Verilog 流水线 CPU 设计文档

p7 编写流程

- 异常中断
  - 添加 eret，syscall，mtc0，mfc0 控制信号
    - mfc0 需要设置 RegWrite 和 RegFrom 以修改寄存器
    - mtc0 的 t_use_rt 设置为 2（在 M 级需要使用 GPR[rt] ）
    - mfc0 的 t_new_e 设置为 2（在 M 级后才能得到写入数据 ）
  - 修改各级部件，支持异常检测
    - 注意异常流水的优先级顺序
    - Load/Store 类指令在 E 级计算地址溢出异常，流水到 M 级统一检测
  - 修改各级部件，处理异常中断
    - npc 跳转到 0x4180
    - 各级（包括 W 级）pc 修改为 0x4180
    - 各级（包括 W 级）其他流水寄存器清空
    - M 级添加 cp0
      - 若异常，EPCOut 需要设置。若为延迟槽指令，设置为受害 PC-4，否则设置为受害 PC。
      - 中断优先级更高，因此对 ExcCode 域赋值时，若检测到中断信号，设置为 0，否则设置为 ExcCodeIn。
- 系统构建
  - 将原 mips 模块更名为 cpu，并修改其 IO 信号
  - 编写 bridge 模块，支持读写 tc0、tc1 和**中断发生器（23 级新增）**。
  - 按要求创建新的 mips 模块，在其中添加 cpu、bridge、tc0、tc1 模块。

测试用例

见文件夹中的 test.asm。

思考题

- 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？
  1. 硬件设备生成信号。
  2. 接口控制器接收信号并通知 CPU。
  3. CPU 通过中断处理或轮询读取数据。
- 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）
  - 理论上是可行的，但即使用户的中断处理程序合法，也会因为编写代码可能存在的漏洞导致系统的安全性风险增加，稳定性降低。此外，支持动态地址也会增加性能和管理的开销。
- 为何与外设通信需要 Bridge？

    - 主机和外设之间存在接口标准、速度带宽、数据格式、寻址方式、中断机制等多种差异，没有桥接器进行协调，会增加 cpu 编码的复杂程度，甚至无法实现通信等功能。
- 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：都是基于 32 位计数器，可以赋初始值，倒数到 0 产生中断。
    - 不同点：
        - 计数器行为：
            - 模式 0：倒计时到 0 后停止计数。
            - 模式 1：倒计时到 0 后自动加载初值，继续倒计数。
        - 中断信号持续时间：
            - 模式 0：中断信号持续有效，直到屏蔽或者重启。
            - 模式 1：由于自动加载，中断信号仅持续一个周期。
        - 应用：
            - 模式 0：定时中断。
            - 模式 1：周期脉冲。
- 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？
    - 清除了该级的 pc，可能导致中断时记录了错误的受害 PC（原本为一个合法地址，记录为空地址），导致 eret 返回错误的地址。
    - 因此，至少应该保留 pc 的值。
- 为什么 jalr 指令为什么不能写成 jalr $31, $31？
    - 返回地址覆盖了跳转目标的地址，导致功能上产生冲突。