浙江大学

本科实验报告

课程名称：	计算机体系结构
姓 名：	张志心
学 院：	竺可桢学院
专 业：	混合班
学 号：	3210106357
指导教师：	常瑞
日 期：	2023 年 12 月 7 日

浙江大学实验报告

课程名称： 计算机体系结构 实验类型： 综合

实验项目名称： 实验2 - 流水线异常和中断设计

学生姓名： 张志心 专业： 计算机科学与技术 学号： 3210106357

同组学生姓名： 无 指导教师： 常瑞 助教： 邱明冉

实验地点： 曹光彪西301 实验日期： 2023 年 12 月 7 日

1 实验目的

- 了解分支预测原理
- 实现以 **BHT** 和 **BTB** 为基础的动态分支预测

2 实验环境

- **HDL**：**Verilog**、**SystemVerilog**
- **IDE**：**Vivado**
- **开发板**：**NEXYS A7**（**XC7A100TCSG324**）

3 实验要求

1. 实现用 **BTB** 和 **BHT** 做动态分支预测
2. 通过仿真测试和上板验证
3. 验收要求指出使用了 **BTB** 和 **BHT** 的跳转指令位置，展示 **PC** 的变化
4. 报告要求给出跳转预测成功和失败的分析

4 实验步骤

- **BHT / BTB** 部分

```
`ifndef WIDE
  `define WIDE(len) [len-1:0]
`endif
module bp_unit (
  input rst,
  input clk,
  input wen,
```

```
  input `WIDE(12) pc_w,
  input `WIDE(12) pc_r,
  input `WIDE(32) addr_in,
  output `WIDE(32) addr_out,
  input taken_w,
  output taken_r
);

  reg `WIDE((2**12)) `WIDE(32) btb;
  reg `WIDE((2**12)) `WIDE(2) bht;

  wire nouse;
  assign {taken_r, nouse} = bht[pc_r];
  // assign taken_r = 1'b0;

  wire [1:0] watch = bht[12'h220];

  assign addr_out = btb[pc_r];
  integer i;
  always @(posedge clk, posedge rst) begin
    if (rst) begin
      for (i = 0; i < 2**12; i = i + 1) begin
        bht[i] <= 2'b01;
        btb[i] <= 32'b0;
      end
    end else if (wen) begin
      if (taken_w) begin
        btb[pc_w] <= addr_in;
      end
      if (taken_w) begin
        if (bht[pc_w] != 2'b11)
          bht[pc_w] <= bht[pc_w] + 1;
      end else begin
        if (bht[pc_w] != 2'b00)
          bht[pc_w] <= bht[pc_w] - 1;
      end
    end
  end


endmodule
```

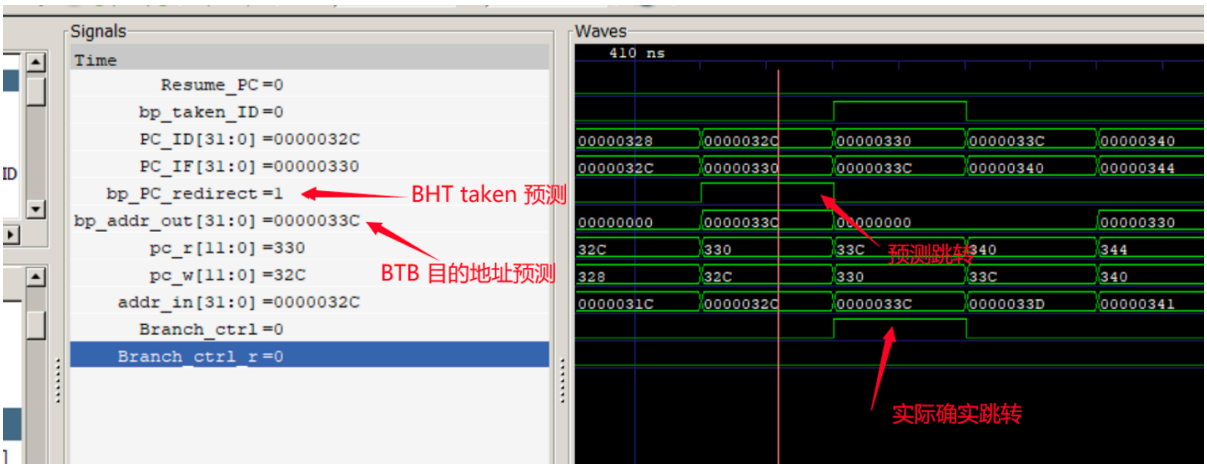
- 在 `RV32core.v` 中加入 `bp_unit` 模块

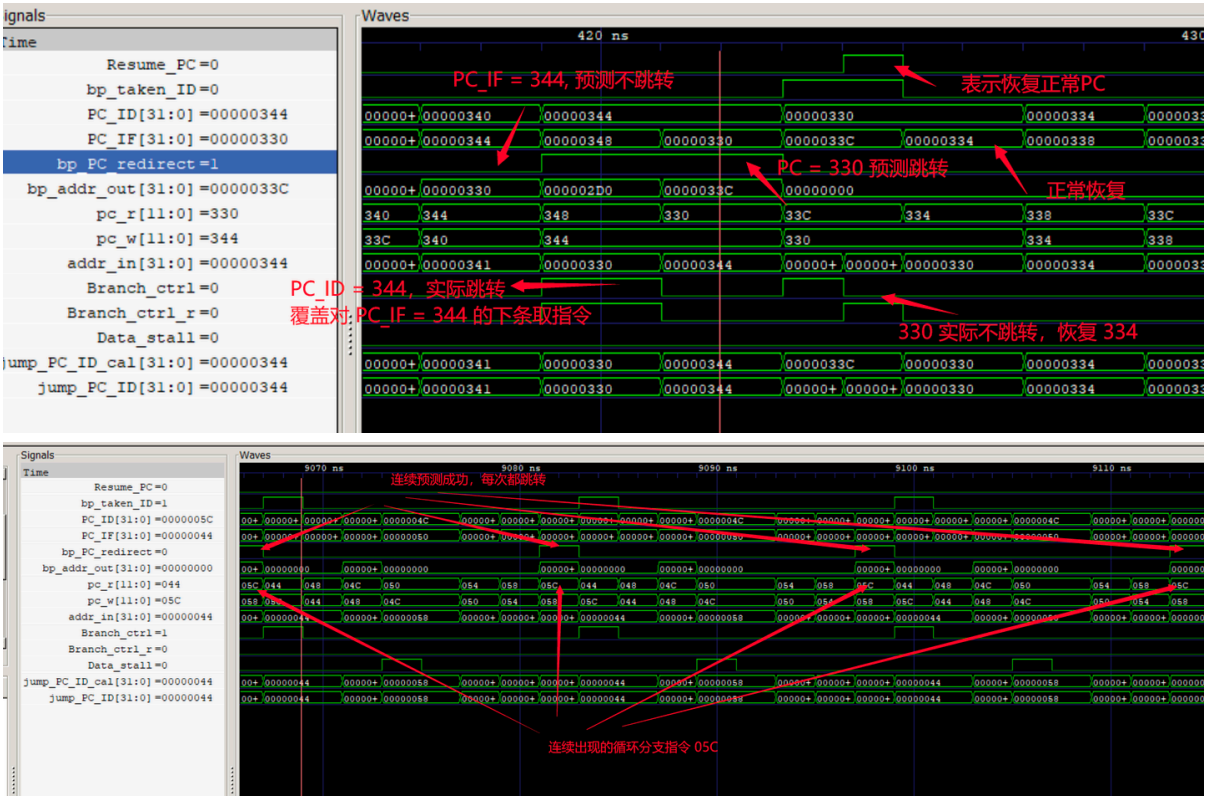
```
bp_unit bp_unit_0 (
        .clk(clk),
        .rst(rst),
        .pc_r(PC_IF[11:0]),
        .pc_w(PC_ID[11:0]),
        .addr_in(jump_PC_ID),
        .wen((~isFlushed_ID) & (~is_stalled)),
        // .wen(1'b1),
        .taken_r(bp_PC_redirect),
        .taken_w(Branch_ctrl),
        .addr_out(bp_addr_out)
    );
```

5 验证

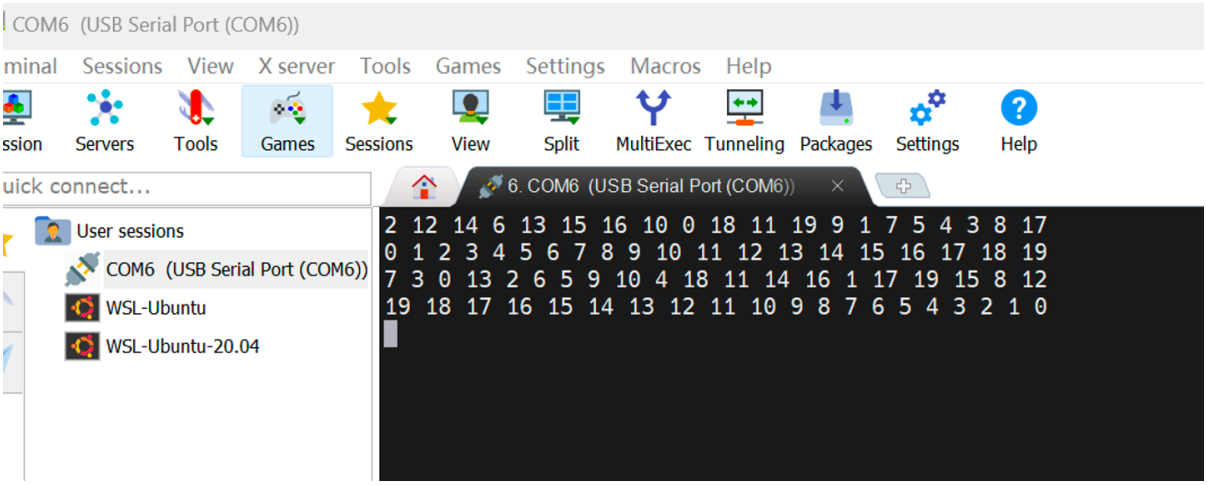
5.1 仿真验证

```
Tilofy@Eun-Desktop MINGW64 /d/2023-f/code/code
$ ./run.sh
core/RV32core.v:312: warning: Port 18 (Addr) of CPUTEST expects 32 bits, got 1.
core/RV32core.v:312:        : Padding 31 high bits of the port.
core/RV32core.v:312: warning: Port 24 (DatatoReg) of CPUTEST expects 2 bits, got 1.
core/RV32core.v:312:        : Padding 1 high bits of the port.
core/RV32core.v:312: warning: Port 25 (PCSource) of CPUTEST expects 2 bits, got 1.
core/RV32core.v:312:        : Padding 1 high bits of the port.
WARNING: core/ROM_D.v:11: $readmemh(core/rom.hex): Not enough words in the file for the requested range [0:255].
WARNING: core/RAM_B.v:20: $readmemh(core/ram.hex): Not enough words in the file for the requested range [0:255].
VCD info: dumpfile wave.vcd opened for output.
VCD warning: $dumpvars: Package ($unit) is not dumpable with VCD.
2 12 14 6 13 15 16 10 0 18 11 19 9 1 7 5 4 3 8 17
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
7 3 0 13 2 6 5 9 10 4 18 11 14 16 1 17 19 15 8 12
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
sim/core_sim.v:23: $finish called at 100002000 (1ps)
```





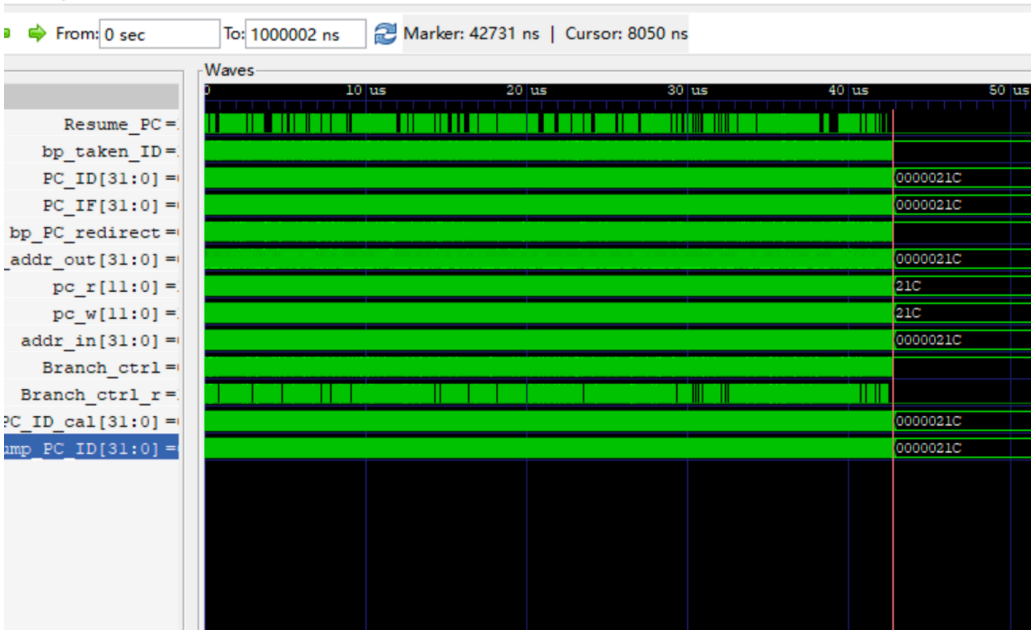
5.2 上板验证



6 思考题

1. 加了分支预测后，仿真跑测试程序，较没加的时候快了多少?

   没有加分支预测的时候大概测试程序需要跑 **45 us**，加上分支预测之后，测试程序需要跑 **42.7 us**，快了 **2.3 us**。



2. 在正确实现BTB和BHT的情况下，有没有可能会出现 BHT 预测分支发生跳转，也就是 branch taken，但是 BTB 中查不到目标跳转地址，为什么？

   有可能。

   一个原因是 BHT 对于该 PC 值的初始预测值就是“跳转”，此时由于之前并没有发生过跳转，所以没有办法在 BTB 中查到目标跳转地址。

   也有可能是发生了异常，导致 BTB 中目标地址并没有正确写入，或者 BHT 的预测出现了错误。

   否则，若 BHT 预测值从“不跳转”变成“跳转”，则至少发生过一次跳转，此时 BTB 中就会写入目标跳转地址。不可能会出现上述情况。

3. 前面介绍的 BHT 和 BTB 都是基于内容检索，即通过将当前PC和表中存储的PC比较来确定分支信息存储于哪一表项。这种设计很像一个全相联的 cache，硬件逻辑实际上会比较复杂，那么能否参考直接映射或组相联的 cache 来简化 BHT/BTB 的存储和检索逻辑？请**简述**你的思路。

   有可能。

   在本次实验中，我就是参考了直接映射的方法来编写 BHT/BTB 的存储和检索逻辑。我使用了 PC 值的低 12 位作为该 PC 的索引值，并将跳转预测值和目标跳转地址写入对应索引值的 BHT 和 BTB 寄存器中。

   该方法避免了编写较为复杂的数据结构来保存 BHT 和 BTB 数据。但是该方法容易发生索引冲突，导致预测失败的概率提升。