分支预测器设计文档

whz

# 功能说明

分支预测器位于流水线的最前端，它根据IF段提供的下一条指令PC给出预测的下一条指令PC，并且能够根据控制单元给出的信号，动态地修改预测结果。

# 模块说明

分支预测器的端口如下表所示

|  |  |  |  |
| --- | --- | --- | --- |
| 性质 | 端口名 | 位宽 | 说明 |
| 输入 | clk | 1 | 全局同步时钟 |
| reset | 1 | 全局复位信号 |
| bpu\_w\_en | 1 | 控制冒险发生标志，同时也是预测表状态更新使能 |
| current\_pc | 32 | 用于索引预测表项，获取预测PC |
| tag\_pc | 32 | 用于索引需要更新的预测表项 |
| next\_pc | 32 | 用于更新对应预测表项的预测PC |
| 输出 | predicted\_pc | 32 | 预测的下条指令PC |

## 预测机制

IF段总是将该段的当前PC加4作为下一条指令PC送入分支预测器的输入端current\_pc。分支预测器会截取current\_pc的一段作为下标，以直接映射的方式索引分支预测表。一个分支预测表的表项具有以下信息：

1. valid：标志该表项的有效性。有效的场合，直接使用存储的预测PC作为下条指令的PC输出；
2. predict：标志该表项对应的预测PC是否可以修改；
3. pc：存储的预测PC，注意其语义并不是分支指令的跳转地址，而是预测的地址，所以可以是顺序地址，也可以是跳转地址。

预测逻辑是纯组合逻辑。分支预测器通过current\_pc取出表项后，直接观察valid位，如果有效就直接输出预测PC；如果valid位无效则直接输出current\_pc作为下条指令PC，相当于不预测。

注意预测逻辑并不像cache那样会比对标签（因为并没有存储标签），所以有可能出现正常的顺序执行指令错误地预测出其他指令的地址，这时分支预测器会直接输出错误的PC。但是控制单元对每一条指令都会进行控制冒险的监测，所以可以保证指令提交时顺序的正确性。

另外要提醒的是一旦valid位有效，它就不会再无效。对应的表项将一直承担预测的任务，包括顺序执行指令的和分支指令的。

分支预测器的预测行为可用如下的框图直观地表示：

## 更新机制

只有在控制冒险发生时分支预测器才会更新预测PC。控制单元检测到控制冒险后，将bpu\_w\_en有效，将MEM段的PC和分支地址计算模块得到的正确的下条指令PC分别送入分支预测器的tag\_pc和next\_pc端口。分支预测器在bpu\_w\_en有效的周期，根据tag\_pc直接索引到对应的预测表项，首先将其valid位置为有效，表明这个表项从此与预测任务相关联了。然后检查predict位，如果为0，则将next\_pc写入表项的预测PC，由于所有predict位在复位时设置为0，所以一个新的valid有效的预测表项刚好predict位为0，允许next\_pc写入。如果predit位为1，则不写入，表示分支预测器允许一次错误。无论predict是否为0，在更新逻辑的末尾，都要将predict翻转，以实现延迟更新的效果。也就是说：

1. 如果当前predict为0，预测PC被更新，翻转predict为1，保证第二次预测错误时预测PC不被更新，即容忍一次预测错误；
2. 如果当前predict为1，预测PC不被更新，此次不更新的语义即容忍一次错误预测，但是要翻转predict为0，表明下次连续预测错误就要被更新。

分支预测器只有在同一个预测表项连续两次预测错误时才会更新预测PC。所以平时bpu\_w\_en无效的场合，表明tag\_pc对应的预测表项提供的预测PC是正确的，这时就要将表项的predict位置为1。需要注意的是，如果无视valid位直接将tag\_pc对应的表项的predict位置1，会影响valid位刚有效的表项的预测PC的写入，所以只有valid位有效的表项的predict位才能在bpu\_w\_en无效的情况下置1。

分支预测器的更新逻辑可以总结为如下的表格：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bpu\_w\_en | valid位 | predict位 | 新predict位 | 预测PC |
| 1 | 1 | 1 | 0 | 保持不变 |
| 0 | 1 | 修改为next\_pc |
| 0 | 一定为0 | 1 | 修改为next\_pc |
| 0 | 1 | 1 | 1 | 保持不变 |
| 0 | 1 | 保持不变 |
| 0 | 一定为0 | 0 | 为无效值 |

# 行为分析

分支预测器在默认情况下，也就是预测表项valid位无效时，直接输出current\_pc，即默认不预测，让IF段对应指令物理上的下一条指令（PC+4）进入流水线，对于分支指令来说，相当于默认不跳转。所以一个预测表项第一次有效时，预测PC填入的一定是分支指令的跳转目标地址。该存储了跳转地址的分支预测器下一次修改预测PC时，写入的PC性质将无法确定:它可能是原来的分支指令的顺序下地址，也可能是其它映射到同一表项的指令的下地址。

# 常见误区

常见的分支预测器，存储的是跳转与否的决策函数，以及跳转指令地址。一般是用输入PC预测出对应的下一条PC地址，并使用标签准确匹配待预测的PC。

在我们的分支预测器实现中，为了减少硬件资源消耗，并没有存储标签，所以无论是跳转指令还是顺序执行指令都有可能被预测；并且我们不是存储跳转决策函数以及跳转地址，而是直接存储预测地址；此外我们用于索引预测表项的输入PC并不是预测PC逻辑上的上一条指令PC，而是上一条指令PC+4。

所以，用于索引更新表项的tag\_pc，虽然来自MEM段，但是并不是MEM段的指令对应的PC，而是其PC+4。如果没有加4，则总是往真正需要更新的预测表项的上一项写入正确的预测地址，结果是分支预测器一直预测错误。

# 仿真测试

首先通过针对分支预测器模块的仿真文件进行单元测试。测试的主要项目是分支预测期的更新机制。即保持current\_pc和tag\_pc恒定的情况下，在特定的时机将bpu\_w\_en有效，对next\_pc进行多次更改，观察预测PC是否确实被更新以及容忍一次错误的策略有没有通过波形图反应出来。

之后将分支预测器接入流水线，通过执行有一定复杂度的测试程序，在对整个流水线的测试中，验证分支预测器功能的正确性。