Lab4 分支预测实验报告

徐煜森 PB16110173

1. **实验目标**

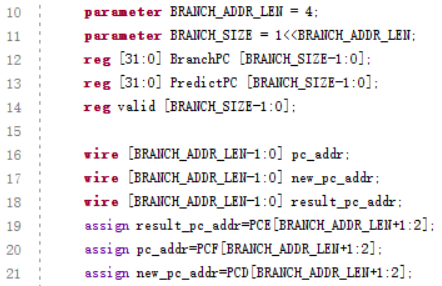
掌握使用BTB和BHT进行分支预测的方法。

1. **实验环境和工具**

环境：Windows 10

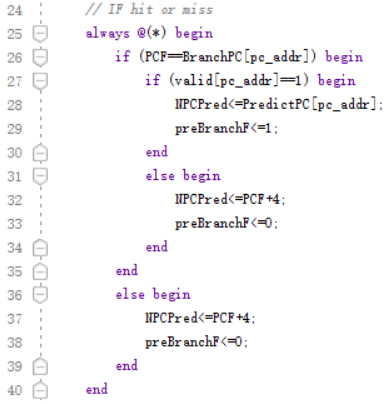
工具：Vivado 2017

1. **实验内容和过程**
2. **BTB**

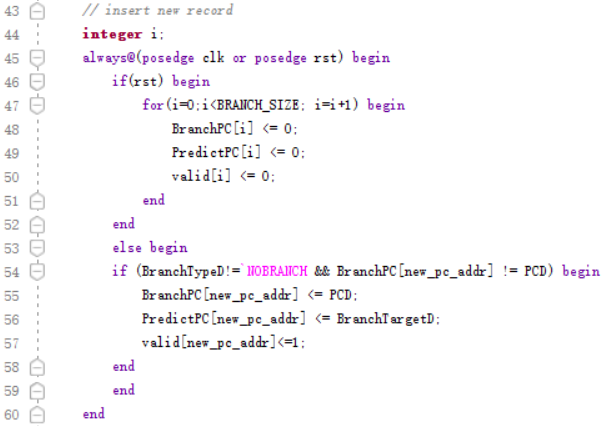


定义BranchPC记录分支指令的地址，PredictPC记录分支指令跳转的地址，valid记录该指令上次是否跳转，BTB总大小为BRANCH\_SIZE。

pc\_addr、new\_pc\_addr、result\_pc\_addr分别为PCF、PCD、PCE的第BRANCH\_ADDR\_LEN+1位到第二位，因为正常对齐取指时PC的第1位和第0位都是0，因此无需记录。其中pc\_addr用于在IF阶段寻址BTB做出分支预测；new\_pc\_addr用于在ID阶段寻址BTB添加新的分支指令；result\_pc\_addr用于在EX阶段寻址BTB更新valid位。

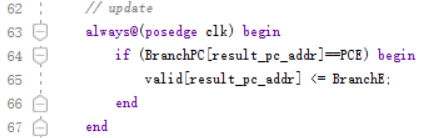


在IF阶段，若PCF值与BTB中相应表项相同且该valid位为1，表示该指令为分支指令且上次跳转，因此预测其跳转并读出相应跳转地址。否则预测其不跳转，取下一条指令地址为PCF+4。



在ID段，如果译码发现指令是分支指令，且其不在BTB中，则将其加入BTB，初始valid置为1。

值得注意的是，此处更新是时序逻辑电路，BranchPC、PredictPC和valid都被期望综合成寄存器，因此使用always@ (posedge clk)。若使用always@(\*)则会将这段电路综合成组合逻辑，与期望不符，实验结果也会产生难以预料的错误。

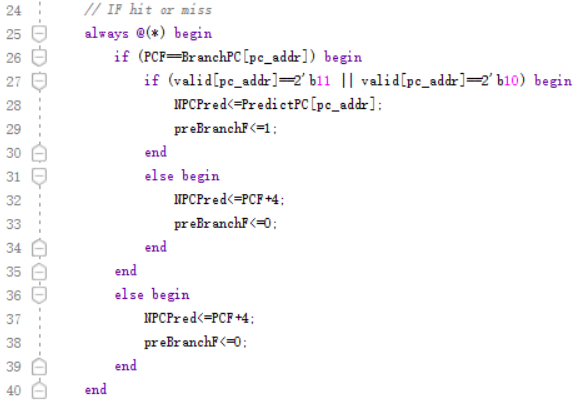


在EX段，若当前指令存在于BTB相应表项中，则记录其本次分支结果BranchE。值得注意的是，此处更新也是时序逻辑电路，原因与上文中ID段的相同。

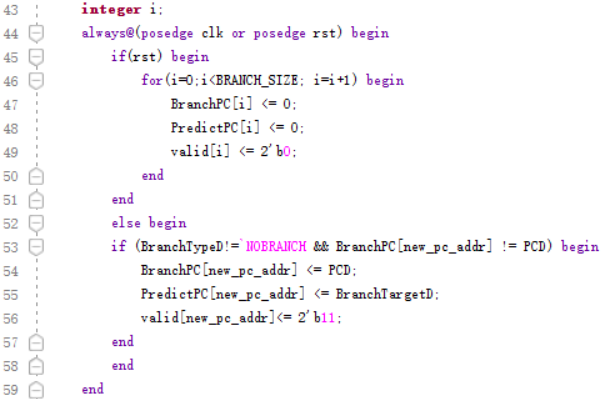
1. **BHT**

BHT主体部分与BTB类似，在本报告中仅叙述其不同之处。

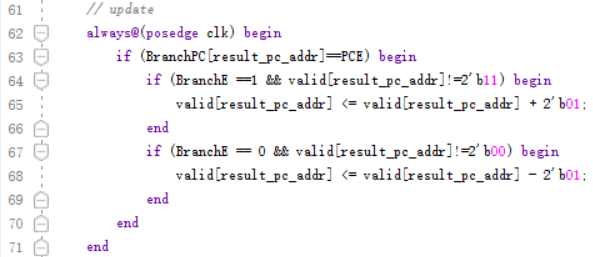
此时valid寄存器组应为2位，表示每一表项的BHT状态机。



在IF段，若valid位为11或10均预测其分支成功。



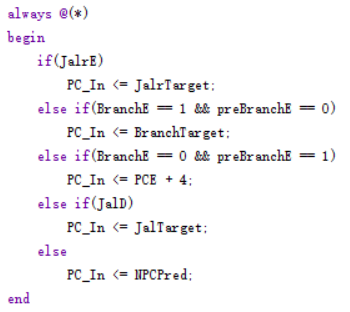
在ID段，插入新表项时valid置为11。



在EX段，更新valid时不仅依据本次分支结果BranchE，也与valid现在的状态相关。此时构成一个分支预测的2位状态机。

1. **其他修改模块**

**NPC\_Generator：**



修改部分：若为分支指令，预测错误分两种情况：预测不跳转实际跳转；预测跳转实际不跳转。PC\_In应与实际情况符合，实际跳转则设为BranchTarget，实际不跳转则设为PCE+4。若既不是J指令也不是预测错的分支指令，则PC\_In设为来自BTB或BHT的预测PC值。

注：对于非J非分支的指令，NPCPred始终为PCF+4，具体操作可见BTB和BHT的实现。

**IDSegReg：**

增加输入preBranchF和输出reg preBranchD，用于记录当前指令预测跳转与否。

**EXSegReg：**

增加输入preBranchD和输出reg preBranchE，用于记录当前指令预测跳转与否。

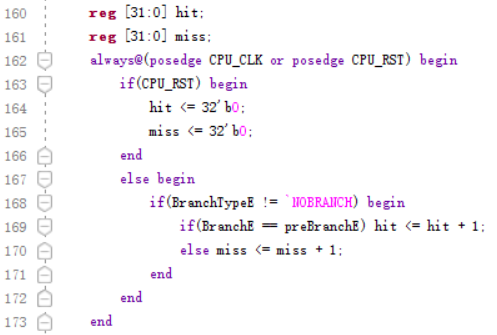
**HarzardUnit：**



改变分支指令时FlushD和FlushE的条件，此时为预测失败时FlushD、FlushE，预测成功时不进行Flush。

**RV32Core：**

除了BTB、BHT和以上修改模块的连线外，增加统计分支预测成功和失败的寄存器。



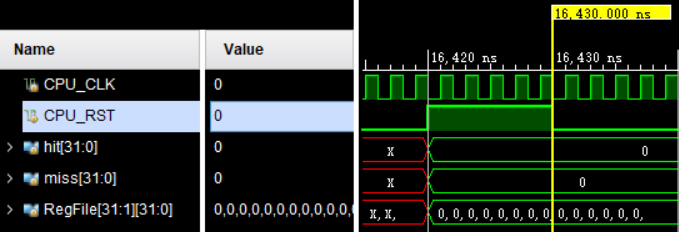
在指令执行到EX时判断是否为分支指令和预测是否成功，注意此处统计时应该先判断EX段的指令是否是分支指令。

1. **实验结果**

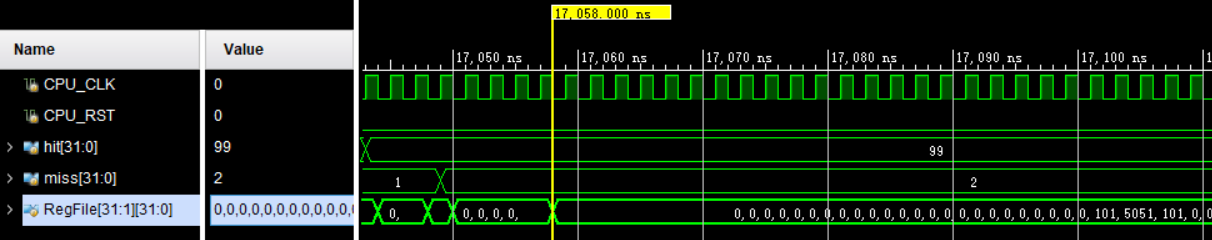
本节展示BTB和BHT两种分支预测分别测试BTB.S和BHT.S的实验结果。

一个周期2ns。

对于所有仿真，CPU\_RST结束的时间都是16430ns



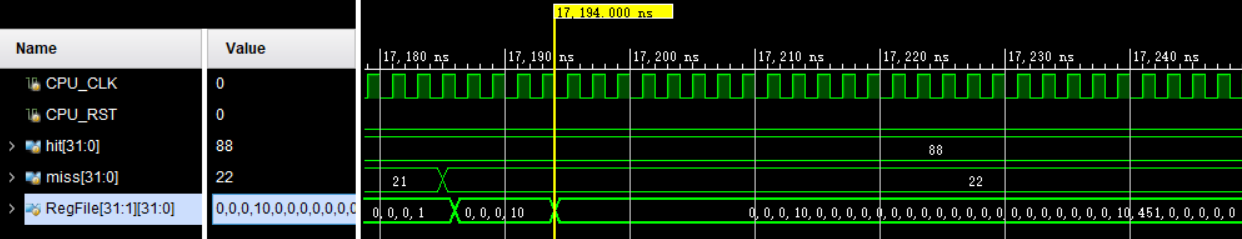
1. **BTB**
   1. **BTB.S**



17058ns结束，总周期数(17058-16430)/2 = 314。

分支预测Hit 99次，Miss 2次。

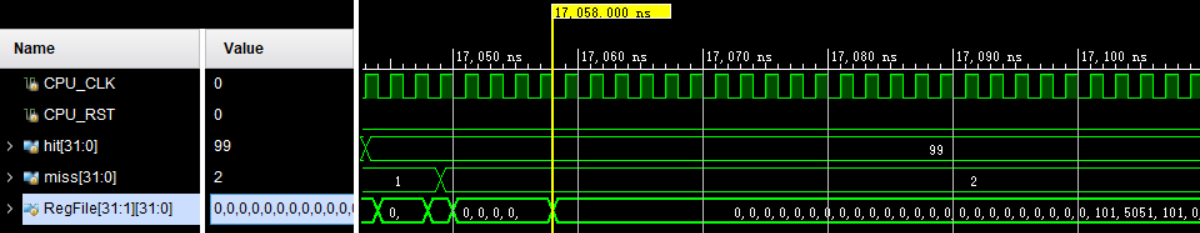
* 1. **BHT.S**



17194ns结束，总周期数(17194-16430) / 2 = 382。

分支预测Hit 88次，Miss 22次。

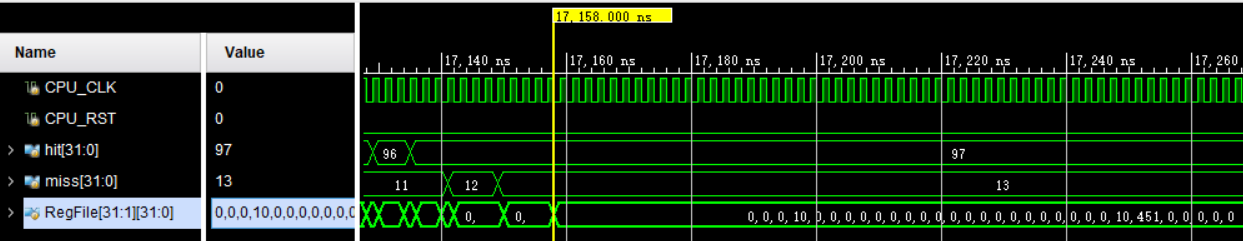
1. **BHT**
   1. **BTB.S**



17058ns结束，总周期数(17058-16430)/2 = 314。

分支预测Hit 99次，Miss 2次。

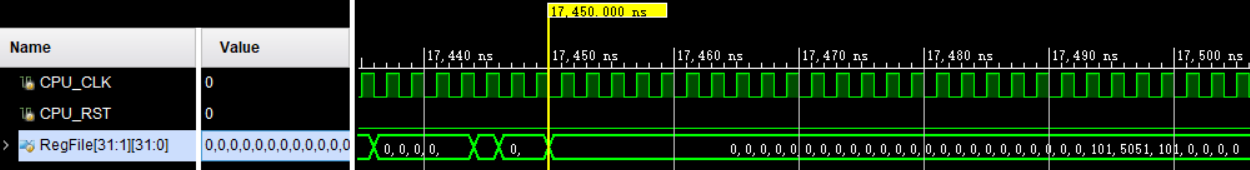
* 1. **BHT.S**



17158ns，总周期数(17158-16430)/2 = 364。

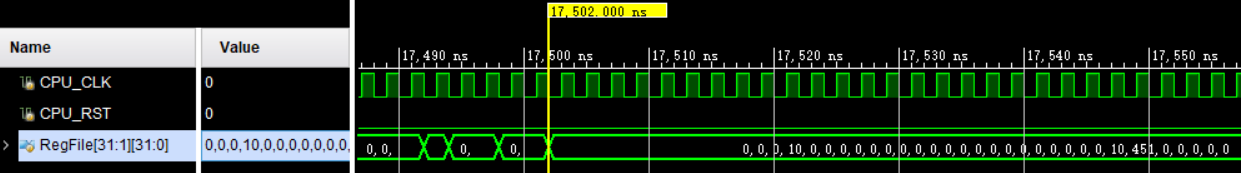
分支预测Hit 97次，Miss 13次。

1. **无分支预测CPU**
   1. **BTB.S**



17450ns结束，总周期数(17450-16430)/2 = 510。

* 1. **BHT.S**



17502ns结束，总周期数(17502-16430)/2 = 536。

1. **实验分析**
2. **BTB.S**

总共307条指令，其中101条分支指令。

表一 BTB.S实验结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Hit数 | Miss数 | 总周期 | 与无分支预测周期数差值 | 加速比 | CPI |
| 无分支预测 | - | - | 510 | - | - | 1.6612 |
| BTB | 99 | 2 | 314 | 196 | 1.6242 | 1.0228 |
| BHT | 99 | 2 | 314 | 196 | 1.6242 | 1.0228 |

1. **BHT.S**

总共335条指令，其中110条分支指令。

表二 BHT.S实验结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Hit数 | Miss数 | 总周期 | 与无分支预测周期数差值 | 加速比 | CPI |
| 无分支预测 | - | - | 536 | - | - | 1.600 |
| BTB | 88 | 22 | 382 | 154 | 1.4031 | 1.1403 |
| BHT | 97 | 13 | 364 | 172 | 1.4725 | 1.0866 |

1. **对比分析**

分支收益：预测成功时减少流水线因为控制相关而导致的Flush，每预测成功一条分支指令可以节约两个周期。

分支代价：本次实验中预测失败时消耗周期数与无预测相同，均为两个周期，可以理解为在仿真中预测失败相较于无预测情况无额外代价。但在实际情况中，预测失败后消耗的周期数往往比无预测情况要多，此时需要评估分支预测带来的收益是否大于代价。

对比表一表二可以发现，对于BTB.S，BTB和BHT的效果一样，因为该测试程序中仅有一个一层循环，BHT的2位状态机预测发挥不出效果。BTB与BHT均为进入第一次循环时有一次预测失败，最后一次循环退出时有一次预测失败。

对于BHT.S，该测试程序中有两重循环，此时可以体现出BHT中2位状态机的效果，BHT比BTB少miss 9次。

1. **实验总结**

总结本次实验遇到的问题：

对于希望将reg综合成寄存器的电路，应写为时序逻辑电路，如always@(posedge clk)。若写为always@(\*)，则会综合成组合逻辑电路。对于BranchPC、PredictPC、valid、hit、miss这些寄存器而言，都应该使用时序逻辑更新，若使用组合逻辑则会产生无法预料的错误结果。

1. **改进意见**

暂无，感谢助教一学期的帮助。