CAlab4 实验报告

- **PB18071569**
- **张万林**

实验目的

- 实现B（Branch Target Buffer）和BHT（Branch History Table）两种动态分支预测器
- 了解动态轨迹预测对性能的影响

实验内容

- 阶段一：在Lab3阶段二的RV32I Core基础上，实现BTB
  - 我们要实现的BTB本质可以理解为是1bit预测器，如果上次这条分支指令跳转，那么这次它也跳转；如果上次不跳，那么这次也不跳
  - BTB实现一个buffer，保存当前地址高位、目标地址和有效位，类似于直接映射的cache，可以直接使用reg实现buffer
  - **buffer 放在取指阶段**，buffer内容读取一个周期内可以完成
  - BTB的命中：当前指令的地位用于寻址，对比指令的高位和buffer中是否相等并且有效位为1，表示命中，则下一条指令的地址不是pc+4，而是buffer中的内容
  - 在IF阶段是否命中信息会随着流水线段寄存器传递到EX阶段，根据实际是否跳转和IF阶段是否命中信息，**在EX阶段对buffer进行修改**
- 阶段二：实现BHT
  - BHT 首先要实现一个N*2的buffer，N为大小，2表示2bit预测
  - 实现一个状态机
  - 用BHT来控制是否跳转（BTB不命中，BHT命中该如何处理？），BHT的根据状态机更新，BTB的更新与之前不同
- 阶段二： **需要在阶段一的基础上实现**，不能仅实现阶段二

实验过程

BTB

设计思路： BTB 在IF阶段进行预测，在EX阶段进行判断预测结果， 以及更新Buffer 。预测方法是：直接读BTB_Buffer看是否命中， 命中则NPC从BTB取， 否则正常取。检查正确性的方法是 比较PPCE与BrNPC。 更新的方法是，根据预测结果。分几种情况更新即可。

1.预测

- 当前指令的地址用于寻址，对比指令的高位和buffer中是否相等并且有效。{ rbtb_tag, rbtb_addr } = PCF, 其中 len(rbtb_addr) = btb_addr_len，而 BTB_SIZE = 1 << btb_addr_len，即 PC 的低位 PC[btb_addr_len-1 : 0] 作为地址，高位作为 tag。
- NPC

```
else
begin
    if(BTBF)
        PC_In <= Pred_BranchTarget;
    else
        PC_In <= PCF+4;
end
```

2.更新

- 检验预测的正确性

```
assign Pred_True = (PPCE==BrNPC)? 1: 0;
assign BTB_Update = BranchE ? (BTBE ? (Pred_True ? 2'b00 : 2'b01) : 2'b10) : (BTBE
? 2'b11 : 2'b00);
```

- 对预测正确/失败 ,分别处理。

  若失败需要进行Flush。

```
else if(BranchE)
            begin
                if(Pred_True)
                begin
                    StallF = 0;
                    FlushD = 0;
                    StallD = 0;
                    FlushE = 0;
                    StallE <= 0 ;
                    StallM <= 0 ;
                    StallW <= 0 ;
                end
                else
                begin
                    StallF = 0;
                    FlushD = 1;
```

```
                    StallD = 0;
                    FlushE = 1;
                    StallE <= 0 ;
                    StallM <= 0 ;
                    StallW <= 0 ;
                end
            end
```

根据成功失败确定NPC

```
else if(BranchE)
begin
    if(Pred_True)
        PC_In <= PCF+4;       //accurately predicted
    else
        PC_In <= BranchTarget;  //mispredicted
end
else if(~BranchE & BTBE)      //should not branch but branched
    PC_In <= PCE+4;
```

- 更新BTB_Buffer

```
    case(web)
        2'b01:  begin   //need to update branch target
                pred_pc[wbtb_addr] = wr_data;
                end
        2'b10:  begin   //need to add entry
                pred_pc[wbtb_addr] = wr_data;
                btb_tags[wbtb_addr] = wtag_addr;
                valid[wbtb_addr] = 1'b1;
                end
        2'b11:  begin   //need to remove entry(invalidate entry)
                valid[wbtb_addr] = 1'b0;
                end
   endcase
```

BTB+BHT

设计思路： BTB 部分不变 。 BHT 部分是实现了一个两位预测器的状态机 。状态机控制产生 是否进行分支的信号 BHT_br 。 该信号和BTB_hit组合产生最终的分支预测信号。其他部分逻辑与只有BTB一样。

1.BHT_br产生逻辑

```
//raddr = PCF[bht_addr_len-1:0]
always @ (*) begin                              //read data
    if((pred_states[raddr]==2'b11) || (pred_states[raddr]==2'b10))
        pred_taken = 1'b1;
    else
        pred_taken = 1'b0;
end
```

状态机

```
//waddr = PCE[bht_addr_len-1:0]
else begin
    if(BranchE)
        case(pred_states[waddr])  //BHTE = pred_states[waddr]
        2'b11:      pred_states[waddr]=2'b11;
        2'b10:      pred_states[waddr]=2'b11;
        2'b01:      pred_states[waddr]=2'b10;
        2'b00:      pred_states[waddr]=2'b01;
        endcase
    else
        case(pred_states[waddr])
        2'b11:      pred_states[waddr]=2'b10;
        2'b10:      pred_states[waddr]=2'b01;
        2'b01:      pred_states[waddr]=2'b00;
        2'b00:      pred_states[waddr]=2'b00;
        endcase
```

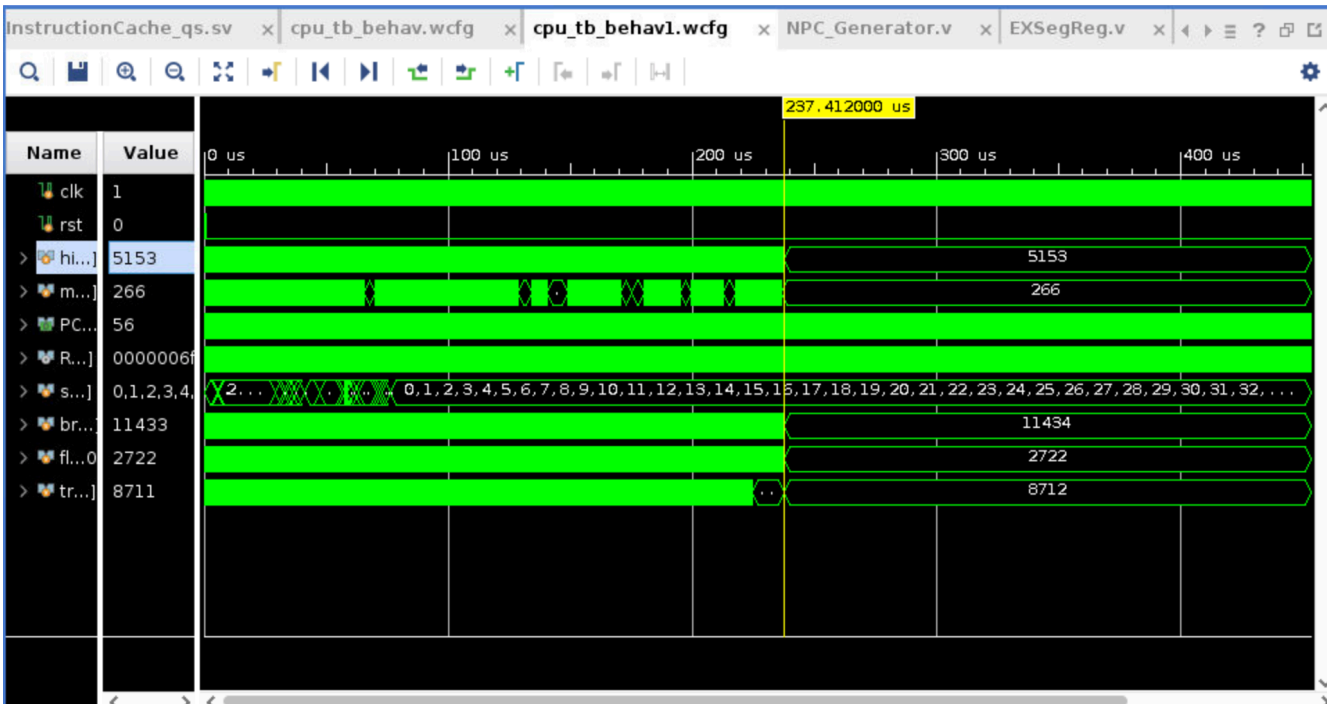
加入BHT后的更新信号

```
    assign BTB_Update = BranchE ? (BTB_hitE ? 2'b00  : 2'b10) : (BTB_hitE ? ( BHT_brE?
2'b00:2'b11): 2'b00);
    //01 need to update branch target
    //10 need to add entry
    //11 need to remove entry
```

实验结果分析

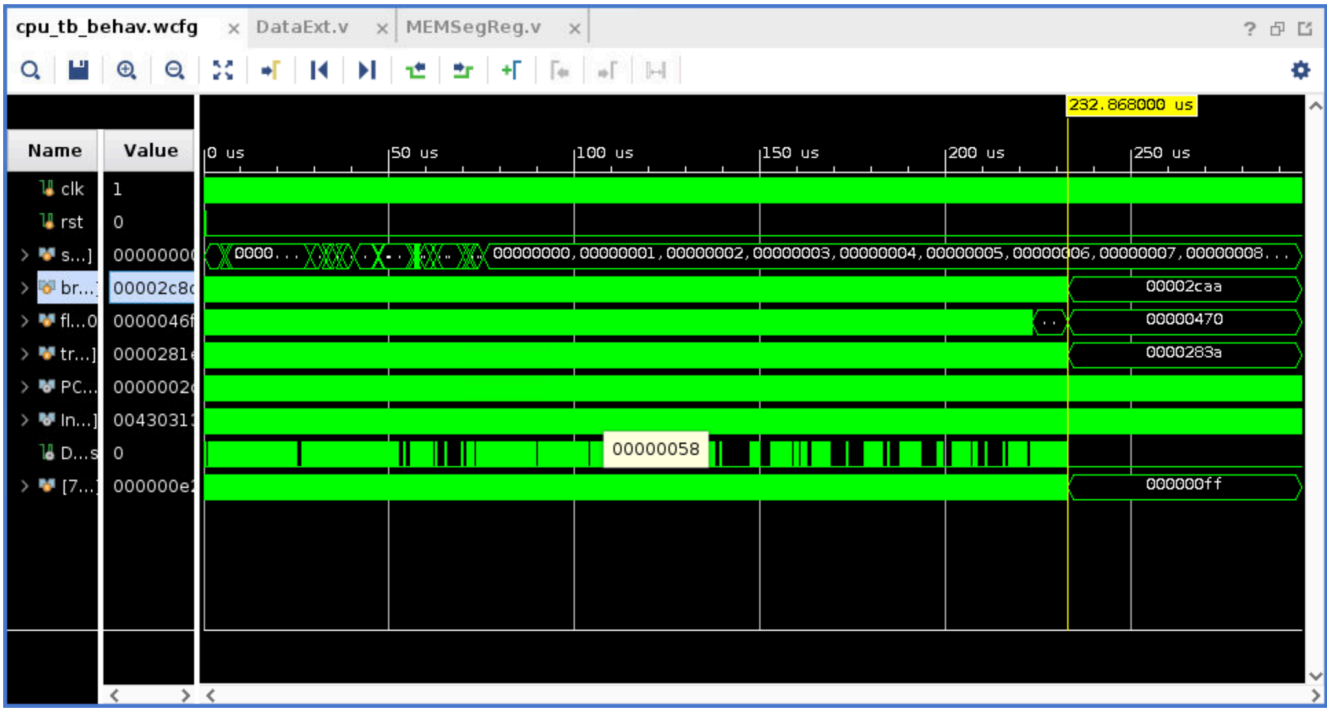
快排256规模

- 不带分支预测



  - 周期数：$\frac{237412}{4} = 59453$
  - 分支路径数量：11434
  - 预测正确次数：8712
  - 错误次数：2722

- 带分支预测的结果

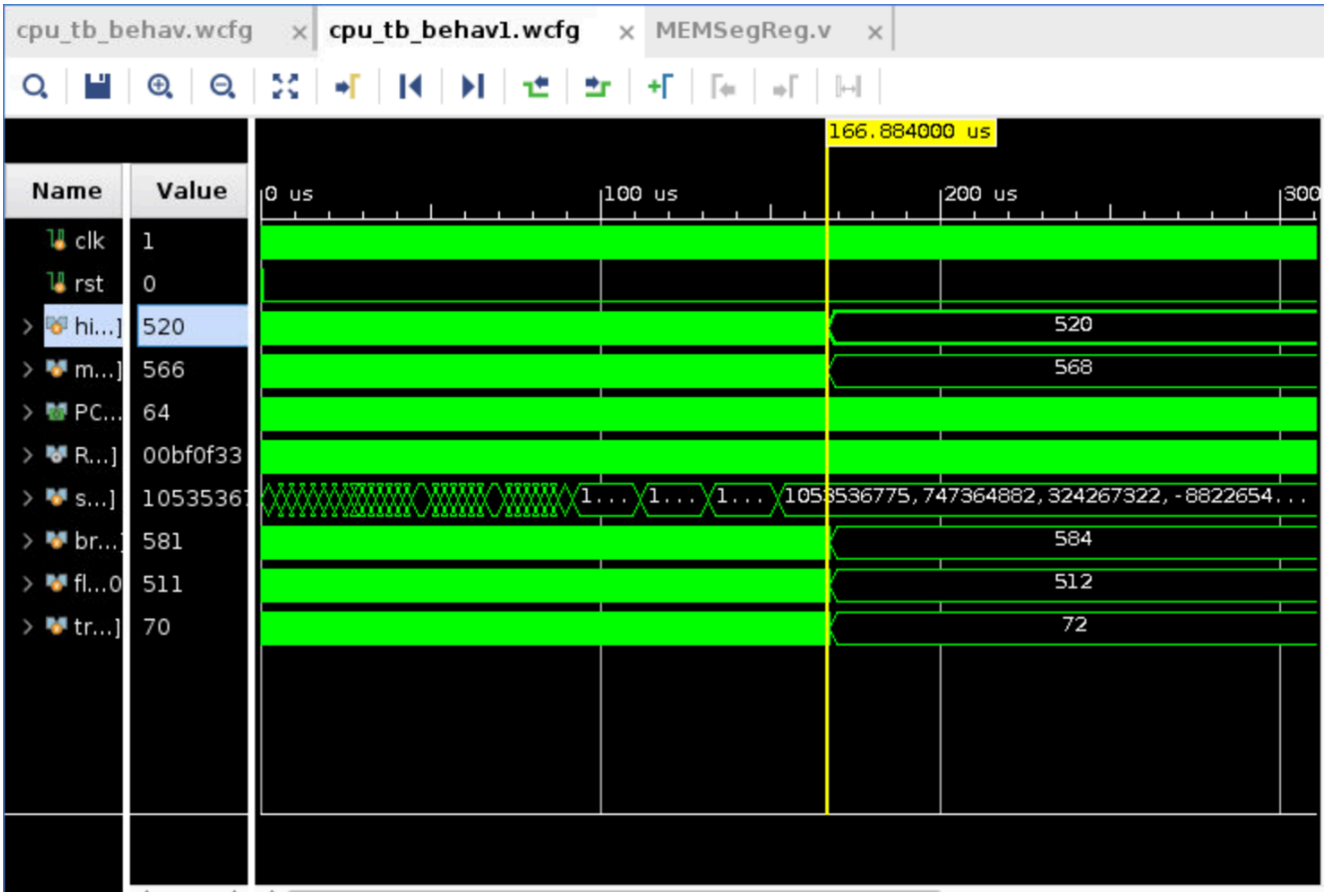


  - 周期数：$\frac{233216}{4} = 58304$
  - 分支路径数量：11434
  - 预测正确次数：10298
  - 错误次数：1136

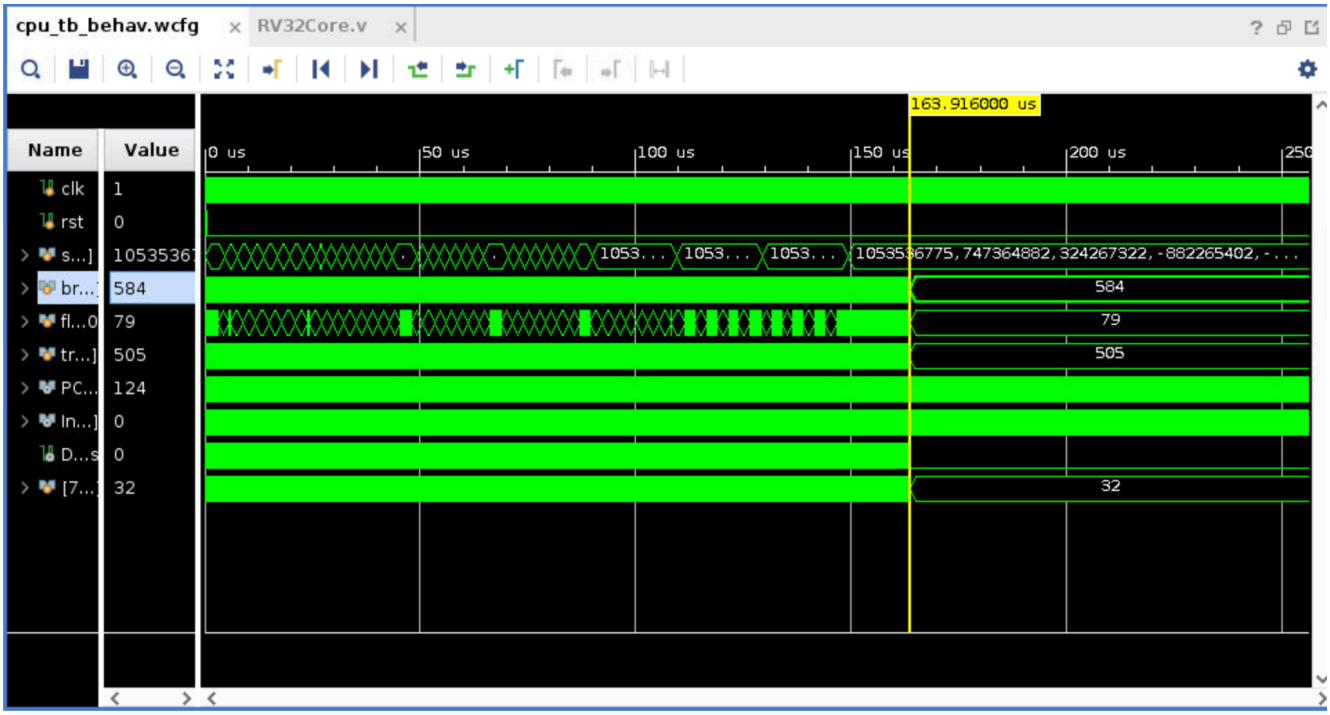
分析： 总周期数减少了$59453-58304=1149(cycle)$ 个周期 。 并且命中率也有所提高

矩阵乘 N=8规模

- 不带分支预测



  - 周期数：$\frac{166884}{4}=41721$
  - 分支路径数量：584
  - 预测正确次数：72
  - 错误次数：512
- 带分支预测的结果

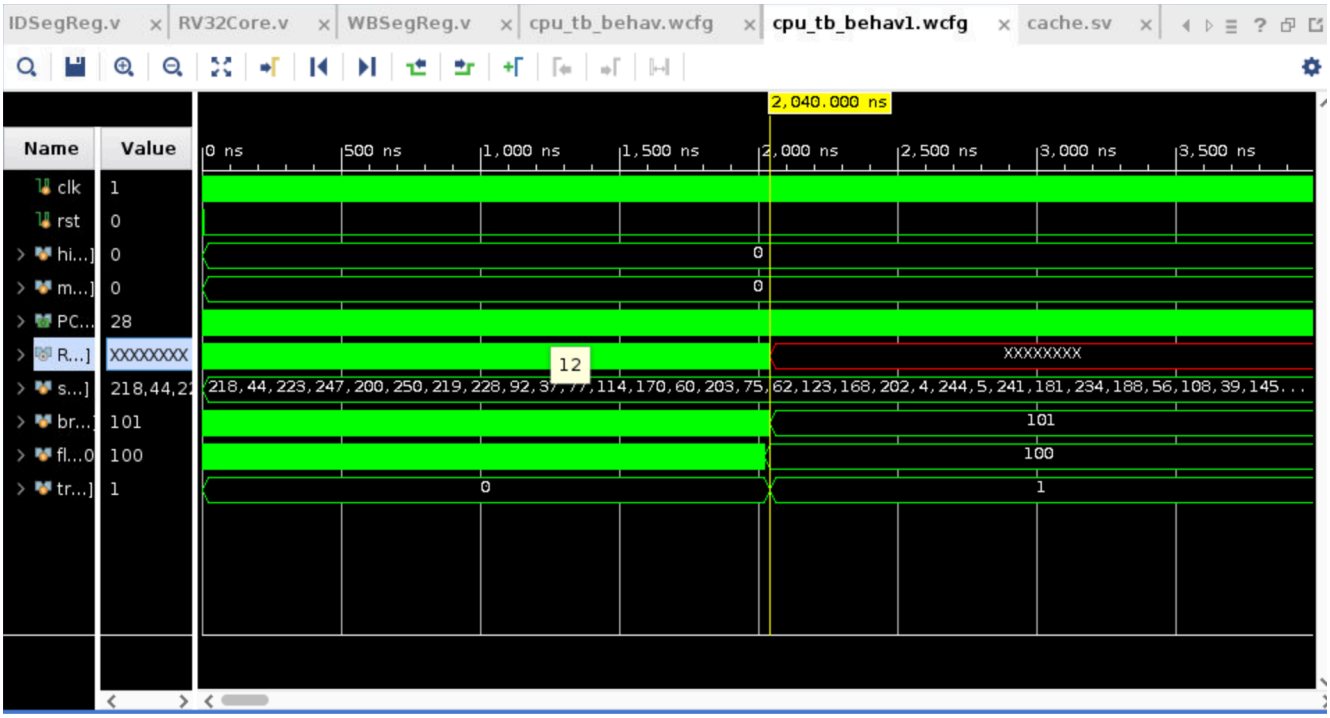


- 周期数：$\frac{163916}{4} = 40979$
- 分支路径数量：584
- 预测正确次数：505
- 错误次数：79

分析： 总周期数减少了$41721 - 40979 = 721(cycle)$ 个周期 。 并且命中率大大 提高。

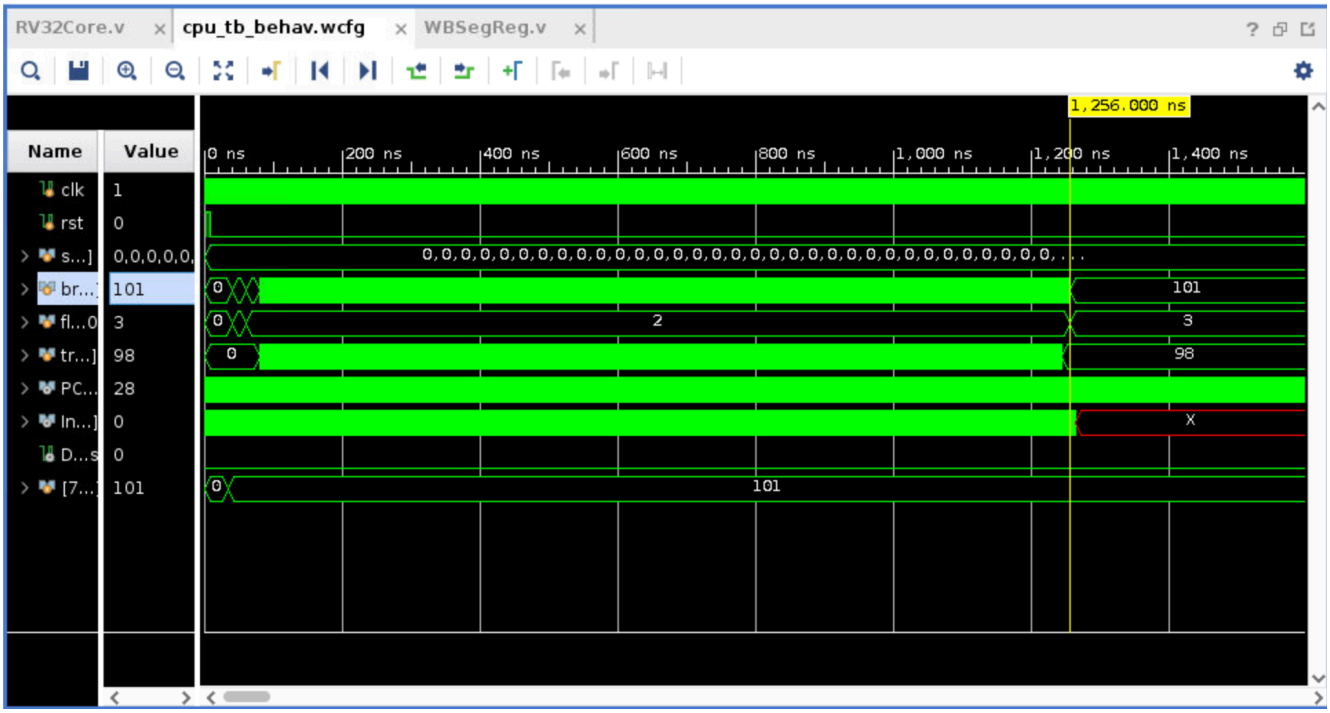
BTB.s

- 不带分支预测



- 周期数：$\frac{2040}{4} = 510$
- 分支路径数量：101
- 预测正确次数：1

    - 错误次数：100
- 带分支预测

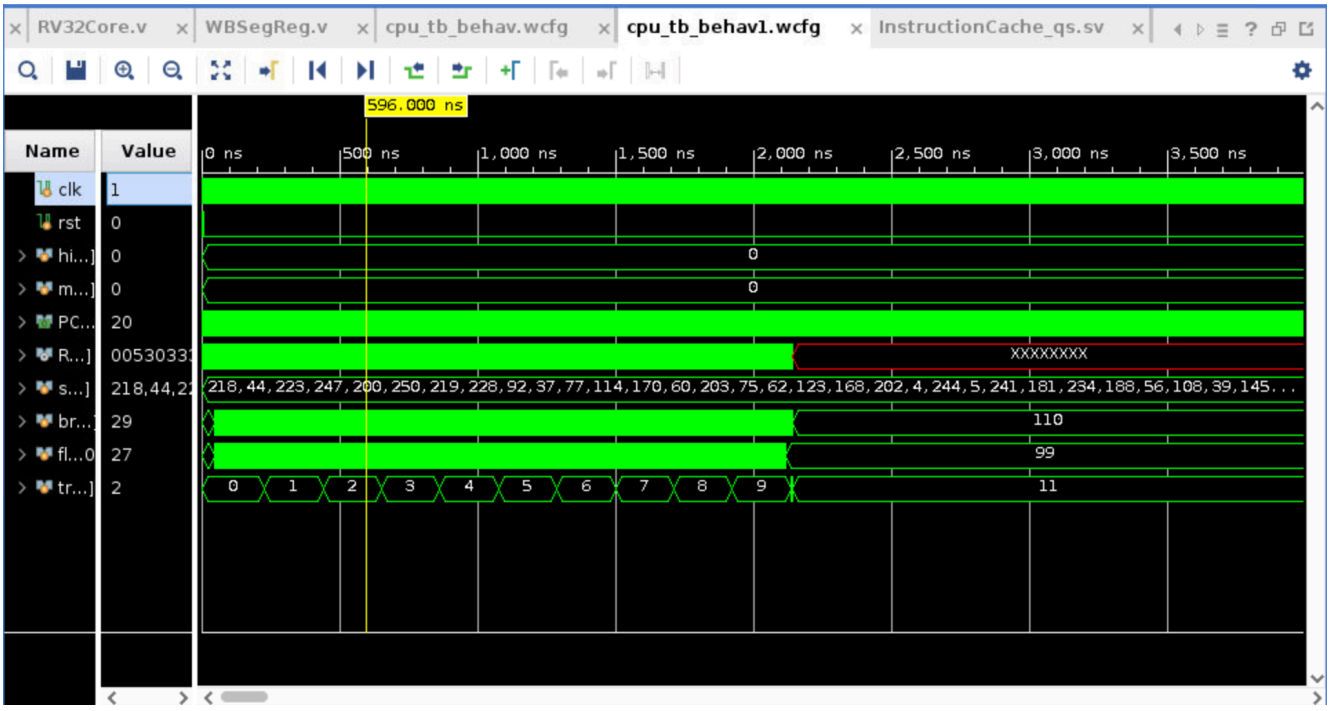


    - 周期数：$\frac{1256}{4}=314$
    - 分支路径数量：101
    - 预测正确次数：98
    - 错误次数：3

分析： 总周期数减少了$510-314=196(cycle)$ 个周期 。 并且命中率大大提高。

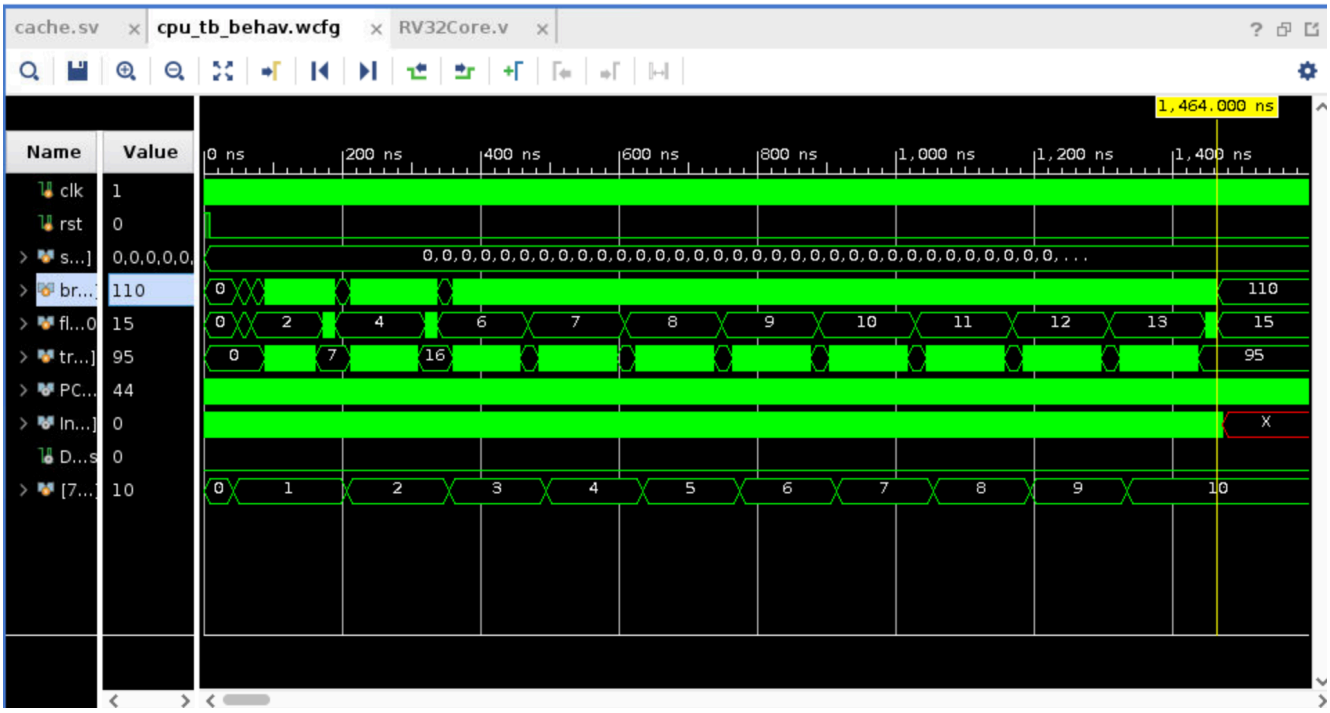
BHT.s

- 不带分支预测



    - 周期数：$\frac{2144}{4}=536$

    - 分支路径数量：110
    - 预测正确次数：11
    - 错误次数：99
- 带分支预测



    - 周期数：$\frac{1464}{4} = 366$
    - 分支路径数量：110
    - 预测正确次数：95
    - 错误次数：15

分析： 总周期数减少了$536 - 366 = 170(cycle)$ 个周期 。 并且命中率大大提高。

结果分析

通过上面的统计数据可以明显看出，周期数都有明显地减少。原因是预测命中率提高 。 四个加速比分别为：

$S_1 = \frac{59453}{58304} = 1.0197$

$S_2 = \frac{41721}{40979} = 1.0181$

$S_3 = \frac{510}{314} = 1.624$

$S_4 = \frac{536}{366} = 1.4644$

实验总结

本次实验进行了分支预测的实现 ， 体会到了分支预测对性能的提升。实验中BTB和BHT都不是很难 ， 难的是把BTB和BTH连接起来 。 刚开始由于代码结构混乱导致 两者结合的代码很难处理 。后来将所有的信号都加到段寄存器，一级一级往后串。这样结构就清晰多了。

最终的实验结果也是比较好的，获得了不错的加速。