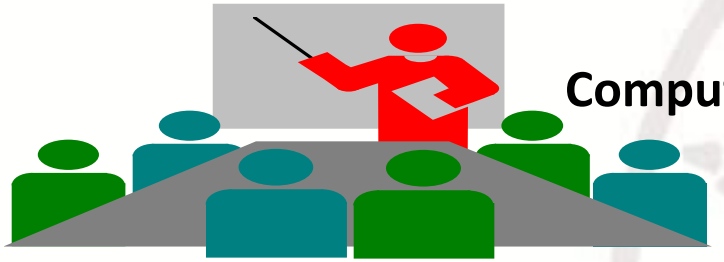


Computer Organization & Design
实验与课程设计

实验七

CPU设计-指令集扩展

施青松
Asso. Prof. Shi Qingsong
College of Computer Science and Technology, Zhejiang University
zjsqs@zju.edu.cn

Course Outline

- 实验目的与实验环境
- 实验任务
- 实验原理
- 实验操作与实现

实验目的

1. 运用寄存器传输控制技术

2. 掌握CPU的核心：指令执行过程与控制流关系

3. 设计数据通路和控制器

4. 设计测试程序

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. Spartan-3 Starter Kit Board/Sword开发板

3. Xilinx ISE14.4及以上开发工具

材料

无

Course Outline

- 实验目的与实验环境
- 实验任务
- 实验原理
- 实验操作与实现

实验任务

1. 扩展实验六CPU指令集

- 重新设计数据通路和控制器
  - 兼容Exp05的数据通路和控制器
  - 替换Exp05的数据通路控制器核
- 扩展不少于下列指令

  R-Type： add, sub, and, or, xor, nor, slt, srl*, jr, jalr, eret；

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

  J-Type： J, Jal*；
- 此实验在Exp06的基础上完成

2. 设计指令集测试方案

3. 设计指令集测试程序

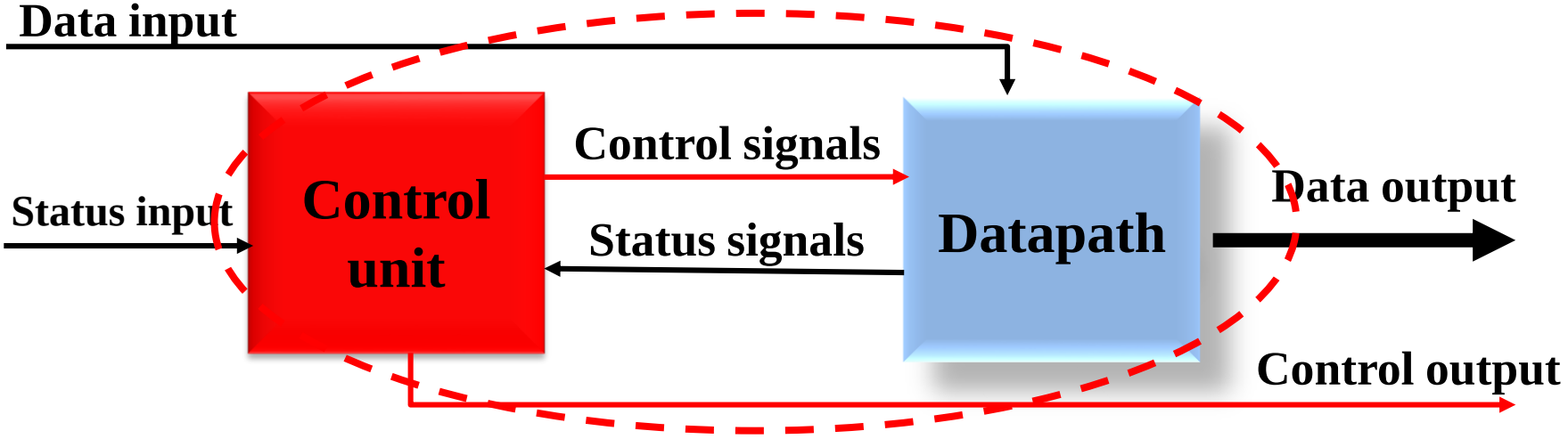
Course Outline

- 实验目的与实验环境
- 实验任务
- 实验原理
- 实验操作与实现

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

控制信号定义

- 需要增加那些通路与操作控制
  - 兼容9条指令通路与控制

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决定输出
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 Exp04	Exp04

控制信号真值表

- 根据数据通路重新设计控制器输出信号真值表

OP	Reg Dst	ALU Src	Mem toReg	Reg Write	Mem Read	Mem Write	Branch	Jump	ALU op1	ALU op0
R-格式 000000										
I-格式LW										
I-格式SW										
I-格式beq										
J-格式										

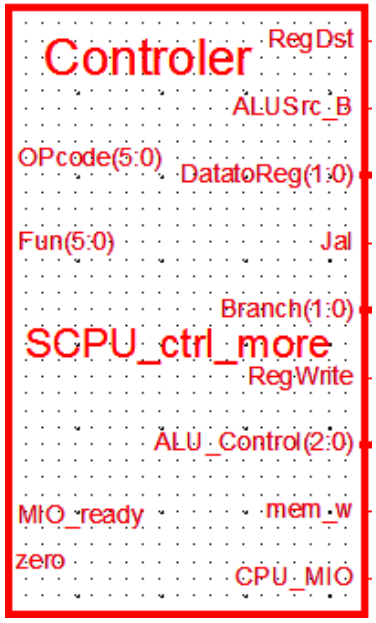
重新设计真值表

需要增加控制信号吗？

需要ALU_{OP}吗？

重新设计数据通路与控制器接口：

- 重新设计接口
  - 扩展后增加了控制信号
  - 数据通路参考接口如右图
    - 模块符号文档：Data_path_more.sym
  - 控制器参考接口信号如下图
    - 模块符号文档：SCPU_ctrl_more.sym



数据通路功能控制器接口信号标准

```
module    SCPU_ctrl_more( input[5:0]OPcode,          //OPcode
                          input[5:0]Fun,             //Function
                          input MIO_ready,           //CPU Wait
                      ……
                      output reg mem_w,
                      output reg [2:0]ALU_Control,
                      output reg CPU_MIO
                      );
endmodule
```

```
module     Data_path_more( input clk,                //寄存器时钟
                           input rst,                //寄存器复位
                           input[25:0]inst_field,    //指令数据域
                           ……
                           output[31:0]ALU_out,
                           output[31:0]Data_out,
                           output[31:0]PC_out
                           );
endmodule
```

Course Outline

- 实验目的与实验环境
- 实验任务
- 实验原理
- 实验操作与实现

CPU之控制器扩展设计

-扩展实验六设计的CPU功能

设计工程：OExp07-ExtSCPU

◎扩展不少于下列指令

R-Type： add, sub, and, or, xor, nor, slt, srl*, jr, jalr, eret；

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

J-Type： J, Jal*；

◎ 集成替换验证通过的新CPU

- 替换实验六(Exp06)中的SCPU模块
- 替换实验六(Exp06)中的SCPU_ctrl模块
- 替换实验六(Exp06)中的Data_Path模块
- 顶层模块延用Exp05
  - 模块名：Top_OExp06_ExtSCPU.sch
  - 需要修改CPU逻辑符号

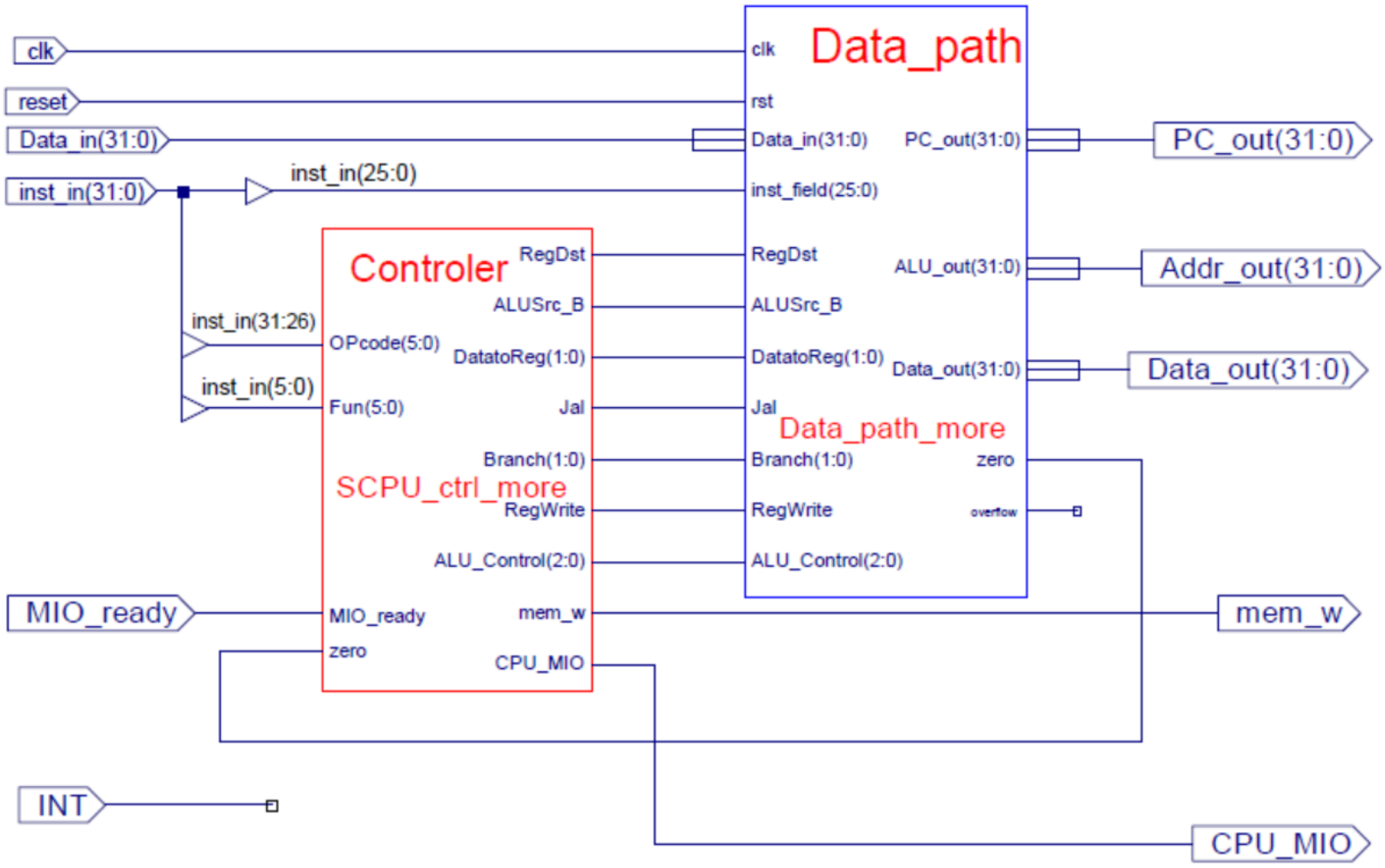
◎ 测试扩展后的CPU功能

- 设计测试程序(MIPS汇编)测试

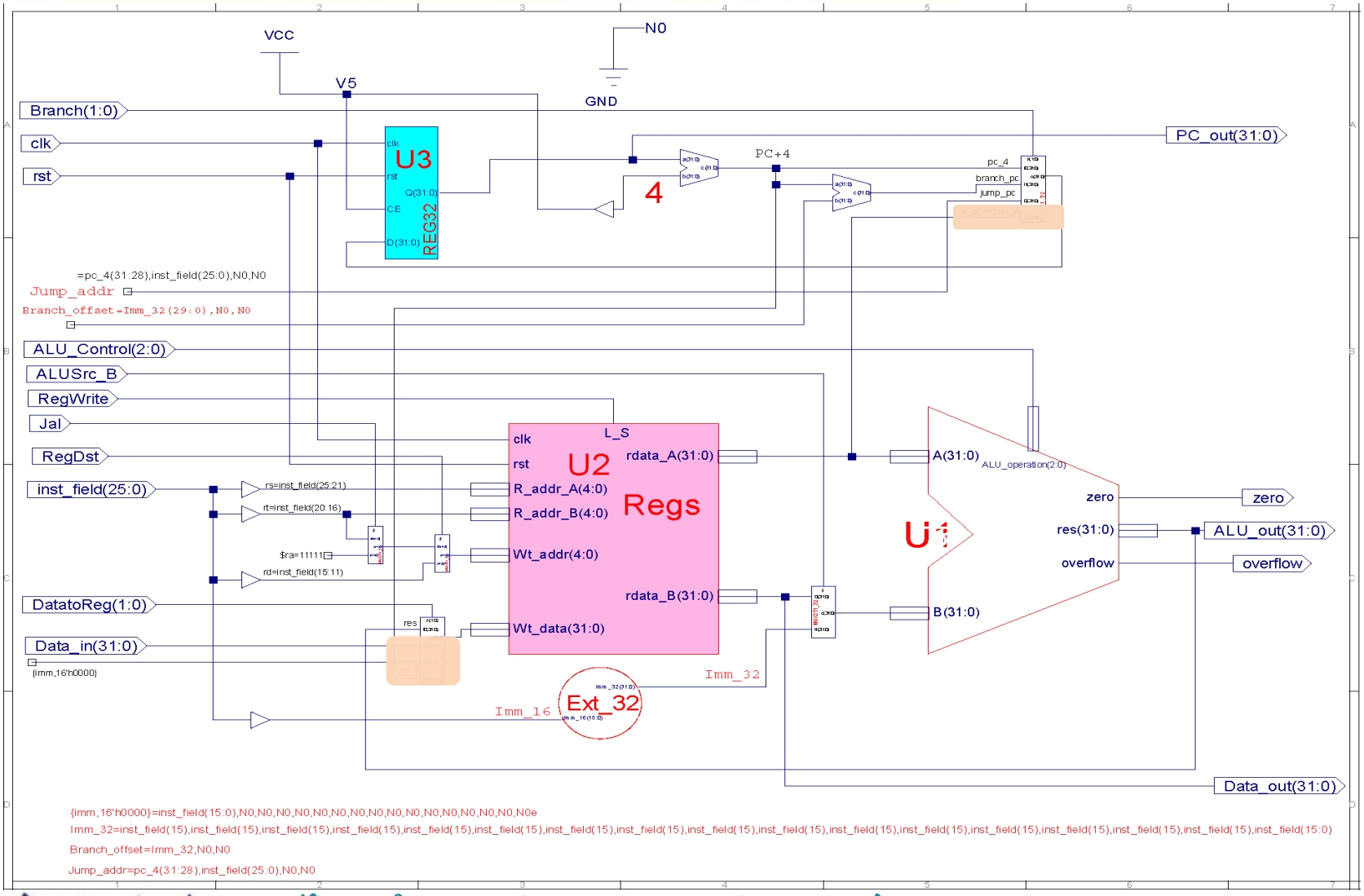
设计要点

- **设计指令扩展后DataPath结构**
  - 需要根据新的接口信号重新设计逻辑符号
  - 在实验五的原理图上扩展
- **根据新DataPath结构设计控制器**
  - 需要根据新的接口信号重新设计逻辑符号
  - 建议用HDL结构化描述
- **设计CPU调用模块**
  - 根据新的控制器和数据通路接口信号设计CPU模块
  - 重新设计CPU逻辑符号
- **仿真新设计的模块**
  - 独立仿真DataPath和控制器
- **集成替换CPU及子模块**
  - 仿真正确后
    - 集成替换CPU、数据通路和控制器模块

扩展指令后的CPU参考模块



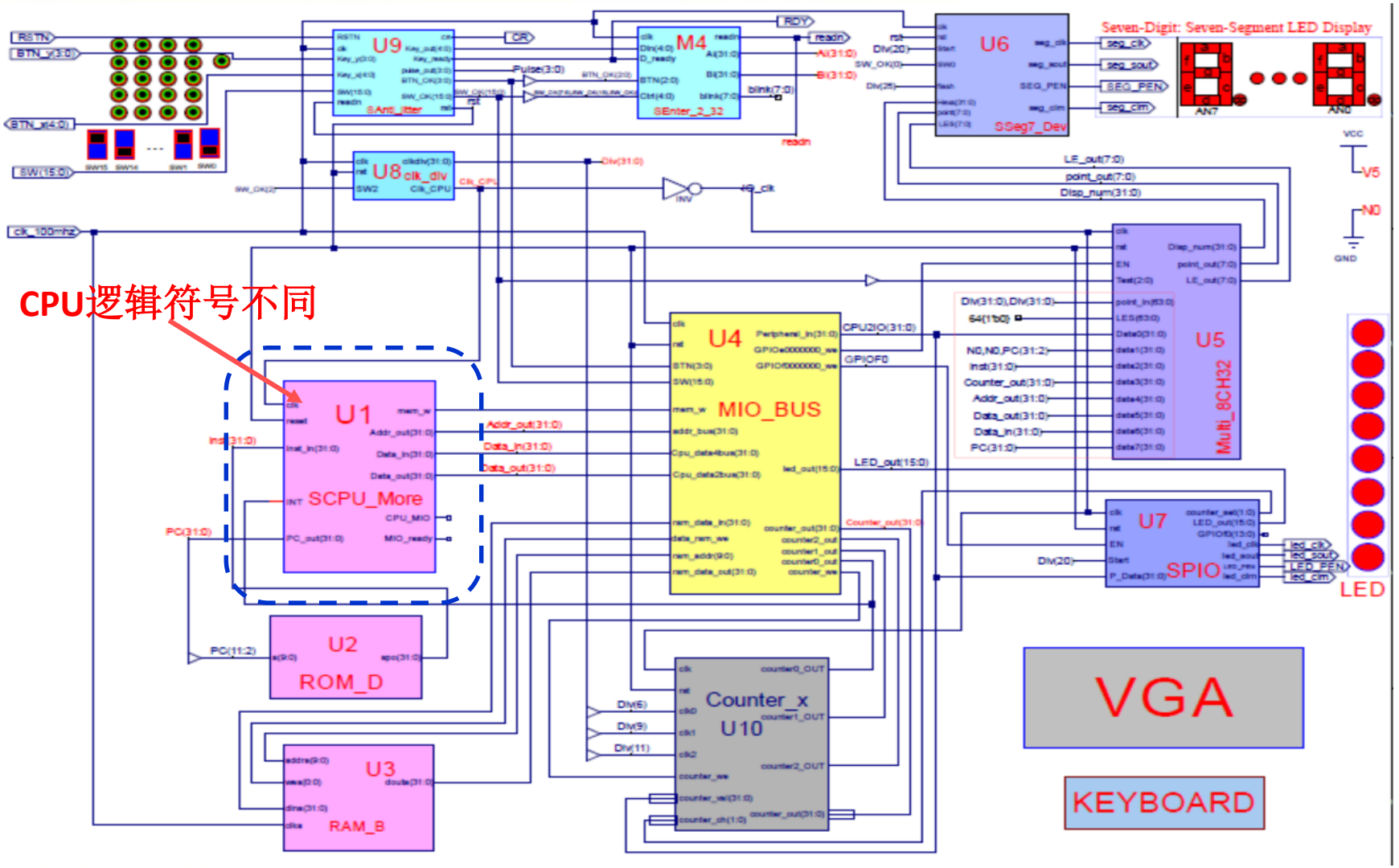
Datapath参考设计



控制器描述参考结构

```
`define CPU_ctrl_signals
{RegDst,ALUSrc_B,MemtoReg,RegWrite,MemRead,MemWrite,Branch,Jump, ALU_Control，…
        ……}
        assign mem_w = MemWrite&&(~MemRead);
        always @* begin
                case(OPcode)
                6‘b000000:                                              //ALU
                  case(Fun)
                    6'b100000: begin CPU_ctrl_signals = ?;  end         //add
                    6'b100010: begin CPU_ctrl_signals = ?;  end         //sub
                                        ……
                    default:    begin CPU_ctrl_signals = ?;  end;
                  endcase
                6'b100011: begin CPU_ctrl_signals = ?;  end             //load
                6'b101011: begin CPU_ctrl_signals = ?;  end             //store
                                                ……
                default:    begin  CPU_ctrl_signals = ?;  end
                endcase
        end
```

实验七的顶层模块结构


CPU逻辑符号不同
U1
SCPU_More
U2
ROM_D
U3
RAM_B
U4
MIO_BUS
U5
Multi_8CH32
U6
SSeg7_Dev
U7
SPIO
U8
clk_div
U9
SAnti_jitter
M4
SEnter_2_32
U10
Counter_x
Seven-Digit: Seven-Segment LED Display
LED
VGA
KEYBOARD

CPU调试与测试

- 调试
  - SCPU_ctrl_more模块仿真
    - 设计测试激励代码仿真测试*
  - Data_path_more模块仿真
    - 设计测试激励代码仿真测试*
- 集成替换
  - 仿真正确后逐个替换Exp06的相应模块
  - 使用DEMO程序目测控制器正常运行
    - DEMO程序与前面实验不一样
    - 也可自行设计

```
memory_initialization_radix=16;
memory_initialization_vector=
08000008, 00000020, 00000020, 00000020, 00000020, 00000020, 00000020, 00000020, 3c03f000,
3c04e000, 3c088000, 2014003f, 3c06f800, 00000827, 0001102a, 202affff, ac660004, 8c650000,
00a52820, 00a52820, ac650000, 21290001, ac890000, 8c0d0014, 8c650000, 00a52820, 00a52820,
ac650000, 8c650000, 00a85824, 21ad0001, 15a00001, 0c000037, 8c650000, 20120008, 0252b020,
02569020, 00b25824, 11600005, 11720009, 20120008, 1172000a, ac890000, 08000018, 15410002,
00005027, 014a5020, ac8a0000, 08000018, 8e290060, ac890000, 08000018, 8e290020, ac890000,
08000018, 8c0d0014, 014a5020, 354a0001, 22310004, 02348824, 01224820, 15210001, 21290005,
8c650000, 00a55820, 016b5820, ac6b0000, ac660004, 03e00008;
```

设计测试记录表格

- **CPU指令**测试结果记录
  - 自行设计记录表格

思考题

- 指令扩展时控制器用二级译码设计存在什么问题？
- 设计bne指令需要增加控制信号吗？
- 设计andi时需要增加新的数据通道吗？


END