P7流水线CPU设计文档

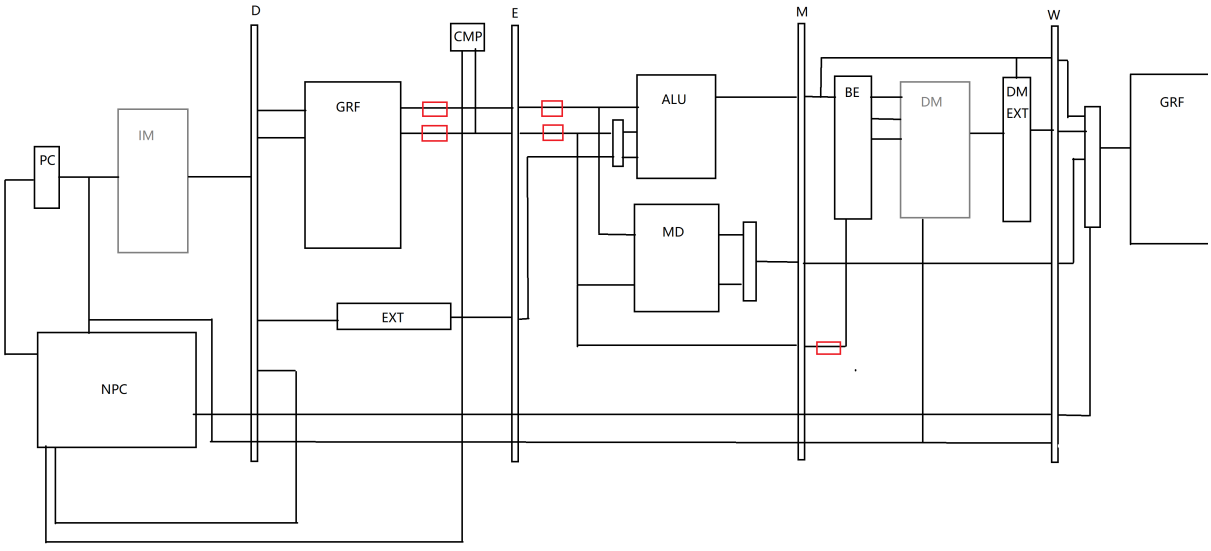
（一）架构设计

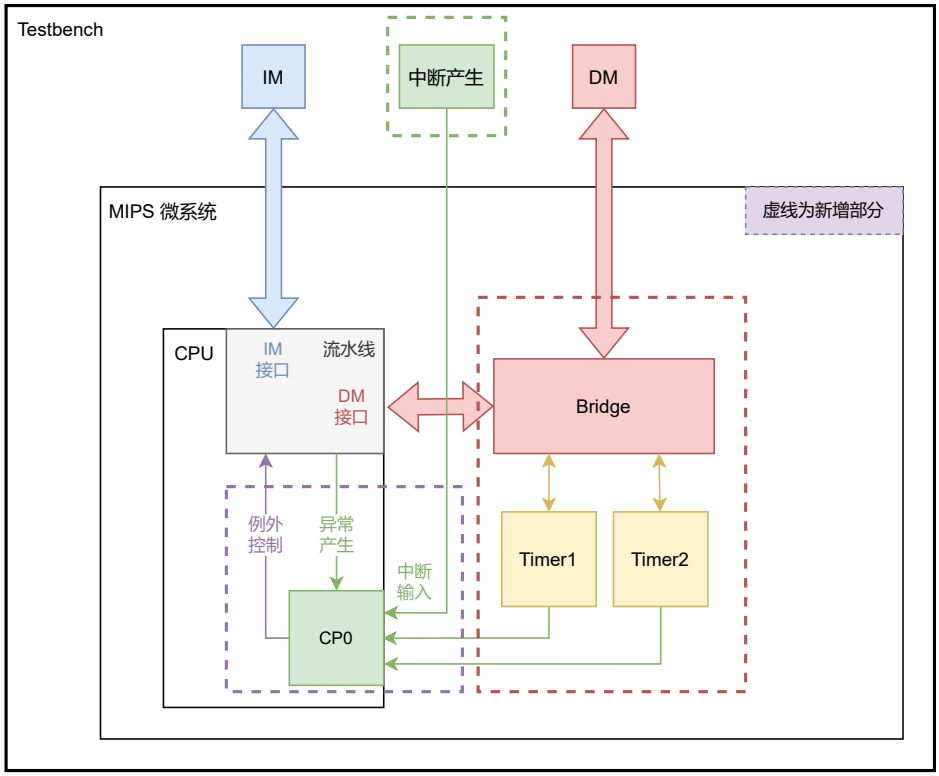
设计说明

- 支持的指令集：**{ LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、 SUB、 SUBU、 MULT、MULTU、 DIV、 DIVU、 SLL、 SRL、 SRA、 SLLV、 SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、 XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、 BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO,`mfc0,mtc0,eret`}**
- IM：容量为20KB (32bit/word x 5Kword)。
- DM：容量为12KB (32bit/word x 3Kword)。
- 需有单独的**乘除法模块**和**数据扩展模块**

指令类

- 单寄存器计算：`addi, addiu, slti, sltiu, andi, ori, xori, sll, srl, sra`
- 双寄存器计算：`add, addu, sub, subu, slt, sltu, and, or, nor, xor, sllv,srlv, srav`
- 双寄存器分支：`beq, bne`
- 单寄存器分支：`bgez, bgtz, blez, bltz`
- 写内存：`sw, sh, sb`
- 读内存：`lw, lh, lhu, lb, lbu`
- 读乘除法寄存器：`mfhi, mflo`
- 写乘除法寄存器：`mthi, mtlo`
- 计算乘除法；`mult, multu, div, divu`
- 跳转并链接：`jal`
- 跳转寄存器：`jr`
- 跳转寄存器并链接：`jalr`
- 跳转`j`
- 加载高位：`lui`
- `mfc0,mtc0,eret`





（二）关键模块概述

MIPS(顶层模块)

端口定义

名称	位宽	方向	描述
clk	1	I	时钟信号
reset	1	I	同步复位信号
interrupt	1	I	外部中断信号
i_inst_rdata	32	I	i_inst_addr 对应的 32 位指令
m_data_rdata	32	I	m_data_addr 对应的 32 位数据
macroscopic_pc	32	O	宏观PC（M级PC）
i_inst_addr	32	O	需要进行取指操作的流水级 PC（一般为 F 级）
m_data_addr	32	O	数据存储器待写入地址
m_data_wdata	32	O	数据存储器待写入数据
m_data_byteen	4	O	字节使能信号
m_inst_addr	32	O	M 级 PC
w_grf_we	1	O	grf 写使能信号
w_grf_addr	5	O	grf 中待写入寄存器编号
w_grf_wdata	32	O	grf 中待写入数据
w_inst_addr	32	O	W 级 PC

Bridge(系统桥)

- 存储器与计时器通过系统桥模块与CPU相连

模块功能定义

- 输出地址
- 地址匹配：为每个设备产生一个译码信号
- CPU读数据
- CPU写数据

模块接口定义

名称	位宽	方向	描述
CPU侧：			
PrAddr	32	I	CPU输出地址
PrWE	4	I	CPU输出字节写使能
PrWD	32	I	CPU输出写数据
PrRD	32	O	CPU读数据
设备侧：			
DEVAddr	32	O	设备读写地址
DEVWD	32	O	设备写数据
DMWE	4	O	DM写使能
Timer0WE	1	O	Timer0写使能
Timer1WE	1	O	Timer1写使能
DMData	32	I	DM输出数据
Timer0Data	32	I	Timer0输出数据
Timer1Data	32	I	Timer1输出数据

地址映射

	地址或地址范围	备注
数据存储器	0x0000_0000 至 0x0000_2FFF	
指令存储器	0x0000_3000 至 0x0000_6FFF	
PC 初始值	0x0000_3000	
Exception Handler 入口地址	0x0000_4180	
计时器 0 寄存器地址	0x0000_7F00 至 0x0000_7F0B	计时器 0 的 3 个寄存器 0x0000_7F00:ctrl 0x0000_7F04:preset 0x0000_7F08:count
计时器 1 寄存器地址	0x0000_7F10 至 0x0000_7F1B	计时器 1 的 3 个寄存器 0x0000_7F10:ctrl 0x0000_7F14:preset 0x0000_7F18:count

Timer(计时器)

- 根据设定的时间来产生中断信号

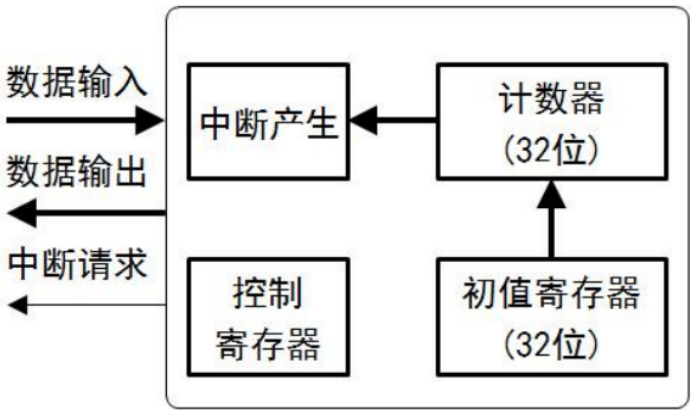


图 1-1 Timer/Counter 内部基本结构

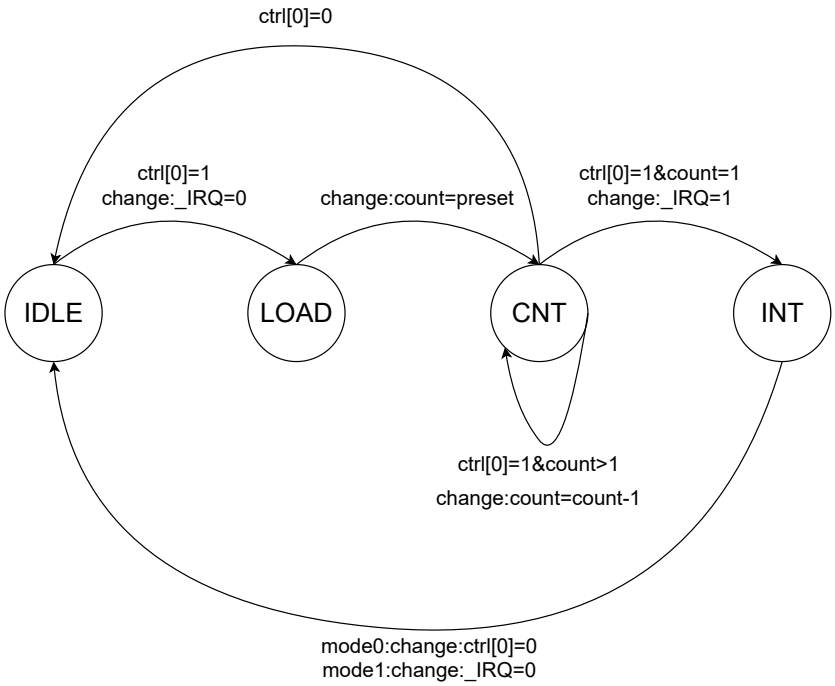
模块接口定义

信号名	位宽	方向	描述
clk	1	I	时钟
reset	1	I	复位信号
Addr	**[31:2]**	I	地址输入
WE	1	I	写使能
Din	32	I	数据输入
Dout	32	O	数据输出
IRQ	1	O	中断请求

模块行为描述

- 控制寄存器决定该计数起停控制等。
- 初值寄存器为 32 位计数器提供初始值。
- 根据不同的计数模式，在计数为 0 后，计数器或者自动装填初值并重新倒计数，或者保持在 0 值直至计数器使能再次被设置为 1。
- 使用 store 类指令修改 TC 寄存器值的优先级高于 TC 自修改的优先级。
- 当计数器计数时，若计数器使能被 store 类指令修改为 0 则停止计数。
- 当计数器工作在模式 0 并且在中断允许的前提下，当计数器计数值为 0 时，中断产生逻辑产生中断请求(IRQ 为 1)。

时序图



模式0

- 通常用于定时产生中断

模式1

- 通常用于产生周期性脉冲

控制寄存器（CTRL）

- 当读取 CTRL 寄存器时，未定义位始终为 0；当写入 CTRL 寄存器时，未定义位被忽略。
- 可读可写

名称	位	描述
reserved	31:4	保留
IM	3	中断屏蔽：0禁止中断，1允许中断
Mode	2:1	模式选择：00：方式0,01：方式1
enable	0	计数器使能：0：停止计数，1：允许计数

初值寄存器（PRESET）

- 可读可写

计数值寄存器（COUNT）

- 只读

计时器使用说明

第一次使用计时器时，用户应当首先通过两个时钟周期为计时器的 `ctrl` 寄存器和 `preset` 寄存器赋初值。

赋初值操作：

- 将WE置为1
- 将Addr低2位设为0，此时为 `ctrl` 寄存器赋初值。
- 将Addr低2位设为01，此时为 `preset` 寄存器赋初值。

计时操作：

-

CPU

端口定义

名称	位宽	方向	描述
clk	1	I	时钟信号
reset	1	I	同步复位信号
HWInt	6	I	外部中断信号
i_inst_rdata	32	I	i_inst_addr 对应的 32 位指令
m_data_rdata	32	I	m_data_addr 对应的 32 位数据
i_inst_addr	32	O	需要进行取指操作的F级PC
m_data_addr	32	O	数据存储器待写入地址
m_data_wdata	32	O	数据存储器待写入数据
m_data_byteen	4	O	字节使能信号
m_inst_addr	32	O	M 级 PC
w_grf_we	1	O	grf 写使能信号
w_grf_addr	5	O	grf 中待写入寄存器编号
w_grf_wdata	32	O	grf 中待写入数据
w_inst_addr	32	O	W 级 PC

PC

模块功能定义

- 一个存储当前指令地址的32位寄存器

模块端口定义

名称	位宽	方向	描述
clk	1	I	时钟
reset	1	I	复位信号，复位至0x00003000
en	1	I	写使能
PC_I	32	I	下一个PC值
PC_O	32	O	当前PC值

IM

模块功能定义

外置，指令存储器

模块端口定义

名称	位宽	方向	描述
i_inst_addr	32	I	当前指令地址
i_inst_rdata	32	O	当前地址对应指令

NPC

模块功能定义

- 次地址计算单元，**处于D级寄存器前面**，驱动PC。
- 非跳转指令时，NPC每次自增4，对于beq指令，如果Equal为真，NPC将Imm26[15:0]左移两位并符号扩展后与当前IFU提供的PC值相加，输出。对于J,JAL指令，**NPC将Imm26左移两位后与IFU给出的PC[31:28]拼接后输出。**对于JR指令，PC直接输出RS。
- 对于PC8端口，NPC需要将当前PC值加8后输出。
- 支持跳转到异常处理程序
- 跳转优先级：IntReq>NPCOp

模块规格定义

模块端口定义

名称	位宽	方向	描述
PC	32	I	PC传入的当前PC值
Imm26	26	I	26位指令立即数
RA	32	I	要跳转的寄存器中正确值
NPCOp	4	I	4'b0000:pc+4 4'b0001:beq 4'b0010:J,jal 4'b0011:jr 4'b0100:bne 4'b0101:BGEZ 4'b0110:BGTZ 4'b0111:BLEZ 4'B1000:BLTZ
Equal	1	I	1：相等，0：不相等
IntReq	1	I	中断请求，若IntReq=1,则跳转到0x4180
NPC	32	O	32位下一个地址输出
PC8	32	O	输出PC+8值**（被PCM的输出值取代）**

GRF

模块功能定义

- 可以同时读出两个32位数据或者在时钟上升沿且写信号有效情况下写入一个32位数据。
- **支持内部转发**
- 如果A1=A3并且写使能为1，将WD转发至RD1，如果A2=A3并且写使能为1，将WD转发至RD2。与此同时将WD写入目标寄存器。(注意如果目标寄存器为$0则不转发)

模块规格定义

- 32个寄存器
- 同步复位
- **0 号寄存器**的值始终保持为 0。其他寄存器**初始值（复位后）均为 0**，无需专门设置。

模块端口定义

名称	位宽	方向	描述
Reset	1	I	同步复位信号
clk	1	I	时钟
RegWrite	1	I	写使能信号，高电平有效
A1	5	I	5位寄存器地址值，将指向寄存器的数据读出到RD1
A2	5	I	5位寄存器地址值，将指向寄存器的数据读出到RD2
A3	5	I	5位寄存器地址值，为WD指定写入的寄存器
WD	32	I	要写入的数据
NowPC	32	I	当前指令PC值
RD1	32	O	A1数据输出
RD2	32	O	A2数据输出

EXT

模块功能定义

符号扩展或者无符号扩展

模块规格定义

模块端口定义

名称	位宽	方向	描述
EXTOp	1	I	EXTOp=1:符号扩展 EXTOp=0:无符号扩展
ExtIn	16	I	位扩展输入16位数据
EXTOut	32	O	位扩展输出32位数据

CMP

模块功能定义

比较两个32位数据大小

模块规格定义

模块端口定义

名称	位宽	方向	描述
D1	32	I	第一个数据
D2	32	I	第二个数据
Equal	1	O	Equal = C1==C2?1:0;

ALU

模块功能定义

- 非乘除类计算
- 支持输出溢出信号

模块端口定义

名称	位宽	方向	描述
SrcA	32	I	第一个运算数
SrcB	32	I	第二个运算数
Shamt	5	I	移位位数
ALUOp	5	I	5'b00000:a+b 5'b00001:a-b 5'b00010:a OR b 5'b00011:a NOR b 5'b00100:a XOR b 5'b00101:a AND b 5'b00110:b << 16 (LUI) 5'b00111:b << shamt 5'b01000:逻辑右移：b>>shamt 5'b01001:算术右移：b>>shamt 5'b01010:SLLV 5'b01011:SRLV 5'b01100:SRAV 5'b01101:a<b?1:0(有符号) 5'b01110:a<b?1:0(无符号)
AO	32	O	运算结果
OverFlow	1	O	溢出

MD

模块功能定义

- 乘除计算单元
- 乘法计算花费5周期，除法计算花费10周期
- 自 Start 信号有效后的第 1 个 clock 上升沿开始，乘除部件开始执行运算，同时 Busy 置位为 1。
- 在运算结果保存到 HI 和 LO 后，Busy 位清除为 0。
- 当 Busy 或 Start 信号为 1 时，mfhi、mflo、mthi、mtlo、mult、multu、div、divu 均被阻塞，即被阻塞在 IF/ID。
- 数据写入 HI 或 LO，均只需 1 个 cycle。

模块端口定义

名称	位宽	方向	描述
clk	1	I	时钟
reset	1	I	复位
Start	1	I	计算开始信号
MDOp	3	I	3'b000:mult 3'b001:multu 3'b010:div 3'b011:divu 3'b100:MTHI 3'b101:MTLO
SrcA	32	I	第一个源操作数
SrcB	32	I	第二个源操作数
Busy	1	O	置1表示尚在计算中
HI	32	O	结果
LO	32	O	结果

BE

模块功能定义

- 位于DM前，根据m_data_addr[1:0]与当前指令产生m_data_byteen
- 根据指令与地址重新生成输入数据
- 产生异常时，m_data_byteen为0

模块端口定义

名称	位宽	方向	描述
BEOp	2	I	2'b00:none 2'b01:sw 2'b10:sh 2'b11:sb
MemA	32	I	待写入的数据存储器相应地址
MemD	32	I	未处理的存储器输入数据
IntReq	1	I	产生异常时为1
m_data_byteen	4	O	四字节使能,对应字节写使能
m_data_addr	32	O	待写入的数据存储器相应地址
m_data_wdata	32	O	处理后的存储器输入数据

DM

模块功能定义

外置数据存储器

模块端口定义

名称	位宽	方向	描述
m_data_addr	32	I	待写入/读出的数据存储器相应地址
m_data_wdata	32	I	待写入数据存储器相应数据
m_data_byteen	4	I	四位字节使能
m_inst_addr	32	I	M 级 PC
m_data_rdata	32	O	数据存储器存储的相应数据

DMEXT

模块功能定义

扩展从DM输出的数据

模块端口定义

名称	位宽	方向	描述
DMEXTOp	3	I	000：无扩展 001:无符号字节数据扩展 010：符号字节数据扩展 011：无符号半字数据扩展 100：符号半字数据扩展
MemA	32	I	待读出的数据存储器相应地址
Din	32	I	输入 32 位数据
MemO	32	O	扩展后的32位数据

流水线寄存器

流水线寄存器跟随前级命名，模块的输入的基本数据通路中的多路选择器放在流水线寄存器之后。

流水线寄存器一共有三部分

- 数据通路中的流水线寄存器：流水包括PC
- 控制器中的流水线寄存器：流水控制信号与A1,A2,A3,Tnew
- EP中的流水线寄存器：流水异常编码

流水线寄存器的清空与赋值操作

- 异常产生时（Req==1'b1）：清空所有流水线寄存器
- 复位时（reset==1'b1）：清空所有流水线寄存器
- 暂停时(Stall==1'b1)：清空除E级PC，BD外的所有E级流水线寄存器，并冻结所有D级流水线寄存器

D级寄存器

名称	位宽	描述
clk	1	时钟
en	1	高电平写使能
reset	1	高电平清零
Req	1	当前是否要进入异常
Instr_D_I/O	32	指令
PC8_D_I/O	32	当前F级指令PC值加8
PC_D_I/O	32	当前指令PC值
BD_D_I/O	1	当前指令是否是延迟槽指令

E级寄存器

名称	位宽	描述
clk	1	时钟
reset	1	高电平清零信号
resetPC	1	是否将PC_E复位
Req	1	当前是否要进入异常
RD1_E_I/O	32	GRFRD1读出值
RD2_E_I/O	32	GRFRD2读出值
EXT32_E_I/O	32	32位立即数扩展
Shamt_E_I/O	5	移位位数
PC8_E_I/O	32	跳转指令写入寄存器值
PC_E_I/O	32	当前指令PC值
BD_E_I/O	1	当前指令是否是延迟槽指令

M级寄存器

名称	位宽	描述
clk	1	时钟
reset	1	高电平清零
Req	1	当前是否要进入异常
AO_M_I/O	32	ALU输出
MD_M_I/O	32	MD输出
RD2_M_I/O	32	DM写入值
PC8_M_I/O	32	JAL写入GRF值
PC_M_I/O	32	当前指令PC值
BD_M_I/O	1	当前指令是否是延迟槽指令

W级寄存器

名称	位宽	描述
clk	1	时钟
reset	1	高电平清零
MemO_W_I/O	32	DM输出数据
AO_W_I/O	32	ALU输出数据
MD_W_I/O	32	MD输出数据
PC8_W_I/O	32	JAL写入GRF值
PC_W_I/O	32	当前指令PC值
CP0_W_I/O	32	CP0输出值

MUX

	000	001	010	011	100	5	6	7	out	Sel
功能 MUX:										
GRFWDM	AO_W_O	MD_W_O	MemO_W_O	PC8_W_O	CP0_W_O				GRFWDM_O	GRFWDM_Sel
GRFA3M	Instr_D_O[15:11](rd)	Instr_D_O[20:16](rt)	0x1F						GRFA3M_O	GRFA3M_Sel
SRCBM	RD2MFE_O	EXT32_E_O							SRCBM_O	SRCBM_Sel
MDM	MD.HI	MD.LO							MDM_O	MDM_Sel
PCM	PC_O	EPC							i_inst_addr	PCM_Sel
转发 MUX:										
RD1MFD	GRF.RD1	AO_M_O	MD_M_O	PC8_M_O	PC8_E_O				RD1MFD_O	RD1MFD_Sel
RD2MFD	GRF.RD2	AO_M_O	MD_M_O	PC8_M_O	PC8_E_O				RD2MFD_O	RD2MFD_Sel
RD1MFE	RD1_E_O	GRFWDM_O	AO_M_O	MD_M_O	PC8_M_O				RD1MFE_O	RD1MFE_Sel
RD2MFE	RD2_E_O	GRFWDM_O	AO_M_O	MD_M_O	PC8_M_O				RD2MFE_O	RD2MFE_Sel
RD2MFM	RD2_M_O	GRFWDM_O							RD2MFM_O	RD2MFM_Sel

CU

- 指令改用独热码驱动，用define定义

模块功能定义

非中断状态时的控制

模块端口定义

名称	位宽	方向	描述
clk	1	I	
reset	1	I	
Instr_I	32	I	当前D级指令
Busy	1	I	E级MD信号
ClrInstr	1	I	EP输入信号，若为1，则将D级指令置为nop
A1	5	O	GRF读地址
A2	5	O	GRF读地址
A3_Out	5	O	GRF写地址
NPCOp	4	O	
EXTOp	1	O	
ALUOp	5	O	
MDOp	3	O	
Start	1	O	
SRCBM_Sel	1	O	
MDM_Sel	1	O	
BEOp	2	O	
DMEXTOp	3	O	
RegWrite	1	O	
GRFWDM_Sel	3	O	
Stall	1	O	
RD1MFD_Sel	3	O	
RD2MFD_Sel	3	O	
RD1MFE_Sel	3	O	
RD2MFE_Sel	3	O	
RD2MFM_Sel	1	O	
InstrCode_D	64	O	给EP的D级指令编码
InstrCode_E	64	O	给EP的E级指令编码
CP0Write	1	O	CP0写使能
CP0A	5	O	CP0写地址或读地址
EXLClr	1	O	EXL清除信号
BD	1	O	当前D级指令为跳转指令时BD为1，否则为0
PCM_Sel	1	O	PCM选择信号

功能部件控制信号

- EXCEL中完成

控制器信号流水

- 加粗表示该级输出

信号/流水级	**E**	**M**	**W**
ALUOp[4:0]	**ALUOp_E**		
MDOp[2:0]	**MDOp_E**		
Start	**Start_E**		
SRCBM_Sel	**SRCBM_Sel_E**		
MDM_Sel	**MDM_Sel_E**		
BEOp[1:0]	BEOp_E	**BEOp_M**	
DMEXTOp[2:0]	DMEXTOp_E	**DMEXTOp_M**	
GRFWDM_Sel[2:0]	GRFWDM_Sel_E	GRFWDM_Sel_M	**GRFWDM_Sel_W**
RegWrite	RegWrite_E	RegWrite_M	**RegWrite_W**
A1[4:0]	A1_E	A1_M	A1_W
A2[4:0]	A2_E	A2_M	A2_W
A3[4:0]	A3_E	A3_M	**A3_W**
Tnew[1:0]	Tnew_E	Tnew_M	
InstrCode[63:0]	**InstrCode_E[63:0]**		
CP0Write	CP0Write_E	**CP0Write_M**	
CP0A	CP0A_E	**CP0A_M**	
ERET**(EXLClr)**	ERET_E	**ERET_M**	
MTC0	MTC0_E	MTC0_M	

Tuse信号：

- Tuse_RS0：

  ```
  BEQ|BNE|BGEZ|BGTZ|BLEZ|BLTZ|JR|JALR
  ```

- Tuse_RS1：

  ```
  ADDI|ADDIU|SLTI|SLTIU|ANDI|ORI|XORI|ADD|ADDU|SUB|SUBU|SLT|SLTU|AND|OR|NOR|XOR|SLLV
  |SRLV|SRAV|SW|SH|SB|LW|LH|LHU|LB|LBU|MTHI|MTLO|MULT|MULTU|DIV|DIVU
  ```

- Tuse_RT0:

  ```
  BEQ|BNE
  ```

- Tuse_RT1：

```
SLL|SRL|SRA|ADD|ADDU|SUB|SUBU|SLT|SLTU|AND|OR|NOR|XOR|SLLV|SRLV|SRAV|MULT|MULTU
|DIV|DIVU
```

- Tuse_RT2:

```
SW|SH|SB|MTC0
```

- Tuse_MD:

```
MFHI|MFLO|MTHI|MTLO|MULT|MULTU|DIV|DIVU
```

Tnew类型：

ALUType（TnewE=1）：

```
ADDI|ADDIU|SLTI|SLTIU|ANDI|ORI|XORI|SLL|SRL|SRA|ADD|ADDU|SUB|SUBU|SLT|SLTU|AND|OR|NOR
|XOR|SLLV|SRLV|SRAV|MFHI|MFLO|LUI
```

DMType（TnewE=2）：

```
LW|LH|LB|LHU|LBU|MFC0
```

PCType（TnewE=0）：

```
JAL|JALR
```

暂停与转发策略矩阵

- 对于当前指令，计算Tuse_RS0,Tuse_RS1,Tuse_RT0,Tuse_RT1,Tuse_RT2与Tnew.
- Tuse:处于D级的指令再过多少周期需要使用寄存器输出值
- Tnew：位于当前级(E或M)的指令再过多少周期产生**写入寄存器中的值**
- 暂停条件：
  - &当前指令Tuse小于某级指令Tnew
  - &当前Tuse的读取寄存器地址与后级Tnew对应的寄存器写地址相同
  - &后级写使能为1
  - &读取寄存器地址不为0
  - | D级为MD相关指令，E级Start或Busy为1
  - |D级为eret且E级或M级为mtc0指令且CP0写地址为14
- **暂停操作：**
  - 冻结D级流水线寄存器（D.EN = ~Stall）
  - PC写使能为0(PC.EN = ~Stall)
    - 问题：暂停时到来外部中断，PC无法写入异常处理程序入口值
    - 解决办法：PC使能：(~Stall|Req)
  - 清空E级所有流水线寄存器(E.reset = Stall)包括CU中所有E级控制信号，Tnew_E
- **转发选择信号生成规则：**
  - 当前位点的读取寄存器地址与某转发输入来源的写入寄存器地址相等
  - 当前位点读取寄存器地址不为0
  - **转发源Tnew为0（已经产生了新值）**
  - 根据GRFWDM_Sel判断M级转发PC8还是AO还是MDO
  - 有多个转发源时，选择流水级靠前的
  - 转发源写使能为1

RS策略矩阵

Tuse/Tnew	ALU_E	DM_E	PC_E	ALU_M	DM_M	PC_M	ALU_W	DM_W	PC_W
	1	2	0	0	1	0	0	0	0
0	**S**	**S**	F	F	**S**	F	F	F	F
1	F	**S**	F	F	F	F	F	F	F

RT策略矩阵

Tuse/Tnew	ALU_E	DM_E	PC_E	ALU_M	DM_M	PC_M	ALU_W	DM_W	PC_W
	1	2	0	0	1	0	0	0	0
0	**S**	**S**	F	F	**S**	F	F	F	F
1	F	**S**	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F

CP0（协处理器0）

模块接口定义

信号名	方向	用途	产生来源及机制
clk	I	时钟信号	
reset	I	复位信号	
A[4:0]	I	读写 CP0 寄存器编号	执行 `mfc0`，`mtc0` 指令时产生
DIn [31:0]	I	CP0 寄存器的写入数据	执行 `mtc0` 指令时产生
en	I	CP0 寄存器写使能，优先级低于中断	执行 `mtc0` 指令时产生
PC [31:0]	I	中断/异常时的 PC	
ExcCode[4:0]	I	流水线中发生的异常的类型	EP
HWInt[5:0]	I	6 个设备中断 HWInt[0] Timer0 HWInt[1] Timer1 HWInt[2] 外部中断，检测到外部中断并且当前允许中断后向0x7F20写入数据	外部设备
EXLSet	I	置位 SR 的EXL 位	暂时用不到
EXLClr	I	置 0 SR 的EXL 位	执行 `eret` 指令时产生
BD	I	判断M级指令是否为分支延迟槽指令	由D级产生
Req	O	中断请求	由 CP0 模块确认响应中断
EPC[31:0]	O	EXC 寄存器输出至 NPC	输出值为字对齐
DOut[31:0]	O	CP0 寄存器的输出数据	执行 `mfc0` 指令时产生

模块规格定义

- 位于流水线M级
- 包含四个寄存器：SR,Cause,EPC,PRId

SR 状态寄存器 地址：12

SR寄存器中EXL位由CP0自行更改，其它位只能由外部输入信号更改。

名称	对应位	功能
IM[5:0]	SR[15:10]	为 6 位中断屏蔽位，分别对应 6 个外部中断。相应位置 1 表示允许中断，置 0 表示禁止中断。
IE	SR[0]	为全局中断使能。该位置 1 表示允许中断，置 0 表示禁止中断。
EXL	SR[1]	为异常级。该位置 1 表示已进入异常，不再允许中断，置 0 表示允许中断。

Cause 原因寄存器 地址：13

名称	对应位	功能
IP[5:0]	Cause[15:10]	为 6 位**待决**的中断位，分别对应 6 个外部中断，相应位置 1 表示有中断，置 0 表示无中断。（每周期均更新）
ExcCode[4:0]	Cause[6:2]	异常编码，记录当前发生的是什么异常。产生异常时才更新。若为外部异常，则该域为0，若为内部异常，则如实记录
BD	Cause[31]	分支延迟，只要异常发生在延迟槽的指令，BD就会置位，EPC就指向分支指令。发生异常时才更新

EPC 寄存器 地址：14

EPC 寄存器负责保存中断/异常时的 PC 值。

PRId 寄存器 地址：15

- 只读

模块功能定义：异常处理

内部异常的流水

- 由EP完成

检测中断

流水线响应中断的条件：

- &6个外部中断请求，至少有1个有效且未被屏蔽
- &全局中断使能有效（SR的IE=1）
- &当前不处于中断服务程序中（SR的EXL=0）

流水线响应异常条件：

- &当前不响应外部中断
- &异常码有效
- &当前不处于中断服务程序中

支持的异常：

异常与中断码	助记符与名称	指令与指令类型	描述
0	`Int`(外部中断)	所有指令	中断请求，来源于计时器与外部中断
4	`AdEL`(取指异常)	所有指令	PC地址未字对齐 or PC地址超过 `0x3000 ~ 0x6ffc`
	`AdEL`(取数异常)	lw	取数地址未与 4 字节对齐
		lh，lhu	取数地址未与 2 字节对齐
		lh，lhu，lb，lbu	取 Timer 寄存器的值
		load 型指令	计算地址时加法溢出
		load 型指令	取数地址超出 DM、Timer0、Timer1 的范围
5	`AdES`(存数异常)	`sw`	存数地址未 4 字节对齐
		`sh`	存数地址未 2 字节对齐
		`sh`，`sb`	存 Timer 寄存器的值
		store 型指令	计算地址加法溢出
		store 型指令	向计时器的 Count 寄存器存值
		store 型指令	存数地址超出 DM、Timer0、Timer1 的范围
10	`RI`(未知指令)	-	未知的指令码，仅考虑OP,FUNCT
12	`Ov`(溢出异常)	`add`，`addi`，`sub`	算术溢出

- 分支跳转指令无论跳转与否，延迟槽指令为受害指令时 `BD` 均需要置位。（BD的产生及流水在CU中完成）
- 发生取指异常后视为 `nop` 直至提交到 CP0。（由EP控制，在CU中判断是否将Instr置为0）
- 发生 `RI` 异常后视为 `nop` 直至提交到 CP0。 （什么都不做，此时没有指令线被驱动，所有信号均为0，A1,A2，A3可能不是0，但因为写使能为0，所以不会有转发，因为Tuse没有被驱动，所以不会有暂停，最后的结果和nop等效）
- `load` 与 `store` 类算址溢出按照 `AdEL` 与 `AdES` 处理。

CP0产生中断

同一个周期内完成：

中断操作：

- 保存：将M级指令的PC值写入EPC（若当前中断M级指令为延迟槽指令，则需保存PC-4）
- 跳转：产生中断处理程序入口地址并写入PC
- 关中断：EXL置位，防止再次进入
- 清空所有流水线寄存器
- 将D,E,M级的流水线PC置为0X4180（异常处理程序入口）

异常操作：

- 保存：将M级指令的PC值写入EPC（若产生异常指令为延迟槽指令，则需保存PC-4），将ExcCode写入Cause
- 跳转：产生中断处理程序入口地址并写入PC
- 关中断：EXL置位，防止再次进入
- 清空所有流水线寄存器
- 将D,E,M级的流水线PC置为0X4180（异常处理程序入口）

产生中断时避免M级PC为0：暂停时不清空E级PC的流水线寄存器与CU中的BD_E

由ERET指令结束中断

- eret在D级完成译码，并执行。eret在CU中流水到M级，输出，令EXL复位。eret在D级执行跳转操作。
- eret在D级的暂停条件：D级为eret指令并且流水线中存在mtc0指令并且mtc0的写地址为14

为了使eret在跳转时不产生延迟槽，在PC与IM中间插入PCM，PCM由CU控制

eret触发的操作：

- 恢复PC：将EPC写入PC
- 开中断：清除EXL，允许中断再次发生

question：

1.sw类指令发生异常时，是否需要阻止其写入DM： 是，在BE中实现

2.跳转类指令的受害指令

3.异常出现时是否应当顾及中断使能与中断优先级 否

4.Cause的更新

5。BD的更新

6.一条指令发生多个异常

EP

模块功能定义

- 产生F,D,E级的异常并流水
- 控制ClrInstr
- **当前级有异常时，流水前级异常，否否则流水本级异常**

模块端口定义

名称	位宽	方向	描述
clk	1	I	时钟
D_clr	1	I	清除D级ExcCode
D_en	1	I	D级寄存器写使能
E_clr	1	I	清除E级ExcCode
M_clr	1	I	清除M级ExcCode
PC_F	32	I	F级PC
InstrCode_D	64	I	D级指令
InstrCode_E	64	I	E级指令
AO_E	32	I	E级ALU计算结果
OverFlow_E	1	I	ALU溢出
ExcCode	5	O	输出异常值
ClrInstr	1	O	是否清除D级指令

测试

- Req用寄存器延时一个周期
- 检查所有输入输出的位宽是否对齐
- EPC输出值对齐
- CP0优先级

```
//取数异常
.text
    lui $t0 0x1234
    ori $t0 $t0 0xabcd
    addi $a0 $0 4
    sw $t0 0($a0)
    addi $a0 $0 5
    lw $t1 0($a0)
    addi $a0 $0 7
    lh $t2 0($a0)
    lhu $t3 0($a0)
    addi $a0 $0 0x7f00
    lh $t4 0($a0)
    lhu $t4 0($a0)
    lb $t4 0($a0)
    lbu $t4 0($a0)
    addi $a0 $0 0x7f08
    lh $t4 0($a0)
    lhu $t4 0($a0)
    lb $t4 0($a0)
    lbu $t4 0($a0)
    addi $a0 $0 0x7f10
    lh $t4 0($a0)
    lhu $t4 0($a0)
    lb $t4 0($a0)
    lbu $t4 0($a0)
    addi $a0 $0 0x7f18
    lh $t4 0($a0)
    lhu $t4 0($a0)
    lb $t4 0($a0)
    lbu $t4 0($a0)
    lui $a0 0x7fff
    ori $a0 $a0 0xffff
    lw $t5 4($a0)
    addi $a0 $0 0x3000
    lw $t0 0($a0)
    addi $a0 $0 0x7000
    lw $t0 0($a0)
    addi $a0 $0 0x7efc
    lw $t0 0($a0)
    addi $a0 $0 0x7f0c
    lw $t0 0($a0)
    addi $a0 $0 0x7f1c
    lw $t0 0($a0)
    lui $a0 0x7fff
    ori $a0 0xffff
    lw $t0 0($a0)

end:
    beq $0 $0 end
```

```
.ktext 0x4180
    mfc0 $k0 $14
    mfc0 $k1 $13
    ori $a0 $0 0
    eret

//存数异常
    ori $t0 $0 0x1234
    addi $a0 $0 1
    sw $t0 0($a0)
    addi $a0 $0 7
    sw $t0 0($a0)
    addi $a0 $0 13
    sh $t0 0($a0)
    addi $a0 $0 0x7f00
    sh $t0 0($a0)
    addi $a0 $0 0x7f06
    sh $t0 0($a0)
    addi $a0 $0 0x7f10
    sh $t0 0($a0)
    addi $a0 $0 0x7f18
    sh $t0 0($a0)
    addi $a0 $0 0x7f00
    sb $t0 0($a0)
    addi $a0 $0 0x7f06
    sb $t0 0($a0)
    addi $a0 $0 0x7f10
    sb $t0 0($a0)
    addi $a0 $0 0x7f18
    sb $t0 0($a0)
    lui $a0 0x7fff
    ori $a0 $a0 0xffff
    sw $t0 4($a0)
    addi $a0 $0 0x7f08
    sw $t0 0($a0)
    addi $a0 $0 0x7f18
    sw $t0 0($a0)
    lui $a0 0xffff
    ori $a0 $a0 0xfffc
    sw $t0 0($a0)
    addi $a0 $0 0x3000
    sw $t0 0($a0)
    addi $a0 $0 0x7efc
    sw $t0 0($a0)
    addi $a0 $0 0x7f0c
    sw $t0 0($a0)
    addi $a0 $0 0x7f1c
    sw $t0 0($a0)

end:
    beq $0 $0 end
    nop

    mfc0 $k0 $14
    mfc0 $k1 $13
    add $a0 $0 $s0
    addi $s0 $s0 4
    eret

//未知指令
branch:
    ori $t1 $0 0x1234
    ori $t2 $0 0x5678
    ori $s0 $0 0x300c
```

```
    bgezal $t1 branch #0x0000_300c
    addi $s1 $s1 1
    bltzal $t1 branch
    addi $s1 $s1 1
    bc1f branch
    addi $s1 $s1 1
    clo $t1 $t2
    addi $s1 $s1 1
    clz $t1 $t2
    addi $s1 $s1 1
    madd $t1 $t2
    addi $s1 $s1 1
    maddu $t1 $t2
    addi $s1 $s1 1
    msub $t1 $t2
    addi $s1 $s1 1
    msubu $t1 $t2
    addi $s1 $s1 1

end:
    beq $0 $0 end
    nop

    mfc0 $k0 $14
    mfc0 $k1 $13
    addi $s0 $s0 4
    mtc0 $s0 $14
    eret

//溢出异常Ov
.text
    ori $s0 $0 0x3020
    lui $t0 0x7fff
    ori $t0 $t0 0xffff
    ori $t1 $0 1
    lui $t2 0x8000
    ori $t2 $t2 0x0000
    lui $t3 0xffff
    ori $t3 $t3 0xffff
    add $t3 $t0 $t1
    add $t3 $t1 $t0
    add $t0 $t2 $t3
    add $t0 $t3 $t2
    addi $t3 $t0 1
    sub $t0 $t2 $t1
    sub $t0 $t0 $t3

end:
    beq $0 $0 end

    mfc0 $k0 $14
    mfc0 $k1 $13
    addi $s0 $s0 4
    mtc0 $s0 $14
    eret
```