Verilog 实验 lab4 实验报告

PB19071405 王昊元

2022 年 05 月 16 日

1 实验目的

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

2 实验要求

1. 在 Lab3 阶段二的 RV32I Core 基础上，实现 BTB
2. 在阶段一的基础上实现 BHT

3 实验环境

- vlab
- Vivado 2016.3

4 实验核心实现

1. 预测器主要的顶层设计逻辑是，在 EX 检查（检查部分的实现由一条语句在 CPU 最顶层实现）是否为 Br 指令，只有当是 Br 指令且预测失败时，由真正的 Br 信号确定 PC，其余情况都由预测的信号确定（其余的情况包括不是 Br 指令和预测正确的情况）。

```
always@(*)
begin
    if(need_update & (br_pred_out != is_br))
    begin
        branch_prediction_miss = 1;
        // if branch prediction fail, it'll depend on true br signal
        NPC = is_br ? PC_target : PC_out;
    end
    else
    begin
        branch_prediction_miss = 0;
        NPC = br_pred_in ? PC_pred : PC_query + 4;
```

```
    end
end
```

2. 实现一个记录预测结果和 PC+4 的 buffer，长度为 2，为了辅助在 EX 阶段的判断（存 PC+4 是为了在预测失败时且实际不跳转时使用）。

3. 实现 BTB（1bit 预测器），主要实现状态机的转换（如下），是否命中、是否跳转及预测的 PC 等根据状态机的信息输出即可。

```
integer i;
always@(posedge clk or posedge rst)
begin
    if(rst)
    begin
        for(i = 0; i < SET_SIZE; i = i + 1)
        begin
            TAG[i] <= 0;
            DATA[i] <= 0;
            VALID[i] <= 0;
            STATE[i] <= 0;
        end
    end
    else if(need_update)
    begin
        TAG[addr_update] <= tag_update;
        DATA[addr_update] <= update_data;
        VALID[addr_update] <= 1'b1;
        STATE[addr_update] <= is_br;
    end
end
```

4. 实现 2bit BHT，主要实现状态机（参考课程 ppt，00 和 01 表示跳转，10 和 11 表示不跳转，实现如下），是否跳转由所处状态决定。

```
integer i;
always@(posedge clk or posedge rst)
begin
    if(rst)
    begin
        for(i = 0; i < SET_SIZE; i = i + 1)
        begin
            STATE[i] <= 0;
        end
    end
    else
    begin
        if(need_update)
        begin
```

```
            if(is_br)
            begin
                STATE[addr_update] <= (STATE[addr_update] == 0 ? 0 : STATE[
                    addr_update] - 1);
            end
            else
            begin
                STATE[addr_update] <= (STATE[addr_update] == MAX_VAL ? MAX_VAL :
                    STATE[addr_update] + 1);
            end
        end
    end
end
```

5 实验结果及分析

5.1 分析分支收益和分支代价

- 分支收益：一次分支预测命中可带来 2 个时钟周期的收益
- 分支代价：预测失败没有收益，代价为 2 个时钟周期（EX 阶段检验，会影响 IF 和 ID 段已经执行的部分）

5.2 统计未使用分支预测和使用分支预测的总周期数及差值

	btb.s	bht.s	QuickSort.s	MatMul.s
无分支预测	512	538	41859	297795
BTB 分支预测 (差值)	316(196)	382(156)	41932(-73)	295727(2068)
BHT+BTB 分支预测 (差值)	318(194)	370(168)	41156(703)	294182(3613)

5.3 统计分支指令数目、动态分支预测正确次数和错误次数

	btb.s	bht.s	QuickSort.s	MatMul.s
分支指令数目	101	110	10548	4896
BTB 分支预测正确次数/错误次数	99/2	88/22	8479/2069	4092/804
BHT+BTB 分支预测正确次数/错误次数	98/3	95/15	8761/1787	4350/546

5.4 对比不同策略并分析以上几点的关系

1. 使用动态分支预测通常会带来正收益。
2. 但在某些情况下收益不高甚至会产生负收益。例如在 QuickSort 测试中，简单 BTB 预测反而会增大运行周期数。
3. BHT 分支预测效果通常好于 BTB。

4. 不同测试样例上不同分支结构带来的运行周期数优化效果差别显著，表明程序性能优化与程序本身（或者程序类型）有较强的相关性。

6 实验总结

1. 在本次实验中，实现了 BTB 和 BHT 动态分支预测，对动态分支预测的原理也有了更深的理解。
2. 通过各种程序测试和指标分析，体会到动态分支预测对程序性能的实际优化效果。