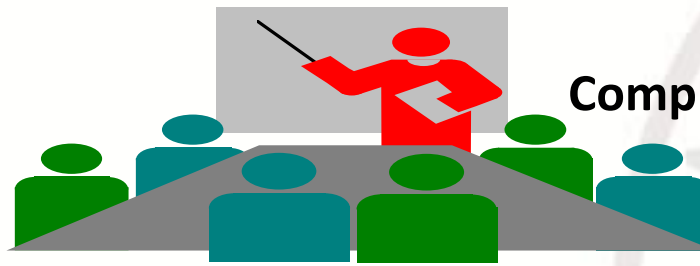




浙江大学
ZHEJIANG UNIVERSITY



Computer Organization & Design

Computer Organization & Design 实验与课程设计

实验八

CPU设计-中断

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. 深入理解CPU结构
3. 学习如何提高CPU使用效率
3. 学习CPU中断工作原理
4. 设计中断测试程序



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

A vertical diagram showing four steps of a course outline. Each step is represented by a white circle on the left, connected by a vertical line, with a corresponding colored rectangular bar to its right. The bars are blue for the first, third, and fourth steps, and yellow for the second step.

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：选修

1. 扩展实验七CPU中断功能

- 修改设计数据通路和控制器
 - 兼容Exp07数据通路增加中断通路
 - 增加中断控制
 - 修改或替换Exp07的数据通路及控制器
- 扩展CPU中断功能
 - 非法指令中断；
 - 算术溢出中断；
 - 外部中断；
- 此实验在Exp07的基础上完成

2. 设计CPU中断测试方案

3. 设计CPU中断测试程序

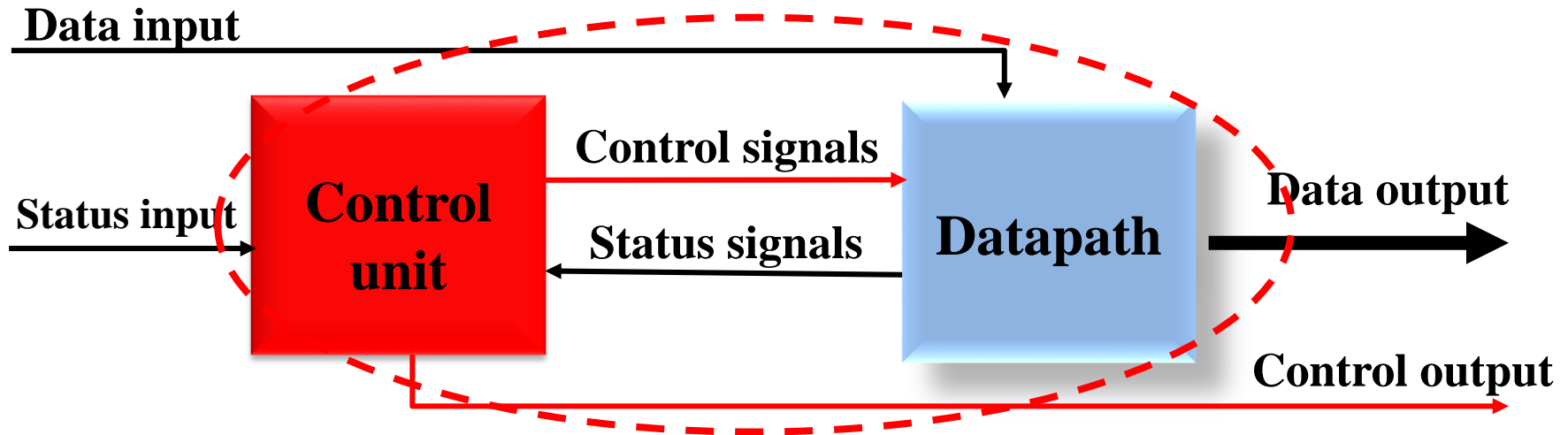
Course Outline



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



MIPS中断结构

□ 协处理器CP0

■ MIPS用来辅助的部件

- 处理异常
- 存储器管理
- 系统配置
- 其他片上功能

□ 常用CP0寄存器

寄存器	编号	作用
BadVAddr	8	最近内存访问异常的地址
Count	9	高精度内部计时计数器
Compare	11	定时常数匹配比较寄存器
Status(SR)	12	状态寄存器、特权、中断屏蔽及使能等，可位控
Cause	13	中断异常类型及中断持起位
EPC	14	中断返回地址
Config	16	配置寄存器，依赖于具体系统



CPO传输指令

□ 控制寄存器访问指令

■ 读CP0指令mfco

□ mfco rt,rd: $GPR[rt] \leq CP0[rd]$

Op=6bit	rs=00000	rt=5bit	Rd=5bit	=11个0
0x10	0	rt	rd	00000 000000

□ 写CP0mtco指令

■ mtc0 rd, rt: $GPR[rd] \leq CP0[rt]$

Op=6bit	rs=00000	rt=5bit	Rd=5bit	=11位0
0x10	4	rt	rd	00000 000000



中断相关指令

□ 异常返回

■ eret

- $PC \leq EPC$; (CP0的Cause和Status寄存器有变化)

Op=6bit 1 19bit FUN

0x0	1	000 0000 0000 0000 0000	011000
-----	---	-------------------------	--------

□ 系统调用

■ syscall

- $EPC = PC + 4$; $PC \leq$ 异常处理地址; Cause和Status寄存器有变化

■ 参数:

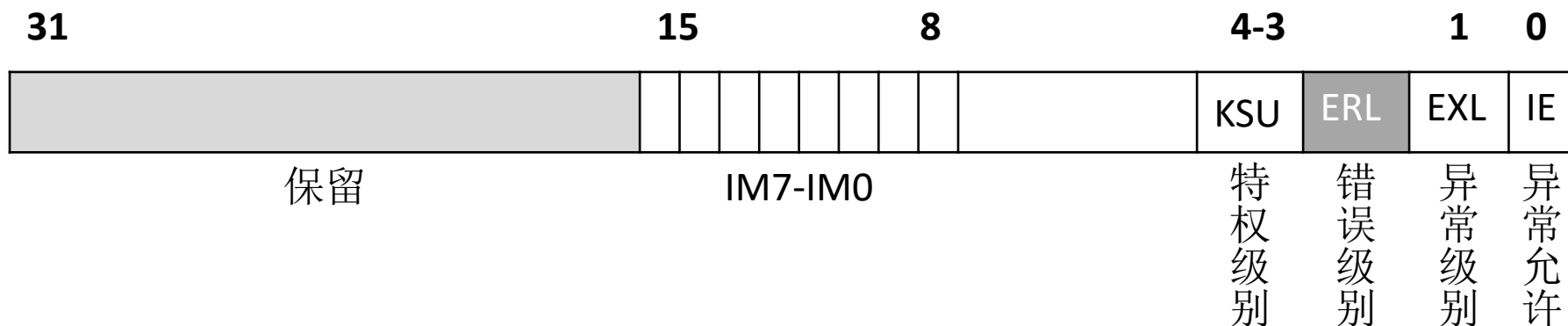
- \$v0=系统调用号: Fig B-9-1
- \$a0~\$a3、\$f12, 返回在\$v0

Op=6bit 20bit FUN

0x0	0000 0000 0000 0000 0000	0011000
-----	--------------------------	---------



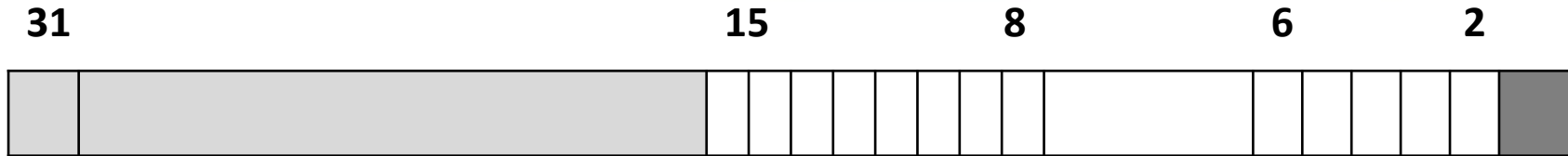
状态寄存器: Status(SR)



- **IE:** IE=1全局中断使能
- **EXL:** 异常设置, 强制CPU进入内核模式并关闭中断, 优先级高于IE
- **ERL:** 数据错误设置, 进入内核模式并关闭中断, 优先级高于IE
- **IM7-0:** 8个中断屏蔽位, 6个外中断, 2个软中断。没有优先级
- **KSU:** 00=内核态、01=监管模式、10=用户态。优先级低于EXL和ERL



Cause寄存器



分支延时

保留

中断挂起位
IP0~IP7

异常编码
ExcCode

ExcCode	名称	异常产生原因
0	Int	外中断（硬件）
4	AdEL	地址错误异常（L/S）
5	AdES	地址错误异常（存储）
6	IBE	取指令的总线错误
7	DBE	L/S的总线错误
8	Sys	系统调用异常
9	Bp	断点异常
10	RI	非法指令异常
11	CpU	没有实现的协处理器
12	Ov	算术上溢异常
13	Tr	陷阱
15	FPE	浮点



MIPS中断响应

□ 初始化

- 设置SR: 关中断 $IE=0$, $KSU\ ERL\ EXL=00\ 0\ 0$
- 系统初始化
- 设置SR: 开中断 $IE=1$, 设置IM7-0

□ 中断响应

- 硬件保存断点: $EPC \leftarrow PC+4$
- 硬件修改PC: $PC \leftarrow$ 向量、硬件关中断
- 中断服务: 保护寄存器、开中断*、服务、恢复寄存器
- 开中断*
- 返回: `eret` (硬件修改Cause和Status寄存器)



中断数据通路和控制器

□ 数据通路

- 需要增加CP0的Cause和Status寄存器
- 必须增加EPC寄存器及 $EPC \leftarrow PC + 4$ 通路
- 增加mfc0、 mtc0和eret指令通道

□ 控制器

- 增加中断检测电路
- 增加异常检测电路
- 中断响应控制
- CP0传输控制



典型处理器中断结构

□ Intel x86中断结构

- 中断向量：000~3FF，占内存最底1KB空间
 - 每个向量由二个16位生成20位中断地址
 - 共256个中断向量，向量编号n=0~255
 - 分硬中断和软中断，响应过程类同，触发方式不同
 - 硬中断响应由控制芯片8259产生中断号n(接口原理课深入学习)

□ ARM中断结构

- 固定向量方式(嵌入式课程深入学习)

异常类型	偏移地址(低)	偏移地址(高)	
复位	00000000	FFFF0000	
未定义指令	00000004	FFFF0004	
软中断	00000008	FFFF0008	
预取指令终	0000000C	FFFF000C	
数据终止	00000010	FFFF0010	
保留	00000014	FFFF0014	
中断请求(IRQ)	00000018	FFFF0018	
快速中断请求(FIQ)	0000001C	FFFF001C	

Course Outline





简化中断设计：ARM模式

□ ARM中断向量表

向量地址	ARM异常名称	ARM系统工作模式	本实验定义
0x0000000	复位	超级用户Svc	内核模式
0x0000004	未定义指令终止	未定义指令终止Und	RI内核模式
0x0000008	软中断（SWI）	超级用户Svc	Sys系统调用
0x000000c	Prefetch abort	指令预取终止Abt	Reserved自定义
0x0000010	Data abort	数据访问终止Abt	Ov
0x0000014	Reserved	Reserved	Reserved自定义
0x0000018	IRQ	外部中断模式IRQ	Int外中断（硬件）
0x000001C	FIQ	快速中断模式FIQ	Reserved自定义

□ 简化中断设计

- 采用ARM中断向量(不兼容MIPS)
 - 实现非法指令异常和外中断
 - 设计EPC
- 半兼容MIPS*
 - Cause和Status寄存器
 - 设计mfc0、mtc0指令

设计方案参考：DataPath

◎ DataPath修改

⌚ CPU复位时IE=0, EPC=PC=0x00000000

⊙ IE=中断使能(重要)

⌚ 修改PC模块增加(ARM模式)

⊙ EPC寄存器, INT触发PC转向中断地址

◆ 相当于硬件触发Jal, 用eret返回

⊙ 增加控制信号INT、RFE/eret

◆ INT宽度根据扩展的外中断数量设定

⌚ 修改DataPath增加(MIPS模式)*

⊙ RI、Ov、EI

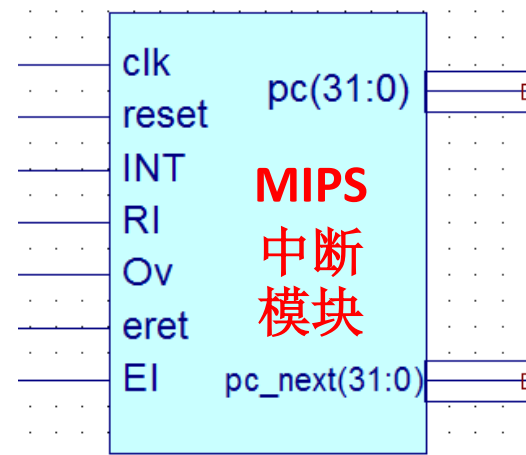
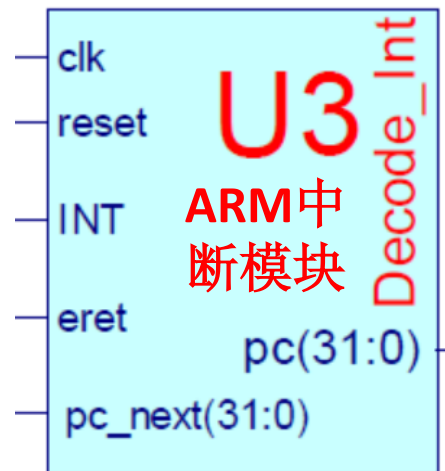
⊙ Cause和Status

⊙ 增加Cause和Status通道

⊙ 增加Wt_Data通道

⊙ 增加控制信号CP0_Write

注意：INT是电平信号，不要重复响应





设计方案参考：控制器

◎ 控制器修改

☞ ARM模式(简单)

- ◎ 仅增加eret指令
- ◎ 中断请求信号触发PC转向，在Datapath模块中修改

☞ MIPS模式(可独立模块)*

- ◎ 扩展mfc0、mtc0指令译码
- ◎ 增加Wt_Write通道选择控制
- ◎ 增加CP0_Write
- ◎ 增加控制信号eret、RI、CP0_Write

◎ 中断调试

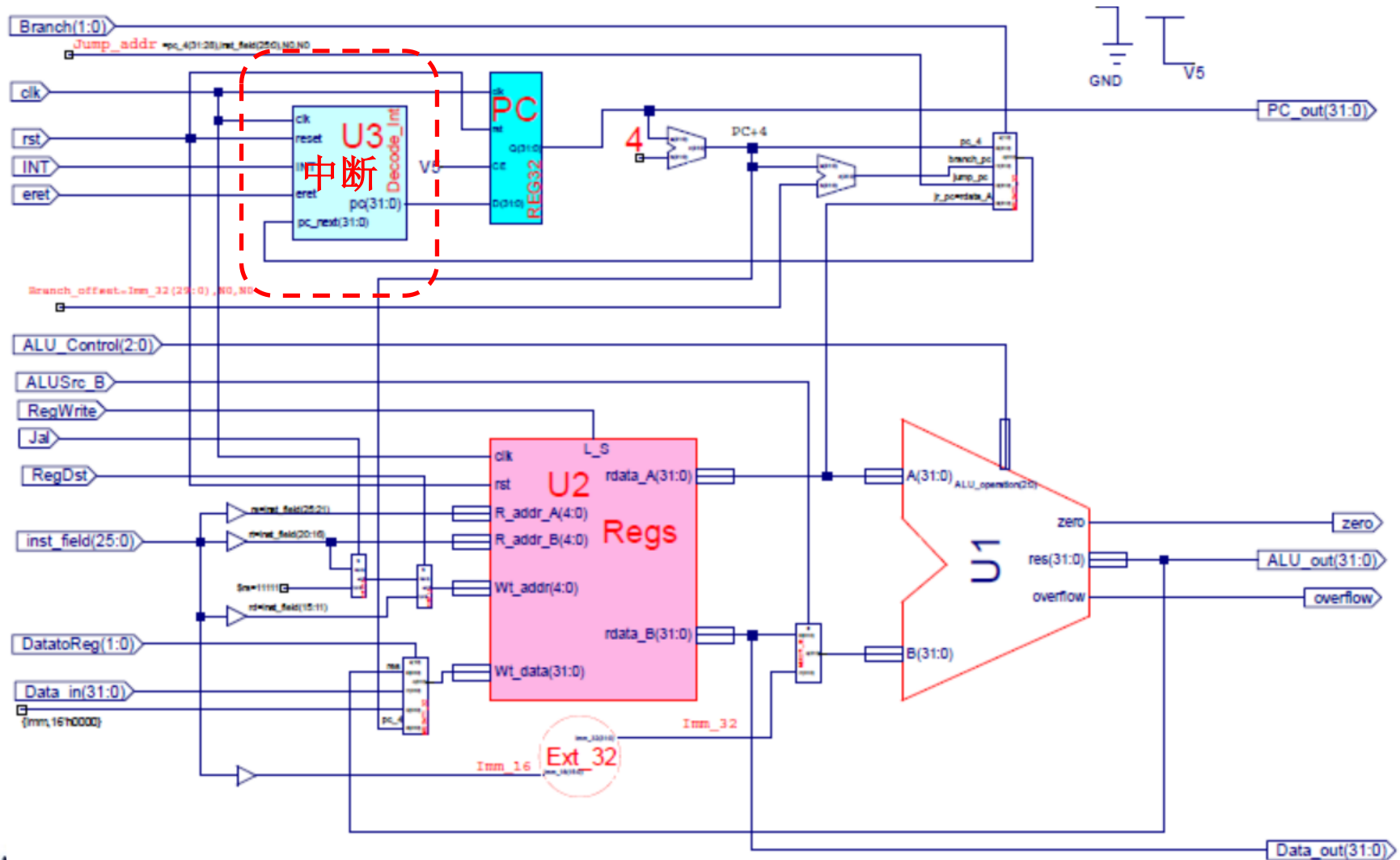
☞ 首先时序仿真

☞ 物理验证

- ◎ 用BTN[0]触发调试：静态或低速
- ◎ 用计数器counter1_OUT调试：动态或高速

◆ 动态时注意死锁

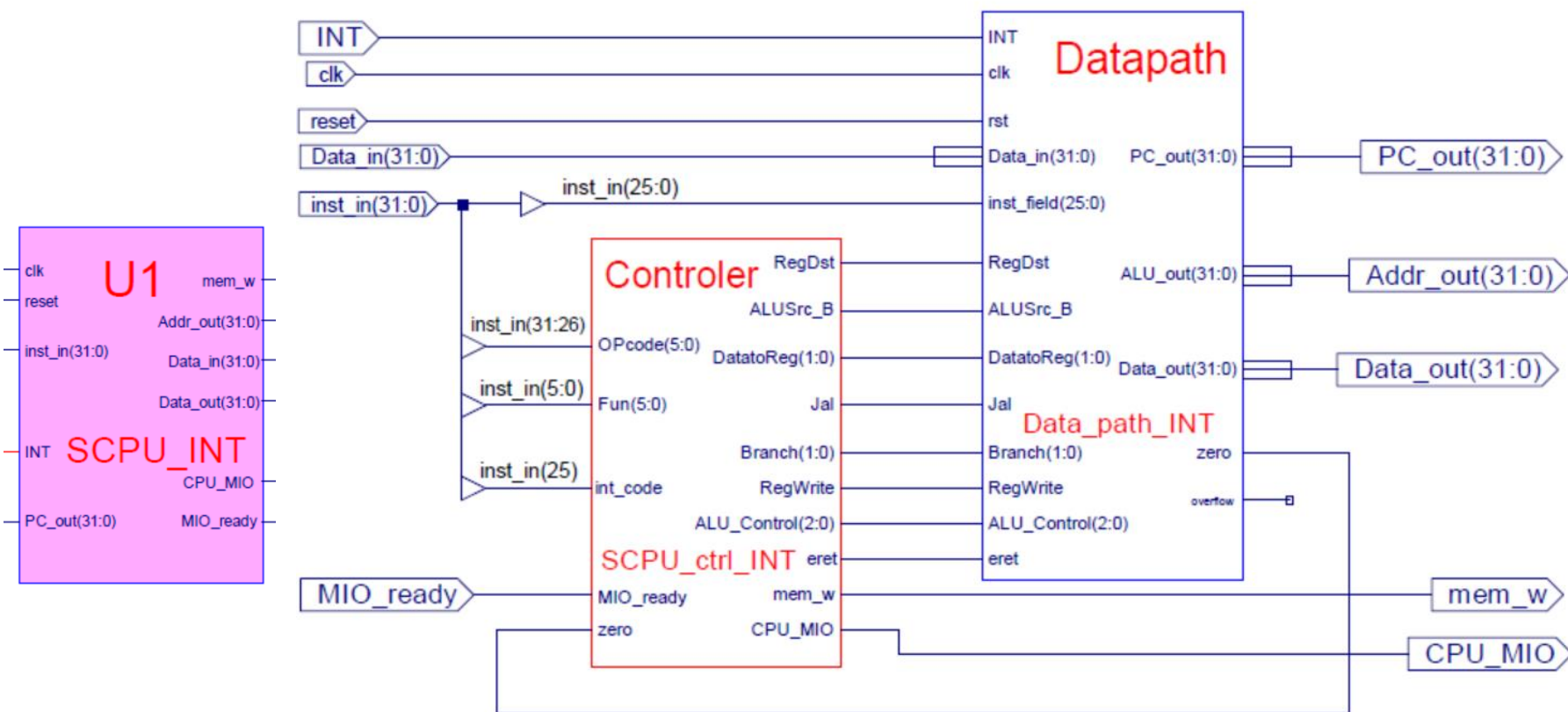
增加简单中断后的DataPath



增加中断后的CPU模块

◎ 注意修改模块逻辑符号

Ⓔ SCPU_INT.sym、SCPU_ctrl_INT.sym和Data_path_INT.sym





思考题



● END