**分支预测Lab4实验报告**

**PB16060130 顾健鑫**

1. **实验目的**

本次实验需要在提供的样板CPU中实现基于BTB和BHT的分支预测器。

1. **实验环境和工具**

本次实验使用的软件环境为Vivado 2017.4的版本。在创建工程项目时，直接导入提供好的模块框架文件并在完成代码的编写后，导入测试文件用于测试，并用Vivado的波形仿真结果作为结果检验依据。

1. **分支预测的具体实现**
2. ***BTB***

本次实验中对CPU中的NPC\_Generator.v，BranchDecisionMaking.v，HazardUnit.v做了修改以实现分支预测，首先介绍对于BranchDecisionMaking.v的修改。

* 1. 对于BranchDecisionMaking模块，将其的BranchE信号做了拓宽处理，变为了两位，分别表示四种情况，2’b00表示预测分支，实际分支；2’b01表示预测不分支（cache命中失败使得预测不分支），实际分支；2’b10表示预测不分支（命中但当前状态为预测不分支），实际分支；2’b11表示预测分支，实际未分支。

这四种不同的BranchE信号分别对应四种情况，BranchE信号回传给NPC\_Generator模块用于状态的更新。

* 1. 对于NPC\_Generator模块，添加了如下输入和输出

input wire [1:0] BranchFlagsF,

input wire [2:0] BranchIndexF,

input wire [1:0] BranchE,

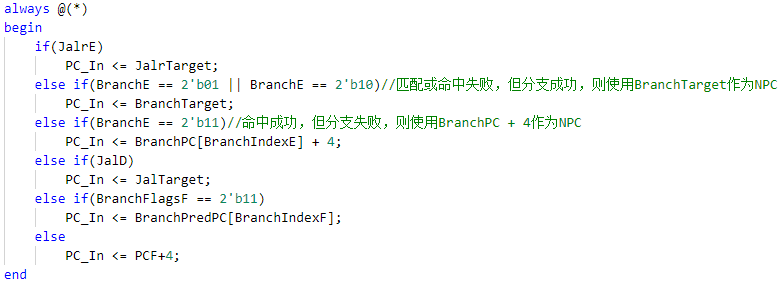
input wire [2:0] BranchIndexE,

output reg [1:0] BranchFlags,

output reg [2:0] BranchIndex

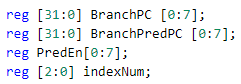
BranchFlags为两位E和F表示为哪一个流水段的信号，其0号位表示是否在cache命中，1为命中，0为缺失；在命中的情况下，其1号位表示预测结果是否为跳转，1为预测分支，0为预测不分支。

BranchFlagsF与BranchE共同决定下一条指令的取指方式，具体的程序逻辑参见下图中的代码以及注释



当BranchE的信号不是00时，表示预测失败，要做相应的回复操作，01和10对应的情况都需要将BranchDecisionMaking模块传回的BranchTarget作为跳转地址；当为11时，要将在Ex段的跳转指令原本的PC+4作为NPC；在以上情况均没有发生时，如果IF段的BranchFlagsF的值为11，则表示预测跳转，需要将cache中对应的跳转地址作为NPC。

为了做Branch指令与其跳转地址的存储，需要使用cache，在此次实验中，使用的为大小为可以存储8个branch条目的基于FIFO替换策略的cache，对于BTB中具体涉及到的代码如下，



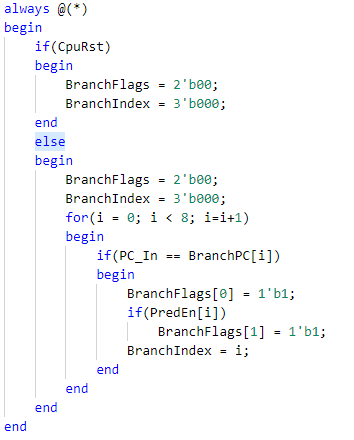
IndexNum为3位的寄存器变量，循环加1来实现FIFO的替换策略，PredEn则为8位的寄存器，每一位的值表示对应的条目的预测结果，BranchPC和BranchPredPC则一一对应的存储分支指令的PC和其跳转地址的PC。

output reg [1:0] BranchFlags,

output reg [2:0] BranchIndex

上述两个寄存器变量的更新使用组合逻辑，并用for循环的方式将当前PC和BranchPC中的值进行一一比较，命中则将BranchFlags[0]置1，在此情况下，若对应的PredEn为1，则将BranchFlags[1]置1。并将BranchIndex设置为此条目的索引值。这两条信号会随流水段传至Ex段并经过判断后，将BranchE和BranchIndex传回NPC\_Generator以便于预测失败的恢复操作。

具体的实现代码如下，



PredEn的更新采用时序逻辑实现，因此NPC\_Generator模块也加入了时钟信号作为输入，分为以下三种情况

* + 1. 命中失败但分支成功：此时应加入相应的BranchPC作为新条目；
    2. 命中成功预测不分支但分支成功：此时应依据BranchIndexE更新PredEn的值（改为0）；
    3. 命中成功预测分支但分支失败：此时应依据BranchIndexE更新PredEn的值（改为1）；

具体的实现代码如下所示



* 1. HazardUnit模块修改相对简单，即当BranchE不是00时，要Flush相应的流水段，实际情况和JalrE需要Flush的流水段相同，因此实现如下

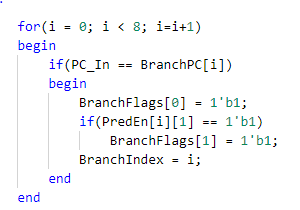
else if(BranchE != 2'b00 | JalrE) {StallF,FlushF,StallD,FlushD,StallE,FlushE,StallM,FlushM,StallW,FlushW} <= 10'b0001010000;

1. ***BHT***

BHT的实现逻辑与BTB完全相同，共用同一套结构，仅需做微小的调整。

首先由于状态的增加PredEn需要从一位变成两位；其次由于此时预测成功也可能需要更新状态，但BranchE取值为00不一定表示为预测成功，可能为无事发生的情形，因此要将BranchFlagsE也进行回传，并对其0号位进行判断是否有命中cache以确定是否为预测成功。

在进行预测判定时，需进行微调，根据状态图，当PredEn[1]为1时，预测分支，因此代码实现如下



在更新状态时，状态相较于BTB有所增加，即不仅仅是预测失败需要进行状态更新，预测成功也需要状态更新，依据上述的实现以及状态转移图，实现如下：在BranchE为00时，需做额外的判断，剩余三种情况类比BTB，但需考虑两种不同情况分别进行不同的状态转移。

begin

     if(BranchE == 2'b00)

     begin

     if(BranchFlagsE[0] == 1'b1)//是预测成功

     begin

     if(PredEn[BranchIndexE] == 2'b01)

     PredEn[BranchIndexE] <= 2'b00;

     else if(PredEn[BranchIndexE] == 2'b10)

     PredEn[BranchIndexE] <= 2'b11;

     else

     PredEn[BranchIndexE] <= PredEn[BranchIndexE];

     end

     end

else if(BranchE == 2'b01)//匹配失败，但分支成功，加入新的表项

begin

BranchPC[indexNum] <= PCE;

BranchPredPC[indexNum] <= BranchTarget;

PredEn[indexNum] <= 2'b10;

indexNum <= indexNum + 1;

end

else if(BranchE == 2'b10)//匹配成功，命中失败，但分支成功，则需将该表项使能

begin

if(PredEn[BranchIndexE] == 2'b00)

PredEn[BranchIndexE] <= 2'b01;

else if(PredEn[BranchIndexE] == 2'b01)

PredEn[BranchIndexE] <= 2'b11;

else

PredEn[BranchIndexE] <= PredEn[BranchIndexE];

end

else if(BranchE == 2'b11)//匹配成功，命中成功，但分支失败，需将该表项置为无效

begin

if(PredEn[BranchIndexE] == 2'b10)

PredEn[BranchIndexE] <= 2'b00;

else if(PredEn[BranchIndexE] == 2'b11)

PredEn[BranchIndexE] <= 2'b10;

else

PredEn[BranchIndexE] <= PredEn[BranchIndexE];

end

end

1. **实验结果**

经过仿真测试，本次实验的两种分支预测都可以正常的运行测试代码，并得到正确的运行结果，不会由于加入分支预测而计算出错误的结果。

两种分支预测器分别运行两种测试代码得到的预测成功与失败次数如下：对于BTB，当加入新表项时，状态设置为1，即预测分支；对于BHT，当加入新表项时，状态设置为10，也是预测分支，具体统计结果如下，

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 测试样例 | 分支总数 | 预测失败 | 预测成功 | 预测成功率 | 总时长(ns) |
| BTB | btb | 101 | 2 | 99 | 98.02% | 1266 |
| bht | 110 | 22 | 88 | 80.00% | 1538 |
| BHT | btb | 101 | 2 | 99 | 98.02% | 1266 |
| bht | 110 | 13 | 97 | 88.18% | 1466 |

1. **实验结果分析**

通过观察实验统计结果，可以得到以下结论

* 1. 分支预测不会影响结果的正确性；
  2. BHT的效果会优于BTB；
  3. 分支预测的成功率越高，则程序运行的时间也越短；

对于BHT优于BTB的结论，是因为bht的测试代码中含有两层循环，当内层循环第一次结束后（即一次分支失败），BHT状态变为10，BTB则变为0,，故在下一次循环到达此处时，BHT预测分支，但BTB预测不分支并在一次预测失败后改为预测会分支。在对于外层循环，两者则应表现一致，即第一次命中失败加入表项后，一直预测成功，直到跳出循环结束程序会有一次预测失败。

1. **实验心得与总结**

通过本次实验，我实现了BTB与BHT两种分支预测方式，对其具体的结构以及预测原理有了更加深刻的认知，并通过对运行结果的分析，更加切实的认识到了分支预测带来的性能上的提升。