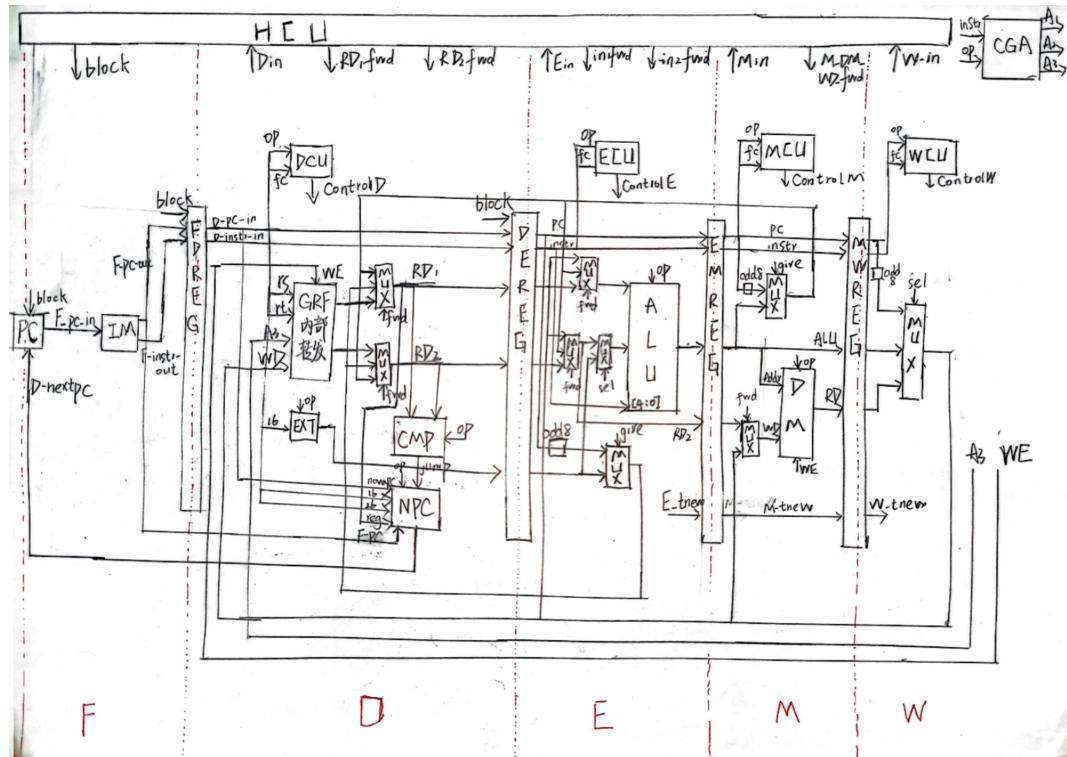


# 设计草稿

## 顶层电路

- 该电路为p5电路，p7与其主要差异有如下几点：

- IM,DM外置。
- 新增乘除槽。
- 新增系统桥，计时器。
- M级增加CPO，其输出结果流水至W。



## 指令分类

- 在p6的基础上新增了ERET、MFC0、MTC0、SYSCALL。
- p6指令的分类

类型名	指令名
cal_rr	add,sub,and,or,slt,sltu
cal_ri	addi,andi(符号扩展),ori
load	lw,lh,lb
store	sw,sh,sb
md_cal	mult,multu,div,divu
md_read	mflo,mfhi
md_write	mtlo,mthi
branch	beq,bne

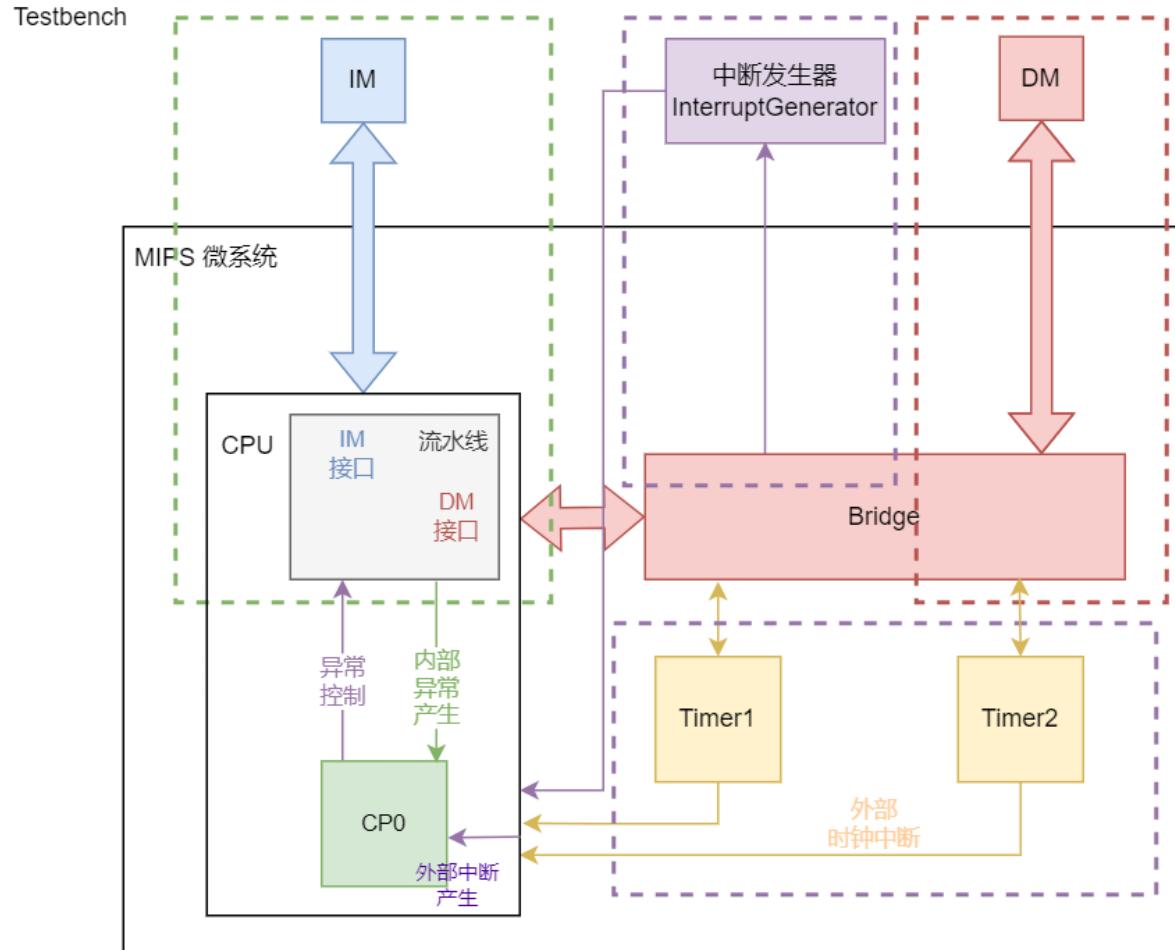
类型名	指令名
shift	sll
lui	lui
jal	jal
jr	jr
j	j

## 测试方案

- 首先将p7顶层新增信号屏蔽掉后提交到p6强测中去检测之前已实现指令是否正确。
- 然后围绕 `eret`, `mfc0`, `mtc0`, `syscall` 四条新增指令手动构造测试数据，重点关注以下几点：
  - eret后一指令不能执行问题；
  - eret与mtc0会造成阻塞的问题；
  - mfc0与其后指令的冲突与转发问题；
  - mtc0与其前指令的冲突与转发问题；
  - 外部中断能否命中问题。

## 补充

### 概念图



## 异常类型

ExcCode 的编码必须遵守规范，不然在评测的时候可能会出现问题。

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。 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	取数地址未与 2 字节对齐。
		lh , lb	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、中断发生器的范围。
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、中断发生器的范围。
8	Syscall (系统调用)	syscall	系统调用。
10	RI (未知指 令)	-	未知的指令码。
12	ov (溢出异 常)	add , addi , sub	算术溢出。

## 测试说明

## 官方测试说明

- 为便于进行测试，我们允许从 0x417C 直接前进到 0x4180，此种情况下 CPU 行为与 P6 一致，不应有中断响应等其他行为。
- 测试数据规范：
  - 测试时不会出现跳转到未加载指令的位置的情况。
  - `eret` 只会出现在中断处理程序中，后可能紧跟另一条非 `nop` 的指令。
  - 测试程序保证不会写入 Cause，但可能写入 SR 和 EPC。
  - 测试程序只会通过指令 `sb $0, 0x7f20($0)` 访问中断发生器（响应中断），且只会在中断处理程序中访问。
  - 中断处理程序会对寄存器和内存进行读写来验证 CPU 的正确性。
  - 中断处理程序执行过程中保证不出现异常，且不会产生中断。
- 官方 tb 示例：
  - 外设不给予中断时，使用的 tb 为：[下载链接](#)。
  - 评测机通过检测同学们的宏观 PC 给予中断信号并对中断进行测试，例如此[下载链接](#)中的 tb 会在处理器的宏观 PC 第一次到达 0x3010 时给予 CPU 一个中断信号。

## 中断处理程序

```
.ktext 0x4180
mfc0 $k0,$12
mfc0 $k0,$13
mfc0 $k0,$14
addi $k0,$k0,4
mtc0 $k0,$14
eret
add $2,$2,$2
```

## 思考题

### 第一题 请查阅相关资料，说明鼠标和键盘的输入信号是如何被 CPU 知晓的？

- 鼠标和键盘类似教程中的中断发生器，两者在有输入信号时会产生一个中断请求信号，输入给 cpu，cpu 则会响应这两种不同的中断（读取鼠标和键盘的输入）。

### 第二题 请思考为什么我们的 CPU 处理中断异常必须是已经指定好的地址？如果你的 CPU 支持用户自定义入口地址，即处理中断异常的程序由用户提供，其还能提供我们所希望的功能吗？如果可以，请说明这样可能会出现什么问题？否则举例说明。（假设用户提供的中断处理程序合法）

- 我认为可以这么做但是可能会有诸多潜在的问题：
  - 不便于程序员设计软件，程序员需要自己思考应该将入口地址设置在哪里；
  - 增加了cpu设计的复杂度，需要设计能够读取入口地址的硬件；
  - 不利于确定异常处理程序是否会与其他程序发生冲突。

### 第三题 为何与外设通信需要 Bridge?

- 首先外设是有非常多种类的，每种外设有不同的地址，如果没有系统桥的话，那么就需要增加许多CPU的接口并修改CPU的内部结构，这显然不符合“高内聚，低耦合”的设计思想。

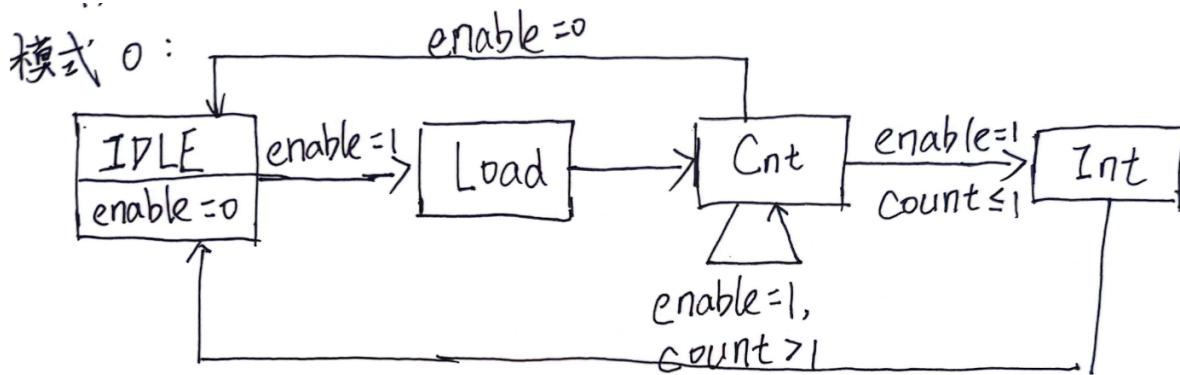
### 第四题 请阅读官方提供的定时器源代码，阐述两种中断模式的异同，并针对每一种模式绘制状态移图。

- 相同点：

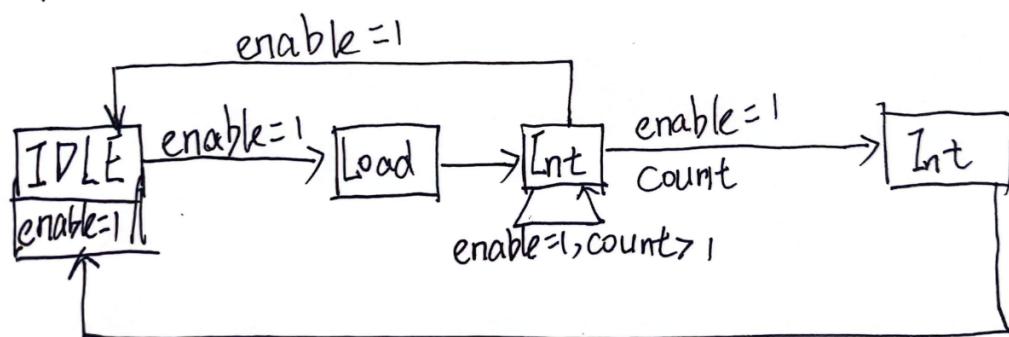
- 两者均是Moore型状态机；
- 均通过控制寄存器、初值寄存器、计数值寄存器实现功能。

- 不同点：

- 模式0通常用于产生定时中断，模式1通常用于产生周期性脉冲。
- 模式0在计数值寄存器数到0后会使控制寄存器使能信号(`ctrl[0])变为0。当外界重新赋值为1时才会重新把要计的时间载入计数器然后开始计数；而模式1在计数值寄存器数到0后会自动将要计的时间载入计数器然后开始载入。
- 模式0在每个计数循环会产生一个或多个周期的中断信号，而模式1只会产生一周期。



模式 1:



### 第五题 倘若中断信号流入的时候，在检测宏观 PC 的一级如果是一条空泡（你的 CPU 该级所有信息均为空）指令，此时会发生什么问题？在此例基础上请思考：在 P7 中，清空流水线产生的空泡指令应该保留原指令的哪些信息？

- 会将 32'd0 写入 EPC，这样的话在中断处理程序结束后会返回到错误的地址；并且 BD 信号也可能出错。
- 如果是reset，pc写为0x3000；
  - 如果是req,pc保留；
  - 如果是block, pc与isBD均保留。

## 为什么jalr 指令为什么不能写成jalr \$31, \$31?

- 不知道是先链接还是先跳转，这会导致当其延迟槽指令异常时重新执行该执行时可能会得到不一样的结果。

Register specifiers rs and rd must not be equal, because such an instruction does not have the same effect when reexecuted. The result of executing such an instruction is UNPREDICTABLE. This restriction permits an exception handler to resume execution by re-executing the branch when an exception occurs in the branch delay slot.