计算机体系结构实验报告LAB04

题目： 动态分支预测  
姓名： 魏钊  
学号： PB18111699

**实验目的**

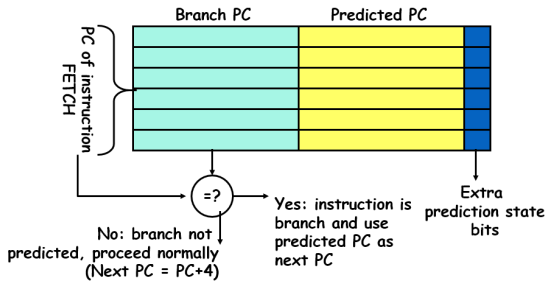
* 实现BTB（Branch Target Buffer）和BHT（Branch History Table）两种动态分支预测器
* 体会动态分支预测对流水线性能的影响

**实验平台**

**Vivado**

**实验设计与过程**

1. **BTB**

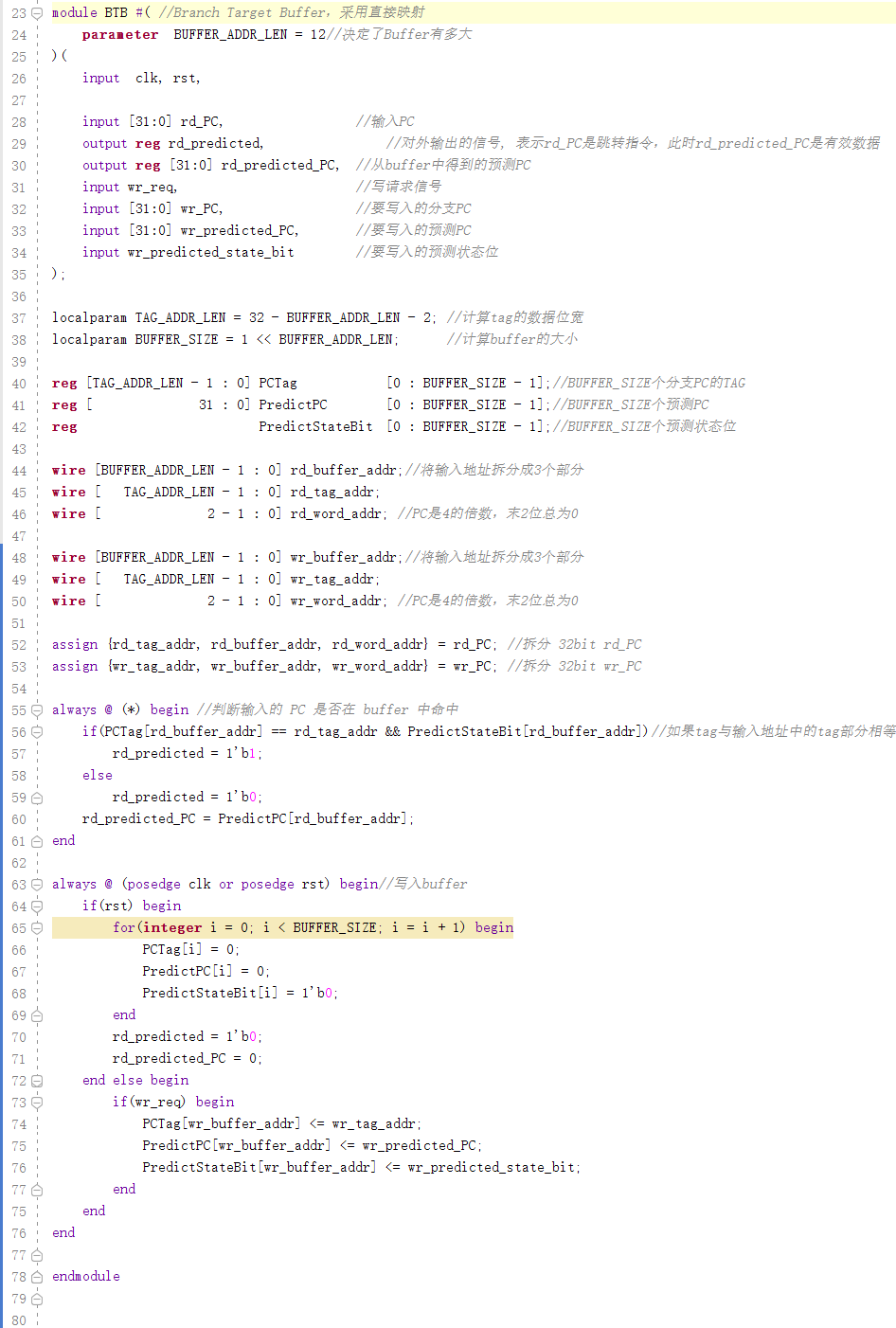


BTB 模块中的 buffer 结构如上图，与 Cache 类似。第一部分 BranchPC 是缓存的地址(tag)列表，采用直接映射策略。 第二部分 PredictedPC 是预测的跳转地址。第三部分为 1 位宽，表示该项是否有效。

输入的 PC 低位寻址到对于表项，对 tag 部分进行比较，如果相等， 并且该项是有效的， 则认为命中， 然后使用 Predicted PC 作为输出， 表示根据历史信息，这是一个分支指令并且预测跳转。

BTB 模块介于 IF 和 ID 段寄存器之间，对于 IF 段寄存器输出的PCF，输入到 BTB 中， BTB 查找 Buffer，如果命中，则输出地址向 NPC模块输入， NPC 模块选择预测 PC 作为下一个 PC，如果不命中，直接使用 PCF+4 作为下一个 PC，同时， 命中的状态位（1 表示进行了预测且预测跳转， 0 表示不预测） 和预测 PC 也随流水段流到 EX 段。

等到当前指令到达了 EX 段， 就可以确定当前指令是否跳转，如果实际跳转但之前未预测跳转或者实际不跳转但之前预测跳转，都需要清空 ID 和 EX 段的寄存器， NPC 模块重新生成下一个 PC。 在实际跳转但之前未预测跳转的情况下，需要把当前 PC 和跳转的 PC 更新写入BTB 的 buffer 中。 在实际不跳转但之前预测跳转的情况下，需要把BTB 的 buffer 中的对应项的有效位置为 0。



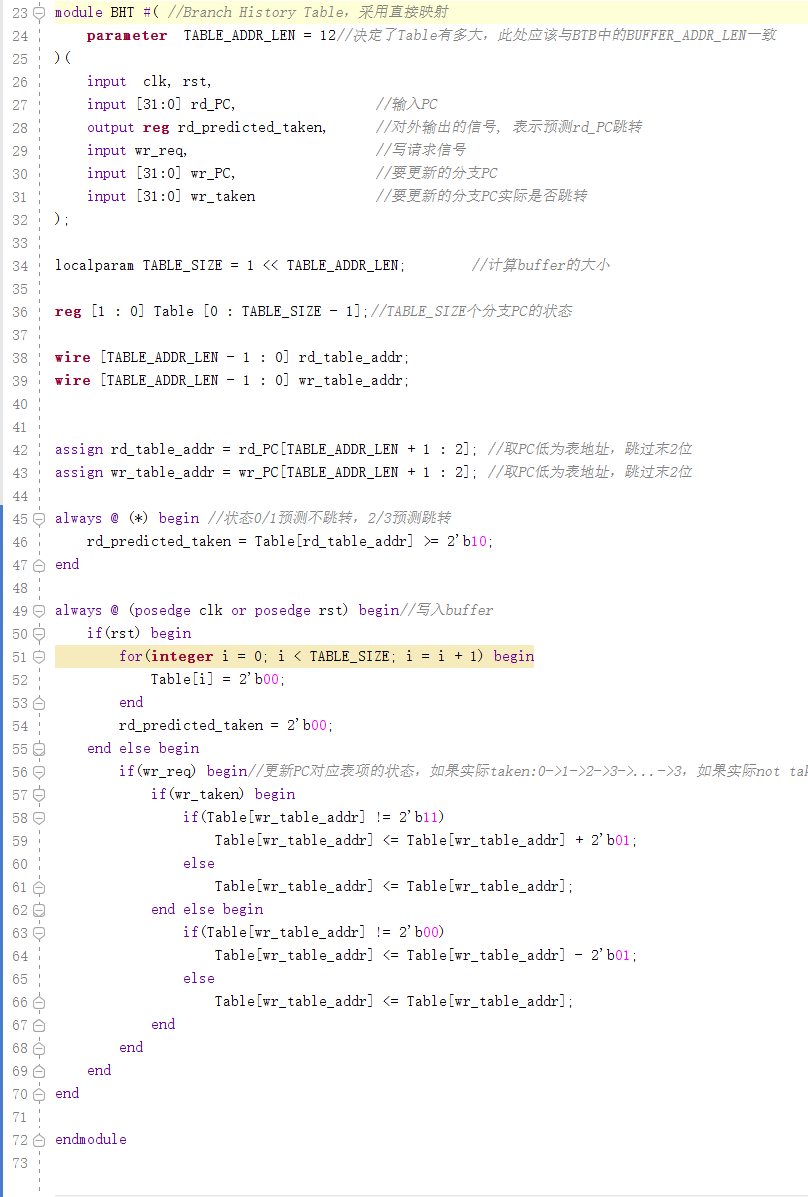
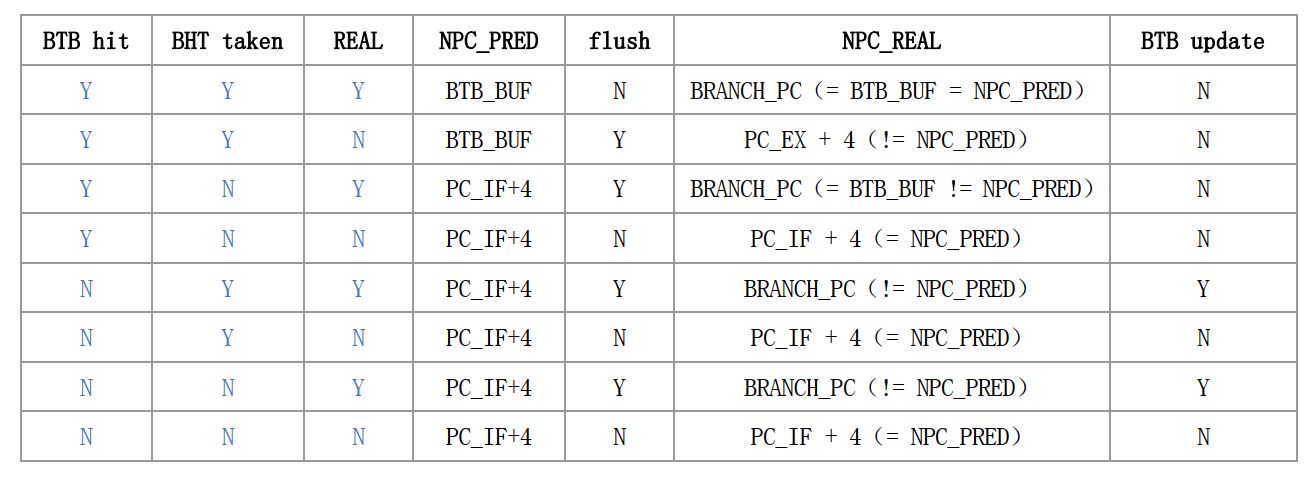
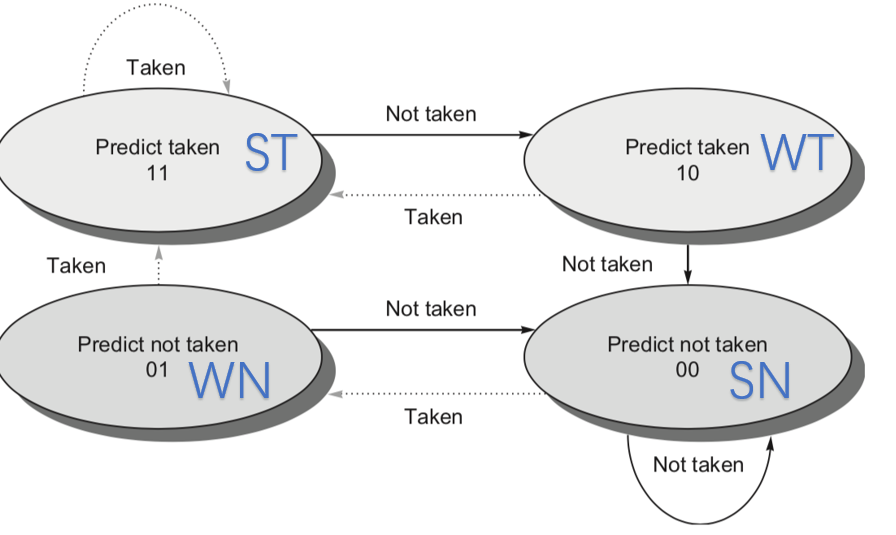
1. **BHT**

分支策略采用 BHT 时， 分支 PC 和预测的 PC 仍然保存在 BTB 模块的buffer 中，但 BTB 的 buffer 不再决定是否预测跳转， BTB 模块的命中与否只决定是否进行预测（或者说当前指令是否为分支指令）。预测的状态位（0 表示不预测， 1 表示预测） 和预测的 PC 也随流水线流到 EX 段。 BTB 的更新方法同上（不再使用有效位，只更新地址）。

BHT 模块则进行预测是否跳转。 维护一个表，每个表项为 2bit 的状态，查表同样采用直接映射，但不再维护和比较 tag。 BHT 模块对于输入的 PC，查到对应表项后，如果 2bit 的值为 0 或 1，则预测跳转，如果为 2 或 3，则预测不跳转，预测结果（0 表示不跳转， 1 表示跳转） 也随流水线流到 EX 段。

等到当前指令到达了 EX 段，就可以确定当前指令是否实际跳转。根据之前传过来的是否预测的状态位（BTB hit）、是否预测跳转状态位(BHT taken)为以及当前的实际跳转结果，我们可以决定是否清空流水线的 ID 和 EX 段进行重新取指令、是否对 BTB 进行更新（见下面的真值表），以及 BHT 如何更新（见下面的状态转换图）。

BHT 的每一项本质是一个 2bit 的状态机， 每次执行分支指令都要更新。 如果 EX 段实际跳转， 则对应表项状态按 0->1->2->3->...->3更新，如果 EX 段实际不跳转， 则状态按 3->2->1->0->...->0 更新。



**实验结果**

* BTB.S
  + 分支收益和分支代价：预测准确收益2个cycle，预测错误2个Cycle
  + 未使用分支预测的总周期数：(1024 - 8) / 2 = 508 cycle = 307 + 100 \* 2 + 1
  + 使用分支预测的总周期数：(632 - 8) / 2 = 312 cycle = 307 + 2 \* 2 + 1
  + 周期差值(未分支预测的总周期数-使用分支预测的总周期数)：196 cycle
  + 分支指令数：101 条
  + 动态分支预测正确次数和错误次数： 多命中98次，节省196个cycle，加速比为1.628
    - 跳转指令101条，正确1次，错误100次
    - 跳转指令101条，正确99次，错误2次
* BHT.S
  + 分支收益和分支代价：预测准确收益2个cycle，预测错误2个Cycle
  + 未使用分支预测的总周期数：(1074 - 8) / 2 = 533 cycle = 335 + 99 \* 2
  + 使用BTB分支预测总数：(774-8) / 2 = 383
  + 周期差值(未分支预测的总周期数-使用BTB分支预测的总周期数)：150 cycle
  + 使用BTB&BHT分支预测的总周期数：(730 - 8) / 2 = 361 cycle = 335 + (11 + 2) \* 2
  + 周期差值(未分支预测的总周期数-使用BTB&BHT分支预测的总周期数)：172 cycle
  + 分支指令数： 110 条
  + BTB动态分支预测正确次数，错误次数：正确86次，错误24次，加速比为1.414
  + BTB&BHT动态分支预测正确次数和错误次数： 多命中8 \* 9 + 7 + 7 = 86次，节省172个cycle，加速比为1.476
    - B: 跳转指令110条，正确11次，错误99次
    - A: 跳转指令110条，正确9 \* 9 + 8 + 8 = 97次，错误13次
* Quicksort.S(256)
  + 分支收益和分支代价：预测准确收益2个cycle，预测错误2个Cycle
  + 未使用分支预测的总周期数：(136700 - 8) / 2 = 68346
  + 使用BTB分支预测总周期数：(138400 - 8) / 2 = 69196
  + 周期差值(未分支预测的总周期数-使用BTB分支预测的总周期数)：-850 cycle
  + 使用BTB&BHT分支预测的总周期数：(135572 - 8) / 2 = 67782
  + 周期差值(未分支预测的总周期数-使用BTB&BHT分支预测的总周期数)：564 cycle
  + 分支指令数： 16178
  + 使用BTB动态分支预测正确次数和错误次数：正确14350次，错误1828，加速比为0.993
  + BTB&BHT动态分支预测正确次数和错误次数： 正确15057次，错误1121次，加速比为1.012
* MatMul.S(16 \* 16)
  + 分支收益和分支代价：预测准确收益2个cycle，预测错误2个Cycle
  + 未使用分支预测的总周期数：(709228 - 8) / 2 = 354610
  + 使用BTB分支预测的总周期数：(693640 - 8) / 2 = 346816
  + 使用BTB&BHT分支预测的总周期数：(692696 - 8) / 2 = 346344
  + 周期差值(未分支预测的总周期数-使用BTB分支预测的总周期数)：7794
  + 周期差值(未分支预测的总周期数-使用BTB&BHT分支预测的总周期数)：8266 cycle
  + 分支指令数： 4624
  + 动态分支预测BTB正确次数和错误次数：正确4076次，错误548次，加速比为1.022
  + 动态分支预测BTB&BHT正确次数和错误次数：正确4312次，错误312次，加速比为1.024

**对比不同策略并分析以上几点关系**

* 分支收益和分支代价：流水段决定
* 未使用分支预测的总周期数：指令数 + 错误预测数 \* 预测错误惩罚
* 使用分支预测总周期数：指令数 + 错误预测数 \* 预测错误惩罚
* 两者差值：(未使用分支预测的错误预测数 - 使用分支预测的错误预测数) \* 预测错误惩罚
* 未使用分支预测的错误预测数：跳转指令 - 循环个数(最后一条)
* BTB：使用分支预测的错误预测数 = 循环个数 \* 2 (启动与退出)
* BHT(从01启动而不是00)：使用分支预测的错误预测数 = 循环个数(最后一条) + 相同循环种数(启动)

**总结** 通过数据分析对比，可以看出BHT和BTB相结合的策略更好，它的预测错误次数少，总周期少，加速比大，分支预测的效果更好。在某些情况下，由于预测失误较多，可能得到使用分支预测的周期多于未使用分支预测周期的情况。