# 控制单元CU

根据电路图对信号说明部分进行了更新和修正

控制单元检测流水按的冒险，控制流水段寄存器的冲刷和阻塞以及异常处理

# **检测的冒险：**

**检测load-use冒险**：

根据输入的id段得到的rs寄存器地址和最终的rt寄存器地址，以及ex段继承自上一流水段的rd寄存器地址和mem\_read访存信号，如果ex段是访存指令（mem\_read有效），且rd与rs和rt中的一个相等，则通过组合逻辑使得pc\_stall, ifid\_stall, idex\_flush有效，在下一个周期在ex段插入气泡，而if段和id段阻塞，mem段和wb段正常执行。

**检测控制冒险：**

分支指令在mem段算出跳转条件是否成立，比较mem段分支地址单元提供的正确下地址和ex段给出的预测器给出的预测下地址，如果不一样，使得ifid\_flush, idex\_flush, exmem\_flush有效，同时通过组合逻辑使得pc\_src选择mem段的正确的跳转地址作为下个周期写入pc的值，并且使得bpu\_w\_en有效，送入mem段的当前pc以及正确的跳转地址，通知分支预测单元进行更新。注意由于分支预测器可能允许非跳转指令误撞，所以cu每个周期都在进行有意义的分支冒险检测。

**绝对跳转指令和过程调用指令**在id段能判断出来，根据输入的指示信号，将ifid\_flush有效，pc\_src改为选择绝对跳转地址或寄存器跳转指令。

**分支和绝对跳转**，分支和load-use的冒险可能同时检测到，由于此时分支指令之后的指令都是错的，所以最优先生成分支指令发生冒险时的信号，而绝对跳转和load-use都需要id段的信息，所以不可能同时发生。

**检测syscall**：

id段生成的syscall信号，但是此时id段可能位于错误分支中。所以要将该信号传递到mem段再触发，因为如果能成功传递到exmem流水段寄存器，说明这个syscall是正确的要执行的指令。冲刷和阻塞信号同分支指令冒险的情况。输出对应的中断向量地址，pc\_src改成选择中断向量。cu\_cp0\_w\_en有效， cu\_exec\_code输出syscall对应的8，cu\_epc\_src选择exmem段的pc。不会与分支跳转冒险同时发生。

**检测eret/ret：**

id段判明，exmem段触发，效果同控制冒险，不过pc\_src改成epc/$31， 并冲刷前三个流水段寄存器，使得ifid\_flush, idex\_flush, exmem\_flush有效，。不可能分支跳转冒险同时发生。

**检测外部中断：**

外部中断信号从cp0传入，判明发生中断后，将cu\_pc\_src修改为中断向量的选择端，cu\_cp0\_w\_en置为有效，cu\_exec\_code赋值为0（表示是外部中断）,根据当前的分支冒险检测结果让cu\_epc选择ex段的pc还是正确跳转地址。

中断向量由cu计算，其他的如跳转地址可以直接连线过去。

## **输入**

### **Id\_ir**

从id段送入的rs的地址，用于检查和ex段的数据依赖

### **mem\_stall**

由存储器送入的阻塞信号，cu负责进一步对流水段寄存器生成阻塞信号

### **ifid\_rs\_addr [4:0]**

从id段送入的rs的地址，用于检查和ex段的数据依赖

### **real\_rt\_addr [4:0]**

从id段送入的经过选择的rt的地址，用于检查和ex段的数据依赖

### **idex\_rd\_addr [4:0]**

从ex段送入的经过选择的真正的目标寄存器地址，用于检查和id段的数据依赖

### **idex\_mem\_read**

从ex段送入的读内存指示，在其有效的情况下数据冒险结果才有效

只需要检查load-use冒险

### **predicted\_idex\_pc [31:0]**

从ex段送入的pc地址，在mem段是分支跳转指令的情况下，此pc是分支预测器给出的预测地址，用于检查控制冒险

### **target\_exmem\_pc [31:0]**

从mem段送入的计算偏移量并且经过条件选择的正确跳转地址，用于检查控制冒险和确定返回地址.

### **cp0\_intr**

由cp0的cause寄存器送入的中断组合信号，是8个中断比特的或

用于指示cu改变pc\_src，提供中断向量地址，情况保存epc并冲刷流水段寄存器

### **id\_jmp**

由id送入的绝对跳转指示信号（包括j和jal），指示cu改变pc\_src并准备冲刷ifid流水段寄存器

### **exmem\_eret**

由exmem送入的异常返回信号，指示cu改变pc\_src，冲刷前三个流水段寄存器

### **exmem\_syscall**

由id生成，传送到exmem的系统调用指示信号，指示cu改变pc\_src，提供中断向量地址，视情况保存epc并冲刷流水段寄存器

## **输出**

### **cu\_pc\_src [3:0]**

选择pc下址：

0 – 绝对跳转地址

1 – 寄存器跳转地址

2 – 中断向量

3 – epc

4 – 正确跳转地址

5 – bpu预测结果（含正常+4）

### **cu\_pc\_stall**

### **cu\_ifid\_stall**

### **cu\_idex\_stall**

### **cu\_exmem\_stall**

### **cu\_wb\_stall**

各寄存器阻塞信号

### **cu\_ifid\_flush**

### **cu\_idex\_flush**

### **cu\_exmem\_flush**

各流水段寄存器冲刷信号

### **cu\_cp0\_w\_en**

cp0写使能，写入cause，关闭中断(移动status)，发生异常时有效

### **cu\_exec\_code [4:0]**

写入cause的原因：

0 – 外部中断

8 – 系统调用

### **cu\_epc**

送入cp0的中断返回地址

检测到mem\_syscall时，将epc赋值为predicted\_idex\_pc,检测到外部中断的时候，根据分支冒险的情况来确定epc。

### **cu\_vector [31:0]**

中断向量地址，不同的中断类型可能不同。（目前默认为0x80000180）

### **cu\_bpu\_w\_en**

bpu写使能，发生控制冒险时有效，用于更新bpu

# 程序说明

程序中每次将所有输出信号进行初始化， 具体如下：

cu\_pc\_src = 4'b0101;

cu\_pc\_stall = 1'b0;

cu\_ifid\_stall = 1'b0;

cu\_idex\_stall = 1'b0;

cu\_exmem\_stall = 1'b0;

cu\_ifid\_flush = 1'b0;

cu\_idex\_flush = 1'b0;

cu\_exmem\_flush = 1'b0;

cu\_cp0\_w\_en = 1'b0;

cu\_exec\_code = 5'b00000;

cu\_epc = 32'h00000000;

cu\_vector = 32'h80000180;

bpu\_write\_en = 1'b0;

后面按照信号说明来进行输出信号的修改。