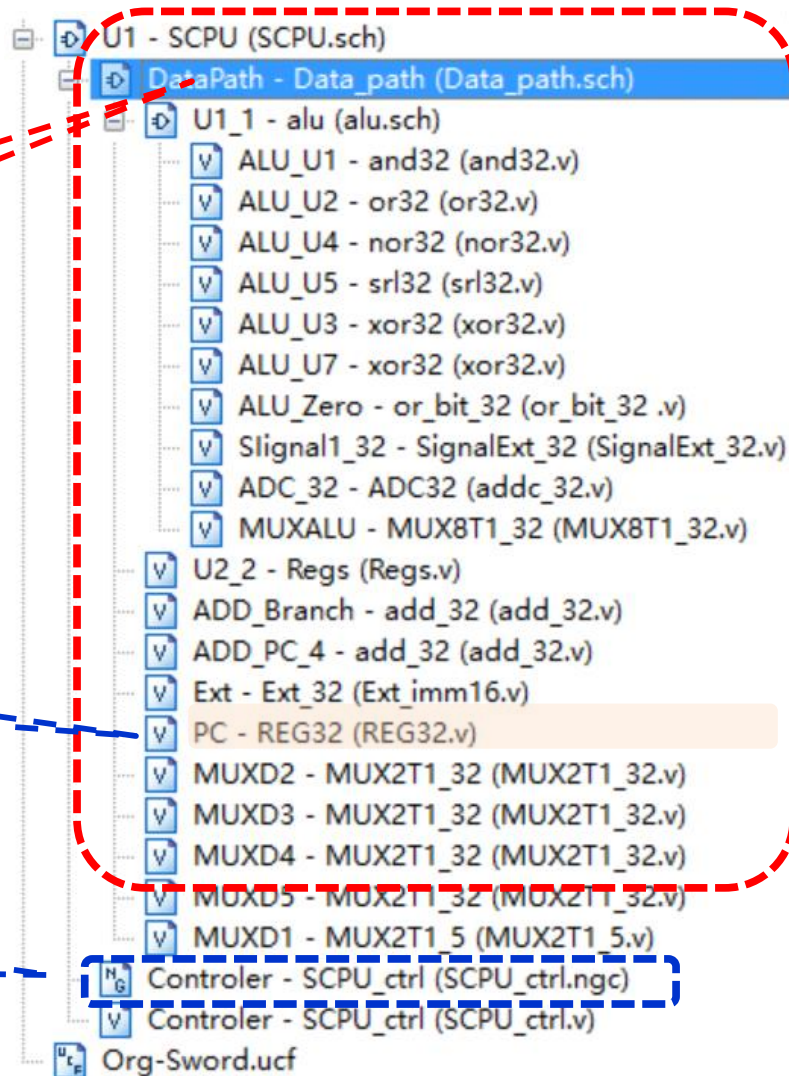


集成替换DapaPath核后的模块层次结构

Exp05完成数据通路替换后的模块调用关系

PC计数器
本实验设计

控制器不变
仍然使用IP核





物理验证

□ 使用DEMO程序目测数据通路功能

■ DEMO接口功能

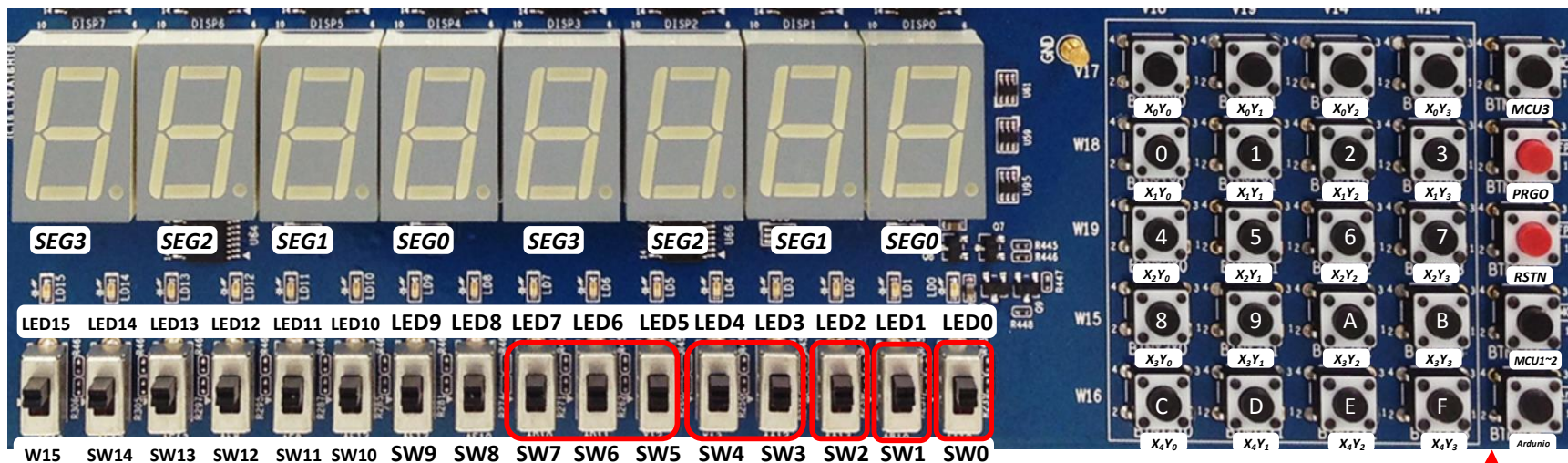
□ SW[7:5]=000, SW[2]=0(全速运行)

- SW[4:3]=00, SW[0]=0, 点阵显示程序: 跑马灯
- SW[4:3]=00, SW[0]=0, 点阵显示程序: 矩形变幻
- SW[4:3]=01, SW[0]=1, 内存数据显示程序: 0~F
- SW[4:3]=10, SW[0]=1, 当前寄存器R9+1显示

□ 用汇编语言设计测试程序

- 测试ALU功能
- 测试Regs访问
- 测试I-格式指令通路
- 测试R-格式指令通路

物理验证-DEMO接口功能



SW[7:5]=显示通道选择

SW[7:5]=000: CPU程序运行输出

SW[7:5]=001: 测试PC字地址

SW[7:5]=010: 测试指令字

SW[7:5]=011: 测试计数器

SW[7:5]=100: 测试RAM地址

SW[7:5]=101: 测试CPU数据输出

SW[7:5]=110: 测试CPU数据输入

SW[0]=文本图形选择

SW[1]=高低16位选择

SW[2]=CPU单步时钟选择

没有使用

DEMO功能, 测试程序可以替换成自己的功能

SW[4:3]=00, 点阵显示程序: 跑马灯

SW[4:3]=00, 点阵显示程序: 矩形变幻

SW[4:3]=01, 内存数据显示程序: 0~F

SW[4:3]=10, 当前寄存器+1显示



测试程序参考：ALU和Regs

□ 设计ALU和Regs测试程序替换DEMO程序

- ALU、Regs测试参考设计，测试结果通过CPU输出信号单步观察
- SW[7:5]=100, Addr_out=ALU输出
- SW[7:5]=101, Data_out=寄存器B输出

```
#baseAddr 0000
loop:  nor r1,r0,r0;      //r1=FFFFFFFF
      slt r2,r0,r1;      //r2=00000001
      add r3,r2,r2;      //r3=00000002
      add r4,r3,r2;      //r4=00000003
      add r5,r4,r3;      //r5=00000005
      add r6,r5,r4;      //r6=00000008
      add r7,r6,r5;      //r7=0000000d
      add r8,r7,r6;      //r8=00000015
      add r9,r8,r7;      //r9=00000022
      add r10,r9,r8;     //r10=00000037
      add r11,r10,r9;    //r11=00000059
      add r12,r11,r10;   //r12=00000090
      add r13,r12,r11;   //r13=000000E9
      add r14,r13,r12;   //r14=00000179
      add r15,r14,r13;   //r15=00000262
```

```
add r16,r15,r14; //r16=000003DB
add r17,r16,r15; //r17=000006D3
add r18,r17,r16; //r18=00000A18
add r19,r18,r17; //r19=000010EB
add r20,r19,r18; //r20=00001B03
add r21,r20,r19; //r21=00003bEE
add r22,r21,r20; //r22=000046F1
add r23,r22,r21; //r23=000080DF
add r24,r23,r22; //r24=0000C9D0
add r25,r24,r23; //r25=00014AAF
add r26,r25,r24; //r26=0001947F
add r27,r26,r25; //r27=0012DF2E
add r28,r27,r26; //r28=001473AD
add r29,r28,r27; //r29=002752DB
add r30,r29,r28; //r30=003BC688
add r31,r30,r29; //r31=00621963
```

j loop;



测试程序参考

□ 设计通道测试程序替换DEMO程序

- 通道测试参考设计。测试结果通过CPU输出信号单步观察
- 通道功能由传输数据结果来指示，如立即数通道观察：14+\$zero

#baseAddr 0000

```
start:                                     //通道结果由后一条指令读操作数观察
      lw  r5, 14($zero);                  //取测试常数55555555。存储器读通道
start_A:
      add r1, r5, $zero;                  //r1: 寄存器写通道。R5:寄存器读通道A输出
      nor r2, $zero, r1;                  //r1: 寄存器读通道B输出。R2:ALU输出通道
      lw  r5, 48($zero); //取测试常数AAAAAAAA。立即数通道:00000048
      beq r2, r5 start_A;                 //循环测试
      j   start;                          //循环测试。立即数通道: 00000014
```

□ 测试的完备性

- 上述测试正确仅表明通道切换功能和总线传输部分正确
- 要测试其完全正确，必须遍历所有可能的情况



数据存储模块测试

□ 设计存储器模块测试程序

- 7段码显示器的地址是E0000000/FFFFFFE0
- LED显示地址是F0000000/FFFFFFF0
- 请设计存储器模块测试程序
 - 测试结果在7段显示器上指示

□ RAM初始化数据同Exp03

F0000000, 000002AB, 80000000, 0000003F, 00000001, FFF70000,
0000FFFF, 80000000, 00000000, 11111111, 22222222, 33333333,
44444444, 55555555, 66666666, 77777777, 88888888, 99999999,
aaaaaaaa, bbbbbbbb, cccccccc, dddddddd, eeeeeeee, FFFFFFFF,
557EF7E0, D7BDFBD9, D7DBFDB9, DFCFFCFB, DFCFBFFF, F7F3DFFF,
FFFFDF3D, FFFF9DB9, FFFFBCFB, DFCFFCFB, DFCFBFFF, D7DB9FFF,
D7DBFDB9, D7BDFBD9, FFFF07E0, 007E0FFF, 03bdf020, 03def820,
08002300;



设计测试记录表格

- 学会实验数据的统计
 - 参考大学物理实验
 - 本实验没有有效数精确计算，但有大量数据表格
- ALU和Regs测试结果记录
 - 自行设计记录表格
- 通道测试结果记录
 - 自行设计记录表格
- 数据存储模块测试记录
 - 自行设计记录表格

思考题



□ 扩展下列指令，数据通路将作如何修改：

R-Type: srl*, jr, jalr, eret*;

I-Type: addi, andi, ori, xori, lui, bne, slti

J-Type: Jal;

□ 增加I-Type算术运算指令是否需要修改本章设计的数据通路？



● END