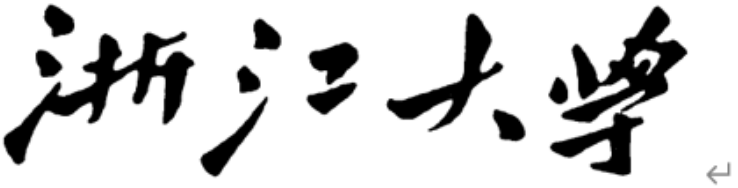
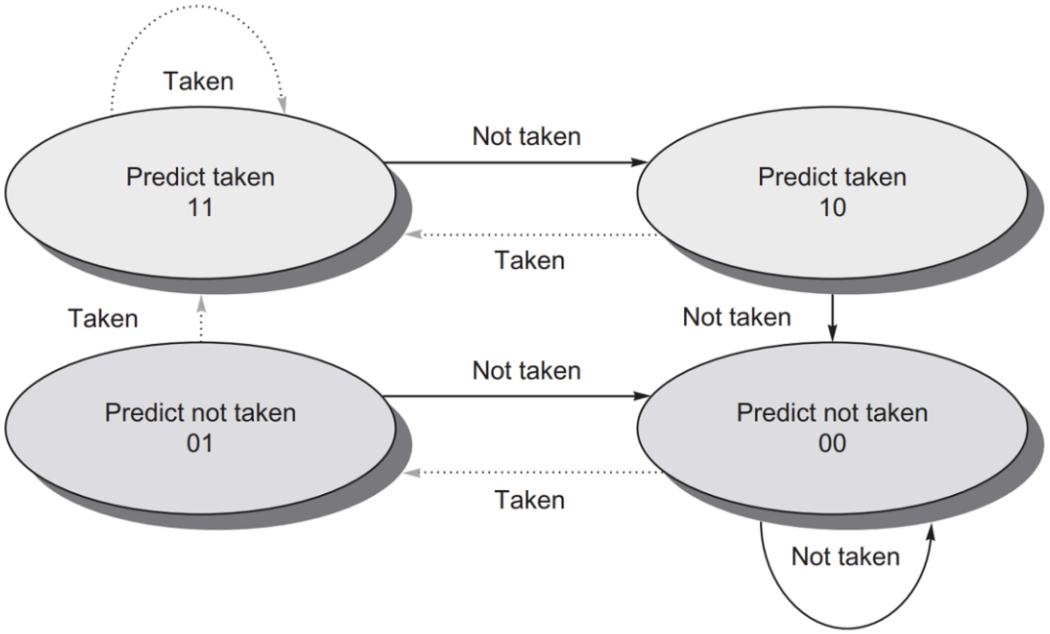


课程名称:	**计算机体系结构**
姓 名:	**杨吉祥**
学 院:	**计算机科学与技术学院**
系:	**竺可桢学院图灵班**
专 业:	**计算机科学与技术**
学 号:	**3230106222**
指导教师:	**常瑞**

一、设计思路

- BranchPrediction.v

  该部分为分支预测模块。`BHT`为预测是否跳转,根据IF地址在`BHT`表中判断是否要跳转,即`predict`信号。`BTB`为预测跳转地址,根据IF地址在`BTB`表中查找跳转地址,即`PC_predict`信号。当指令为分支跳转指令的时候,根据是否进行跳转对`BHT`中的值进行更新,状态图如下图所示。如果指令进行跳转的话,就将跳转地址写入`BTB`中。当分支预测错误的时候就要进行`flush`,`flush`产生的情况有以下几种：预测跳转,实际不跳转;预测不跳转,实际跳转;预测跳转,但预测的地址不是跳转地址;预测不跳转,但预测的地址不是PC+4。因为给的指令地址是从80000000-800003ac,实际地址就是由PC[9:2]决定,我的`BHT`和`BTB`都是由256个entries构成,相当于一个地址对应一个entry,然后一开始`BHT`的值都为0,即不跳转,所以说如果一个地址的指令不属于分支指令或跳转指令的话,那它对应的`BHT`中的值就不会改变,也就是一直预测不跳转,那这样就不会出现分支预测错误,所以只需要判断分支指令是否预测错误即可,据此写出`flush`的判断逻辑。




```
`timescale 1ns / 1ps

module BranchPrediction(
    input clk,
    input rst,
    input [31:0] PC_IF,
    input [31:0] PC_ID,
    input predict_ID,
    input is_branch,
    input isflushed_ID,
    input branch_taken,
    input [31:0] jump_PC,
    output [31:0] PC_predict,
    output predict,
    output isflushed
);

reg [1:0] BHT [0:255];
reg [31:0] BTB [0:255];
```

```
integer i;
wire [7:0] ID_index = PC_ID[9:2];
wire [7:0] IF_index = PC_IF[9:2];

assign PC_predict = BTB[IF_index];
assign predict = BHT[IF_index][1];
assign isflushed = is_branch && ((predict_ID != branch_taken) ||
(branch_taken && PC_IF != jump_PC) || (!branch_taken && PC_IF != PC_ID+4));

always @(posedge clk or posedge rst) begin
    if(rst)begin
        for(i = 0; i < 256; i = i + 1) begin
            BHT[i] <= 2'b0;
            BTB[i] <= 32'b0;
        end
    end
    else if(is_branch)begin
        case(BHT[ID_index])
            2'b00: begin
                if(branch_taken)begin
                    BHT[ID_index] <= 2'b01;
                end
            end
            2'b01: begin
                if(branch_taken)begin
                    BHT[ID_index] <= 2'b11;
                end else begin
                    BHT[ID_index] <= 2'b00;
                end
            end
            2'b10: begin
                if(branch_taken)begin
                    BHT[ID_index] <= 2'b11;
                end else begin
                    BHT[ID_index] <= 2'b00;
                end
            end
            2'b11: begin
                if(!branch_taken)begin
                    BHT[ID_index] <= 2'b10;
                end
            end
        endcase
        if(branch_taken)begin
            BTB[ID_index] <= jump_PC;
        end
    end
end

endmodule
```

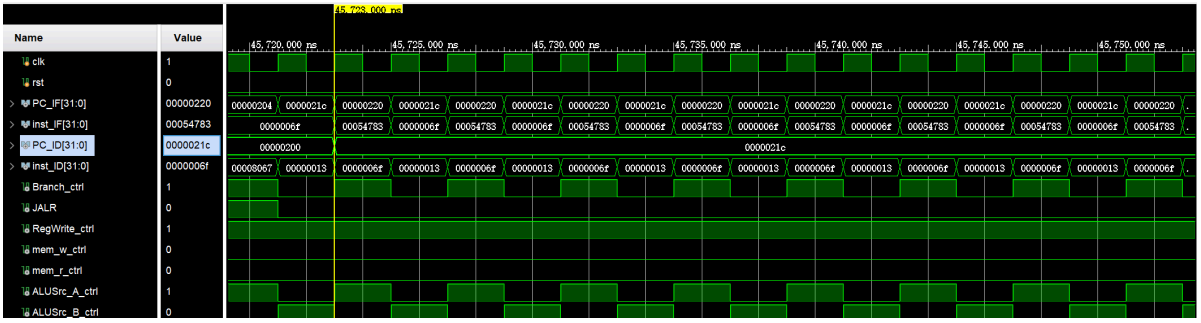
- IF阶段PC的处理

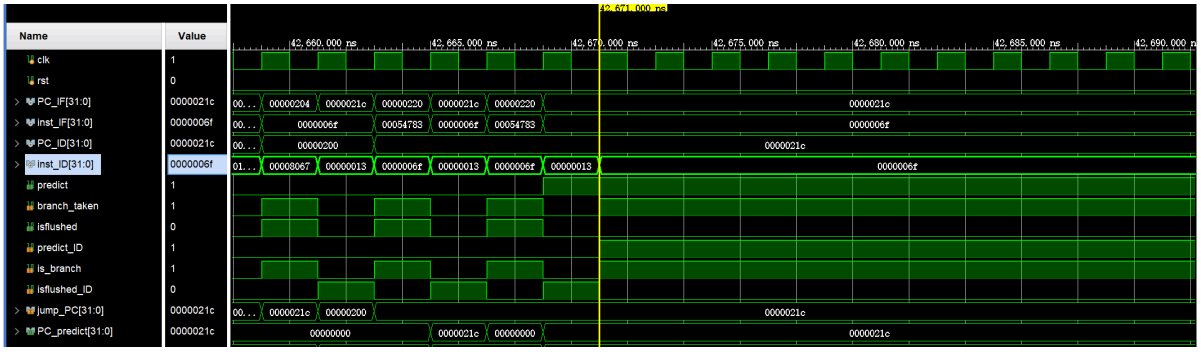
IF阶段PC主要由`flush`,`predict`和`Branch_ctrl`信号决定,首先如果`flush`信号为1,说明分支预测错误,所以此时IF段地址由ID段地址决定,如果是跳转的话,那么IF地址就是跳转地址,如果不跳转,那么IF地址就是ID地址+4。如果`flush`信号为0,说明分支预测正确,那下一个时钟周期的IF地址由此时的IF地址进行预测,根据`predict`和`PC_predict`信号进行预测。同时要将IF_ID寄存器的`flush`信号设为`isflushed`,这样在预测错误的时候能将错误地址的指令`flush`。

```
    BranchPrediction
branch_predictor(.clk(debug_clk),.rst(rst),.isflushed_ID(isflushed_ID),

.PC_IF(PC_IF),.PC_ID(PC_ID),.is_branch(is_branch),.predict_ID(predict_ID),
        .branch_taken(Branch_ctrl),.jump_PC(jump_PC_ID),
        .PC_predict(PC_predict),.predict(predict),.isflushed(isflushed));

    REG32
REG_PC(.clk(debug_clk),.rst(rst),.CE(PC_EN_IF),.D(next_PC_IF),.Q(PC_IF));

    add_32 add_IF(.a(PC_IF),.b(32'd4),.c(PC_4_IF));

    MUX2T1_32 mux_PC1(.I0(PC_4_IF),.I1(PC_predict),.s(predict),.o(PC1));

    MUX2T1_32 mux_PC2(.I0(PC_ID+4),.I1(jump_PC_ID),.s(Branch_ctrl),.o(PC2));

    MUX2T1_32 mux_PC(.I0(PC1),.I1(PC2),.s(isflushed),.o(next_PC_IF));
```

二、思考题

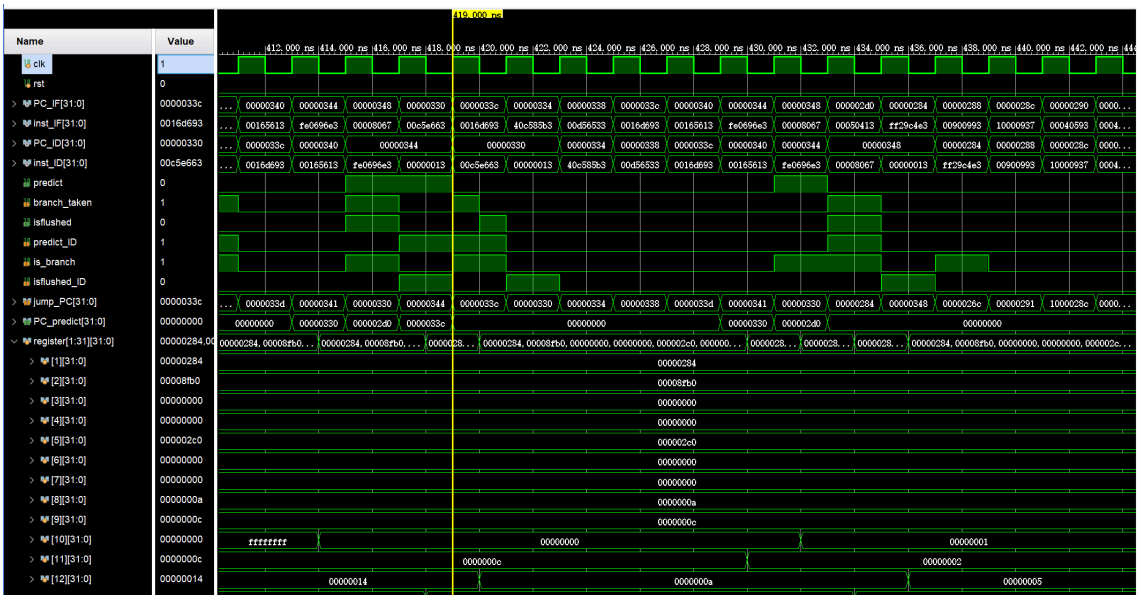
1. 没加分支预测前的仿真时间为45723ns,加了分支预测后的仿真时间为42671ns,快了3us左右,加速比约为1.07。





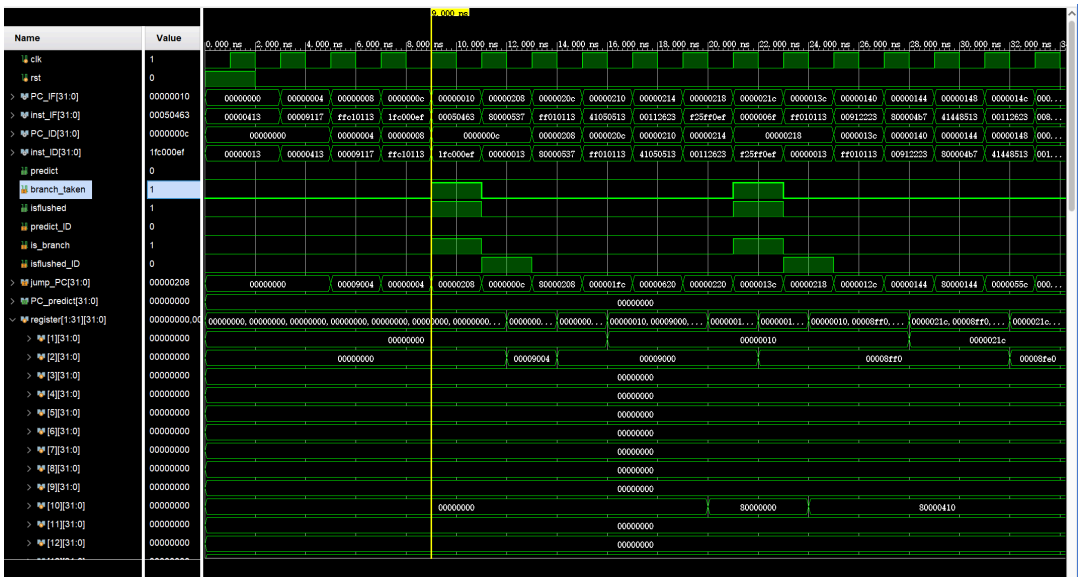
2. ○ 分支预测跳转,实际不跳转

   419ns时ID段地址为0x330,对应指令为`bltu a1,a2,8000033c`,a1,a2分别对应11,12号寄存器,可以看到在417ns时,0x330为IF段地址,此时`predict`为1,预测跳转,由于寄存器的写发生在下降沿,所以在420ns,a1为0xc,a2为0xa,a1大于a2,所以实际情况为不跳转,`branch_taken`为0,`isflushed`为1



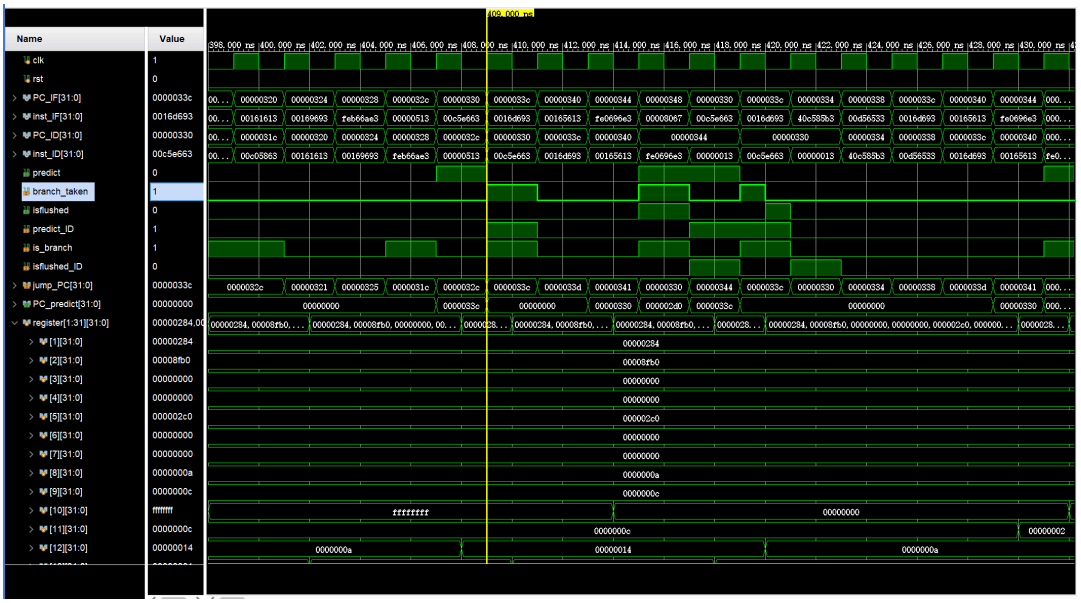
- 分支预测不跳转,实际跳转

  9ns时ID段地址为0xc,对应指令为`jal ra,80000208`,可以看到在7ns时,IF段地址为0xc,此时`predict`为0,预测不跳转,但该指令实际要跳转,所以`branch_taken`为1,`isflushed`为1



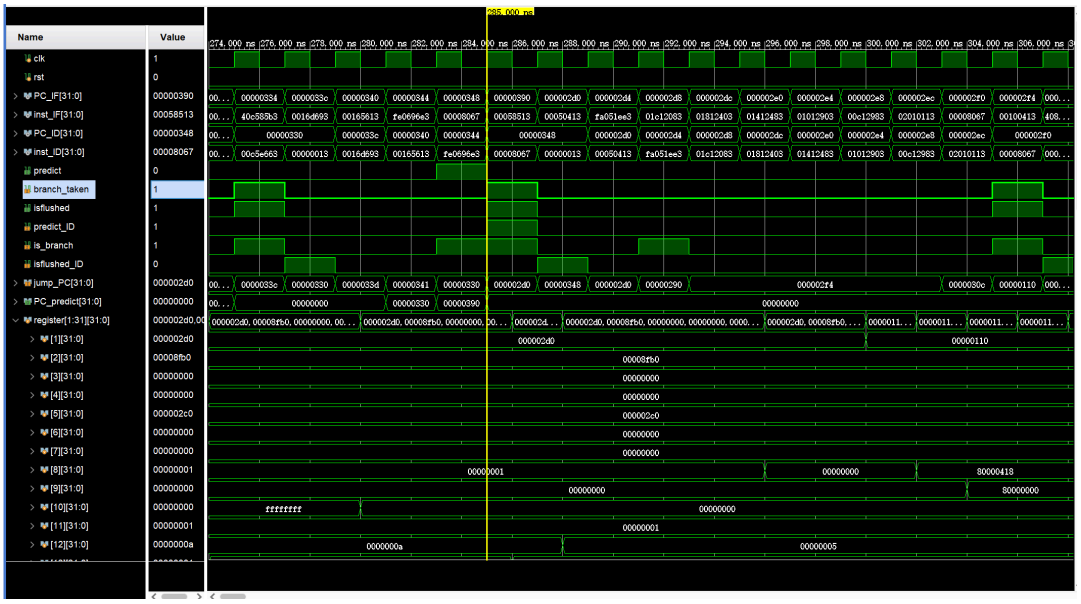
- 分支预测跳转,实际跳转,BTB 给出的跳转地址正确

  409ns时,ID段地址为0x330,对应指令为`bltu a1,a2,8000033c`,a1,a2分别对应11,12号寄存器,可以看到在407ns时,0x330为IF段地址,此时`predict`为1,预测跳转,预测跳转地址`PC_predict`为0x33c,在410ns时,a1为0xc,a2为0x14,a1小于a2,所以实际情况为跳转,跳转地址`jump_PC`为0x33c,与之前预测的跳转地址相同,所以`branch_taken`为1,`isflushed`为0



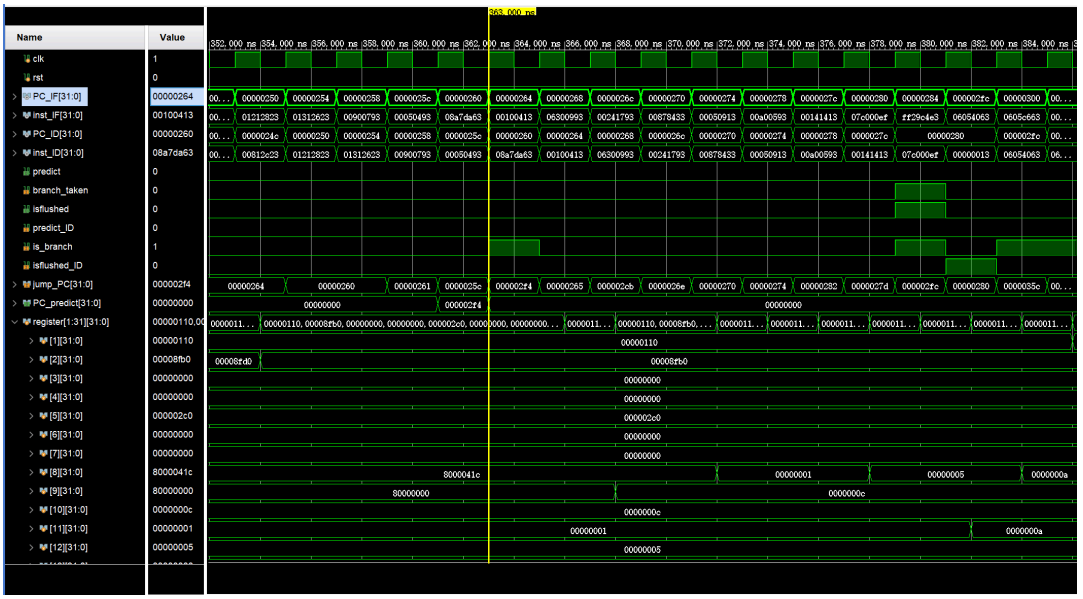
- 分支预测跳转,实际跳转,BTB 给出的跳转地址错误

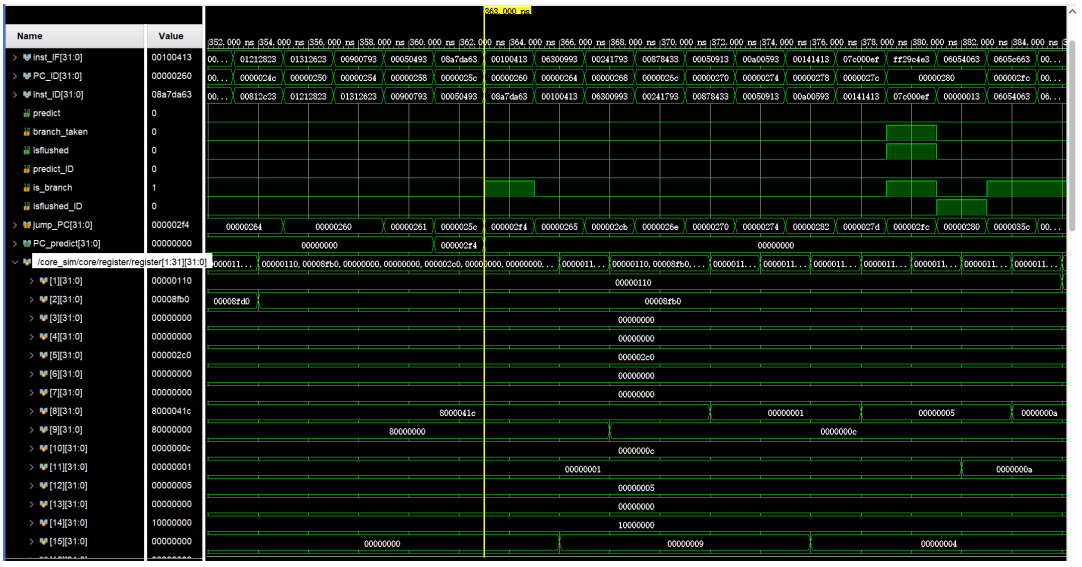
  285ns时,ID段地址为0x348,对应指令为`jalr zero,0(ra)`,ra为1号寄存器,可以看到在283ns时,IF段地址为0x348,此时`predict`为1,预测跳转,预测跳转地址`PC_predict`为0x390,在286ns时,ra为0x2d0,该指令为跳转指令,跳转地址`jump_PC`为0x2d0,与之前预测的跳转地址不同,所以`branch_taken`为1,`isflushed`为1



- 分支预测不跳转,实际不跳转

  363ns时,ID段地址为0x260,对应指令为`bge a5,a0,800002f4`,a0和a5分别对应10号和15号寄存器,,可以看到在361ns时,IF段地址为0x260,此时`predict`为0,预测不跳转,在364ns时,a5为0,a0为0xc,a5小于a0,所以实际情况为不跳转,预测正确,`branch_taken`为0,`isflushed`为0





3. 相比于静态分支预测,动态分支预测的优点是具有更高的预测准确率,适应程序的动态行为,支持复杂的预测模式,缺点是硬件开销大,延迟更高

三、感想

本次实验只需要完成动态分支预测模块即可,工作量比较小,感觉主要的难点在于`flush`的逻辑处理,一开始我是根据ID段地址去查找BHT和BTB表,将BHT表中的是否跳转与ID段得到的是否跳转,BTB表中的跳转地址与ID段得到的跳转地址进行比较,但是这样得不到正确结果,我觉得可能是ID段地址处于IF段时,BHT和BTB表刚好修改了该地址对应的值,导致比较的结果不对,所以我就将IF段的预测信号和预测地址都传入ID段,这样之后再进行比较就得到正确结果了。虽然不是很理解,但终于搞出来了。希望继续努力,顺利完成之后的实验。