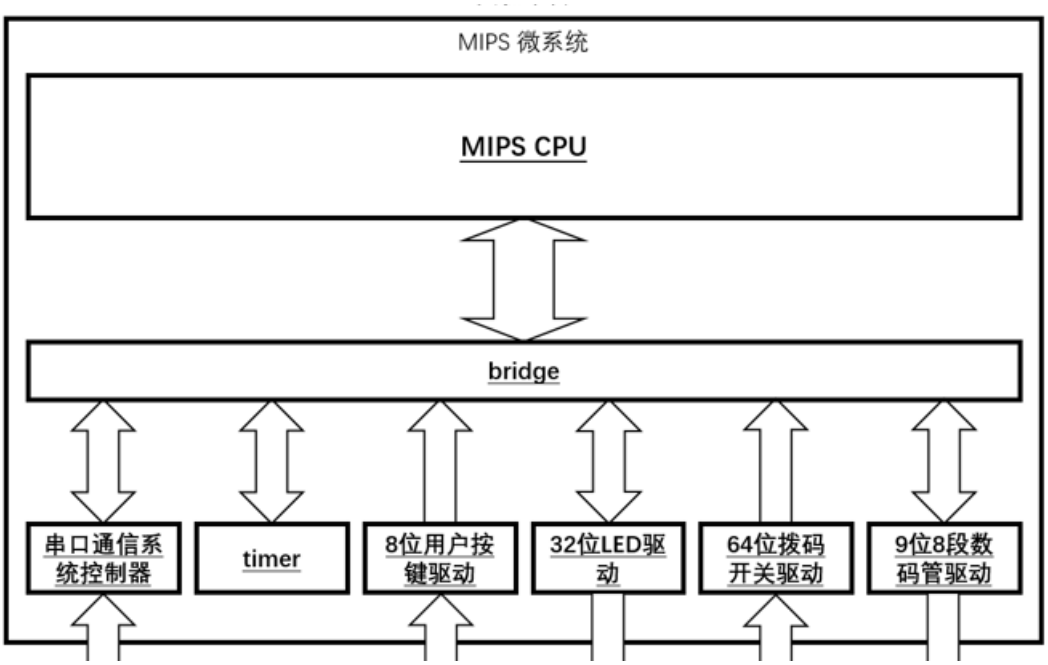
P8 实验设计报告

18373085 张海渝

一. 数据通路

数据通路总图：



因为在 P8 的实验中，需要实现外设和 CPU 的交互，所以在之前 CPU 的数据通路上首先是多了一个桥（Bridge）的设计，通过桥实现了一系列的数据交互以及中断功能

1.Bridge 设计

信号设计：

序号	功能名称	方向	功能描述
1	PrAddr[31:0]	I	写或读外部数据的地址
2	Dout0[31:0]	I	Timer0 中的数据值（对应于 PrAddr）
3	Dout1[31:0]	I	64 位拨码开关所对应的值（对应于 PrAddr）
4	PrWe	I	是否写外设的信号
5	WE0	O	是否写 Timer0 的信号
6	WE1	O	是否写对应数码管的显示的信号
7	PrRD[31:0]	O	输出值 cpu 的值
8	WE3	0	是否写 UART 的信号
9	Dout2[31:0]	I	对应于 user_key 的值
10	Dout3[31:0]	I	对应于 UART 中寄存器的值

解释：

1. 需要注意在将 PrRD 传回的时候需要注意不要将 DM 与这个值混淆,所以可以加上一个 HITDM 信号来区分
2. 对于 WE 来说，有几个外部设备就有几个 WE。
3. 产生信号都是组合逻辑，见下图：

```
module Bridge(PrAddr,Dout0,Dout1,,Dout2,PrWe,WE1,WE0,PrRD,Dout3,WE3);
    input [31:0] PrAddr;
    input [31:0] Dout0,Dout2,Dout1,Dout3;
    input PrWe;
    output WE1,WE0,WE3;
    output [31:0] PrRD;

    wire [31:0] addr;
    assign addr={PrAddr[31:2],2'b0};
    assign PrRD=    addr>=32'h7f00 && addr<=32'h7f0B ? Dout0 :
                    addr>=32'h7f2c && addr<=32'h7f33 ? Dout1 :
                    addr>=32'h7f40 && addr<=32'h7f43 ? Dout2 :
                    addr>=32'h7f10 && addr<=32'h7f2B ? Dout3 : 0 ;
    assign WE0=addr>=32'h7f00 && addr<=32'h7f0B&&PrWe?1:0;
    assign WE1=addr>=32'h7f44 && addr<=32'h7f47&&PrWe?1:0;
    assign WE3=addr>=32'h7f10 && addr<=32'h7f2B&&PrWe?1:0;

endmodule
```

图一：Bridge 模块图

二．CP0 设计

信号设计：

序号	功能名称	方向	功能描述
1	A1[4:0]	I	写入或者写出的寄存器编号
2	Din[31:0]	I	写入 CP0 寄存器中的数值
3	ExcCode[5:0]	I	异常编号(下面会进行具体的说明)
4	HWInt[5:0]	I	外部的中断信号
5	BD	I	判断当前指令是否为延迟槽中的指令
6	WE	I	写使能
7	clk	I	时钟信号
8	reset	I	复位信号
9	EXLclr	I	将 SR 中的 EXL 为复位
10	IntReq	O	是否中断异常信号
11	EPCout[31:0]	O	CP0 中的 EPC 的值
12	Dout	O	对应于 A1 编号的寄存器

解释：

1. CP0 中共有 4 个寄存器，分别为 SR，Cause，EPC，PRID（只需考虑有效位，其余为置为 0）

   SR：SR[15:10]：为允许外部中断位，
   SR[1]：当进入中断或者异常的状态时，置为 1
   SR[0]:全局中断使能，是否允许中断

   Cause: Cause[6:2]为异常类型信号
   Cause[31]为 BD 位
   Cause[15:10]为中断信号位

   EPC: 中断异常时的指令 PC 值（在 BD 为 1 的时候为其前序指令）

   PRID：随意

2. 首先考虑处理中断异常的时候，可知
IntReq=(|(HWInt[5:0]&SR[15:10])&SR[0]&~SR[1])|(ExcCode[5])
当发生中断或者异常的时候需要
3. WE 是在 D 级中的控制器中产生信号传至 M 级（因为 CP0 在 M 级）。
4. 在中断异常信号（IntReq）置位的时候，需要将 EPC 传为当前 PC 值，并且需要考虑 BD 位，BD 为 1 需要将 EPC-4
5. 需要注意在中断异常的时候需要将进入 pc 值为 0x4180 的指令。

三．测试程序

1. 测试 cpu：

1.R 型指令（addu 为例，乘除包含 mflo，mfhi，slt 等类型指令）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-M-RS	subu	E	rs	subu $1,$2,$3 addu $4,$1,$2	转发
R-M-RT	subu	E	rt	subu $1,$2,$3 addu $4,$2,$1	转发

R-W-RS	subu	E	rs	subu $1,$2,$3
				nop
				addu $4,$1,$2
					转发

R-W-RT	subu	E	rt	subu $1,$2,$3
				nop
				addu $4,$2,$1
					转发

I-M-RS	ori	E	rs	ori $1,$0,0x1111 addu $3,$1,$2	转发
I-M-RT	ori	E	rt	ori $1,$0,0x1111 addu $3,$2,$1	转发
I-W-RS	ori	E	rs	ori $1,$0,0x1111 nop addu $3,$1,$2	转发
I-W-RT	ori	E	rt	ori $1,$0,0x1111 nop addu $3,$2,$1	转发

LD-W-RS	lw	E	rs	lw $1,0($2)
				nop
				addu $3,$1,$2
					转发

LD-W-RT	lw	E	rt	lw $1,0($2)
				nop
				addu $3,$2,$1
					转发

LD-M-RS	lw	E	rs	lw $1,0($2) addu $3,$1,$2	暂停

LD-M-RT	lw	E	rt	lw $1,0($2) addu $3,$1,$2	暂停
LD-E-RS	lw	D	rs	lw $1,0($2) addu $3,$1,$2	暂停
LD-E-RT	lw	D	rt	lw $1,0($2) addu $3,$1,$2	暂停
JAL-M-RS	jal	D	rs	jal dfs nop dfs: addu $1,$31,$0	转发
JAL-M-RT	jal	D	rt	jal dfs nop dfs: addu $1,$0,$31	转发
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop addu $1,$31,$0	转发
JAL-W-RT	jal	D	rt	jal dfs nop dfs: nop addu $1,$0,$31	转发

2.I 型指令（ori 为例，其中包含了 sll 等指令的测试）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-M-RS	subu	E	rs	subu $1,$2,$3 ori $4,$1,0x1111	转发
R-W-RS	subu	E	rs	subu $1,$2,$3 nop ori $4,$1,0x1111	转发
I-M-RS	ori	E	rs	ori $1,$0,0x1110 ori $2,$1,0x1111	转发
I-W-RS	ori	E	rs	ori $1,$0,0x1110 nop ori $3,$1,0x1111	转发
LD-W-RS	lw	E	rs	lw $1,0($2) nop ori $3,$1,0x0	转发
LD-M-RS	lw	D	rs	lw $1,0($2) nop ori $3,$1,$2	暂停

LD-E-RS	lw	D	rs	lw $1,0($2) ori $3,$1,$2	暂停
JAL-M-RS	jal	D	rs	jal dfs nop dfs: ori $1,$31,0x0	转发
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop ori $1,$31,0x0	转发

3.LD 型指令（lw 为例）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-M-RS	subu	E	rs	subu $1,$2,$3 lw $4,0($1)	转发

R-W-RS	subu	E	rs	subu $1,$2,$3
				nop
				lw $4,0($1)
					转发

I-M-RS	ori	E	rs	ori $1,$0,0x1110 lw $4,0($1)	转发
I-W-RS	ori	E	rs	ori $1,$0,0x1110 nop lw $4,0($1)	转发

LD-W-RS	lw	E	rs	lw $1,0($2)
				nop
				lw $4,0($1)
					转发

LD-M-RS	lw	D	rs	lw $1,0($2)
				nop
				lw $4,0($1)
					转发

LD-E-RS	lw	D	rs	lw $1,0($2) lw $4,0($1)	暂停
JAL-M-RS	jal	D	rs	jal dfs nop dfs: lw $4,0($31)	转发
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop lw $4,0($31)	转发

4.Store 型指令（sw 为例）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法

R-M-RS	subu	E	rs	subu $1,$2,$3 sw $4,0($1)	转发
R-M-RT	subu	E	rt	subu $1,$2,$3 sw $1,0($0)	转发

R-W-RS	subu	E	rs	subu $1,$2,$3
				nop
				sw $4,0($1)
					转发

R-W-RS	subu	E	rt	subu $1,$2,$3
				nop
				sw $1,0($0)
					转发

I-M-RS	ori	E	rs	ori $1,$0,0x1110 sw $4,0($1)	转发
I-W-RS	ori	E	rs	ori $1,$0,0x1110 nop sw $4,0($1)	转发
I-M-RT	ori	E	rt	ori $1,$0,0x1110 sw $1,0($0)	转发
I-W-RT	ori	E	rt	ori $1,$0,0x1110 nop sw $1,0($0)	转发

LD-W-RS	lw	E	rs	lw $1,0($2)
				nop
				sw $4,0($1)
					转发

LD-W-RT	lw	E	rt	lw $1,0($2)
				nop
				sw $1,0($0)
					转发

LD-M-RS	lw	D	rs	lw $1,0($2)
				nop
				sw $4,0($1)
					转发

LD-M-RT	lw	D	rt	lw $1,0($2)
				nop
				sw $1,0($0)
					转发

LD-E-RS	lw	D	rs	lw $1,0($2) sw $4,0($1)	暂停
LD-E-RT	lw	D	rt	lw $1,0($2) sw $1,0($0)	转发
JAL-M-RS	jal	D	rs	jal dfs nop dfs: sw $4,0($31)	转发
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop	转发

				sw $4,0($31)	

5.B 型指令（beq 为例）：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-E-RS	subu	D	rs	subu $1,$2,$3 beq $1,$2,loop	暂停
R-E-RT	subu	D	rt	subu $1,$2,$3 beq $2,$1,loop	暂停
R-M-RS	subu	D	rs	subu $1,$2,$3 nop beq $1,$2,loop	转发
R-M-RT	subu	D	rt	subu $1,$2,$3 nop beq $2,$1,loop	转发
I-E-RS	ori	D	rs	ori $1,$2,0x1111 beq $1,$2,loop	暂停
I-E-RT	ori	D	rt	ori $1,$2,0x1111 beq $2,$1,loop	暂停
I-M-RS	ori	D	rs	ori $1,$2,0x1111 nop beq $1,$2,loop	转发
I-M-RT	ori	D	rt	ori $1,$2,0x1111 nop beq $2,$1,loop	转发
LD-E-RS	lw	D	rs	lw $1,0($2) beq $1,$2,loop	暂停
LD-E-RT	lw	D	rt	lw $1,0($2) beq $2,$1,loop	暂停
LD-M-RS	lw	D	rs	lw $1,0($2) nop beq $1,$2,loop	暂停
LD-M-RT	lw	D	rt	lw $1,0($2) nop beq $2,$1,loop	暂停
LD-W-RS	lw	D	rs	lw $1,0($2) nop nop beq $1,$2,loop	转发
LD-W-RT	lw	D	rt	lw $1,0($2) nop nop beq $2,$1,loop	转发
JAL-M-RS	jal	D	rs	jal dfs nop	暂停

				dfs: beq $4,$31,loop	
JAL-M-RT	jal	D	rt	jal dfs nop dfs: beq $31,$4,loop	暂停
JAL-W-RS	jal	D	rs	jal dfs nop dfs: nop beq $4,$31,loop	转发
JAL-W-RT	jal	D	rt	jal dfs nop dfs: nop beq $31,$4,loop	转发

6.jr 指令：

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-E-RS	subu	D	rs	subu $1,$2,$3 jr $1	暂停
R-M-RS	subu	D	rs	subu $1,$2,$3 nop jr $1	转发
I-E-RS	ori	D	rs	ori $1,$2,0x1111 jr $1	暂停
I-M-RS	ori	D	rs	ori $1,$2,0x1111 nop jr $1	转发
LD-E-RS	lw	D	rs	lw $1,0($2) jr $1	暂停
LD-M-RS	lw	D	rs	lw $1,0($2) nop jr $1	暂停
LD-W-RS	lw	D	rs	lw $1,0($2) nop nop jr $1	转发
JAL-M-RS	jal	D	rs	jal dfs nop dfs: jr $31	暂停
JAL-W-RS	jal	D	rs	jal dfs nop	转发

				dfs: nop jr $31	

7.乘除指令（以 mult 为例）

测试类型	前序指令	冲突位置	冲突寄存器	测试序列	解决方法
R-E-RS	mfhi	D	HI	mfhi $1 mult $2,$3	暂停

2. 测试异常：

1. AdEL

```
.text
ori  $2,$0,0x1c01 #允许所有的中断
mtc0  $2,$12
jal  loop
nop
nop  #  jr  $ra

loop:
addi  $ra,$ra,-1 #或者改为  ori  $ra,$0,0x4fff
jr  $ra
nop

.ktext  0x00004180
ori  $26,0x3010
mtc0  $26,$14
eret
ori  $1,$0,0x1111
```

2. AdEL,AdEs（相当于 lw，sw 的区别）

```
.text
ori  $2,$0,0x1c01 #允许所有的中断
mtc0  $2,$12
ori  $3,0x7f00
ori  $4,0x0003
lw  $3,0($4)#  可以将 lw 换为 lh，lhu
                #或者将此指令换为 lh(lhu,lb,lbu)  $4,0($3)
                #或者改为 sw  $4,8($3)
nop
nop
nop
.ktext  0x00004180
ori  $3,$0,0x0004
```

```
ori  $4,$0,0x0004
eret
ori  $1,$0,0x1111
```

3. RI

```
.text
ori  $2,$0,0x1c01  #允许所有的中断
mtc0  $2,$12
ori  $3,0x7f00
ori  $4,0x0003
msub  $3,$4
nop
nop
nop
.ktext  0x00004180
ori  $3,$0,0x0004
ori  $4,$0,0x0004
eret
ori  $1,$0,0x1111
```

4. Ov

```
.text
ori  $2,$0,0x1c01  #允许所有的中断
mtc0  $2,$12
lui  $3,0x7fff
lui  $4,0x7fff
add  $5,$3,$4

nop
nop
nop

.ktext  0x00004180
ori  $3,$0,0x0004
ori  $4,$0,0x0004
eret
ori  $1,$0,0x1111
```

3.测试中断

1.测试 Timer(以 Timer0 为例)

```
.text
ori  $4,$0,0x0401
mtc0  $4,$12
ori  $1,$0,0x7f00
ori  $2,$0,0x0009
ori  $3,$0,100
```

```
sw  $2,0($1)  #timer0 中的 ctrl 寄存器
sw  $3,4($1)  #timer0 中的 present 寄存器

loop:
j  loop
nop

.ktext  0x00004180
ori  $26,$0,0x7f00
lw  $27,0($26)
ori  $27,$27,1
sw  $27,0($26)
lw  $27,4($26)
ori  $27,$0,1000  #改变 present 的值
sw  $27,4($26)
eret
nop
```

3. 中断异常同时发生

```
.text
ori  $2,$0,0x1c01  #允许所有的中断
mtc0  $2,$12
lui  $4,0x7fff
lui  $5,0x7fff
ori  $1,$0,0x7f00
ori  $2,$0,0x0009
ori  $3,$0,10
sw  $2,0($1)  #timer0 中的 ctrl 寄存器
sw  $3,4($1)  #timer0 中的 present 寄存器
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
add  $6,$4,$5


.ktext  0x00004180
```

```
ori  $26,$0,0x7f00
lw  $27,0($26)
ori  $27,$27,1
sw  $27,0($26)
eret
nop
```

四．思考题

1. 请查阅相关资料，说一说什么是「FPGA 技术」？它有哪些好处和缺陷?

答：FPGA（Field Programmable Gate Array）是在 PAL、GAL 等可编程器件的基础上进一步发展的产物。它是作为专用集成电路（ASIC）领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。

FPGA 的优点如下：

(1) FPGA 由逻辑单元、RAM、乘法器等硬件资源组成，通过将这些硬件资源合理组织，可实现乘法器、寄存器、地址发生器等硬件电路。

(2) FPGA 可通过使用框图或者 Verilog HDL 来设计，从简单的门电路到 FIR 或者 FFT 电路。

(3) FPGA 可无限地重新编程，加载一个新的设计方案只需几百毫秒，利用重配置可以减少硬件的开销。

(4) FPGA 的工作频率由 FPGA 芯片以及设计决定，可以通过修改设计或者更换更快的芯片来达到某些苛刻的要求（当然，工作频率也不是无限制的可以提高，而是受当前的 IC 工艺等因素制约）。

FPGA 的缺点如下：

(1) FPGA 的所有功能均依靠硬件实现，无法实现分支条件跳转等操作。

(2) FPGA 只能实现定点运算。

总结：FPGA 依靠硬件来实现所有的功能，速度上可以和专用芯片相比，但设计的灵活度与通用处理器相比有很大的差距。

2.在上述步骤中，同学们可能会出现各种各样的问题，例如综合失败、无法布局布线等，或者也有同学会尝试消除所有的 Warning。无论是何种情况，希望同学们能记录下自己的问题和解决的过程，并体现自己对实验的理解（例如对 FPGA 的理解，对 Verilog 语法可综合性的理解等）。

答：从最开始的 P7 开始，我对 P7 进行改进，第一步我先删除了乘除模块，在删除的时候需要知道在 CPU 的数据通路中应该删除那些相关信号，我第一次进行删除的时候出现了删错的情况于是只能重新进行删除。

当我将其删除以后，我进行了第一次综合，但是因为在 always 块中的赋值错误，于是需要重新进行赋值，这也是在综合时最需要注意的一点，然后去掉 initial 块。

在调试的时候需要注意引脚的连接以及需要注意在自己的 CPU 设计中 1 是不是对应于输入中的 1.

3. 简述你的中断实现方案。

答：通过外部是否传输完的信号 RS 进行中断。