# ECEN 651 Lab Report 6

## Zhenlei Song

## Forwarding Unit

module ForwardingUnit(UseShamt , UseImmed , ID\_Rs , ID\_Rt , EX\_Rw , MEM\_Rw,

EX\_RegWrite , MEM\_RegWrite , ALUOpCtrlA , ALUOpCtrlB , DataMemForwardCtrl\_EX,

DataMemForwardCtrl\_MEM);

// Define inputs and outputs

input UseShamt , UseImmed ;

input [4:0] ID\_Rs , ID\_Rt , EX\_Rw , MEM\_Rw;

input EX\_RegWrite , MEM\_RegWrite ;

output reg [1:0] ALUOpCtrlA ;

output reg [1:0] ALUOpCtrlB ;

output reg DataMemForwardCtrl\_EX , DataMemForwardCtrl\_MEM ;

always @(\*) begin

// Control Signal definitions:

if(UseShamt == 1)

ALUOpCtrlA <= 2'b00;

else if (ID\_Rs == EX\_Rw & EX\_RegWrite == 1 & EX\_Rw != 0)

ALUOpCtrlA <= 2'b10;

else if(ID\_Rs == MEM\_Rw & MEM\_RegWrite == 1 & MEM\_Rw != 0)

ALUOpCtrlA <= 2'b01;

else

ALUOpCtrlA <= 2'b11;

// If I-type

if(UseImmed == 1)

ALUOpCtrlB <= 2'b00;

else if (ID\_Rt == EX\_Rw & EX\_RegWrite == 1 & EX\_Rw != 0)

ALUOpCtrlB <= 2'b10;

else if(ID\_Rt == MEM\_Rw & MEM\_RegWrite == 1 & MEM\_Rw != 0)

ALUOpCtrlB <= 2'b01;

else

ALUOpCtrlB <= 2'b11;

// Check data dependancy, case 1

if (EX\_RegWrite == 1 & ID\_Rt == EX\_Rw & EX\_Rw != 0) begin

DataMemForwardCtrl\_EX <= 1'b0;

DataMemForwardCtrl\_MEM <= 1'b1;

end

// Check data dependancy, case 2

else if(MEM\_RegWrite == 1 & ID\_Rt == MEM\_Rw & MEM\_Rw != 0) begin

DataMemForwardCtrl\_EX <= 1'b1;

DataMemForwardCtrl\_MEM <= 1'b0;

end

else begin

// No data dependancy

DataMemForwardCtrl\_EX <= 1'b0;

DataMemForwardCtrl\_MEM <= 1'b0;

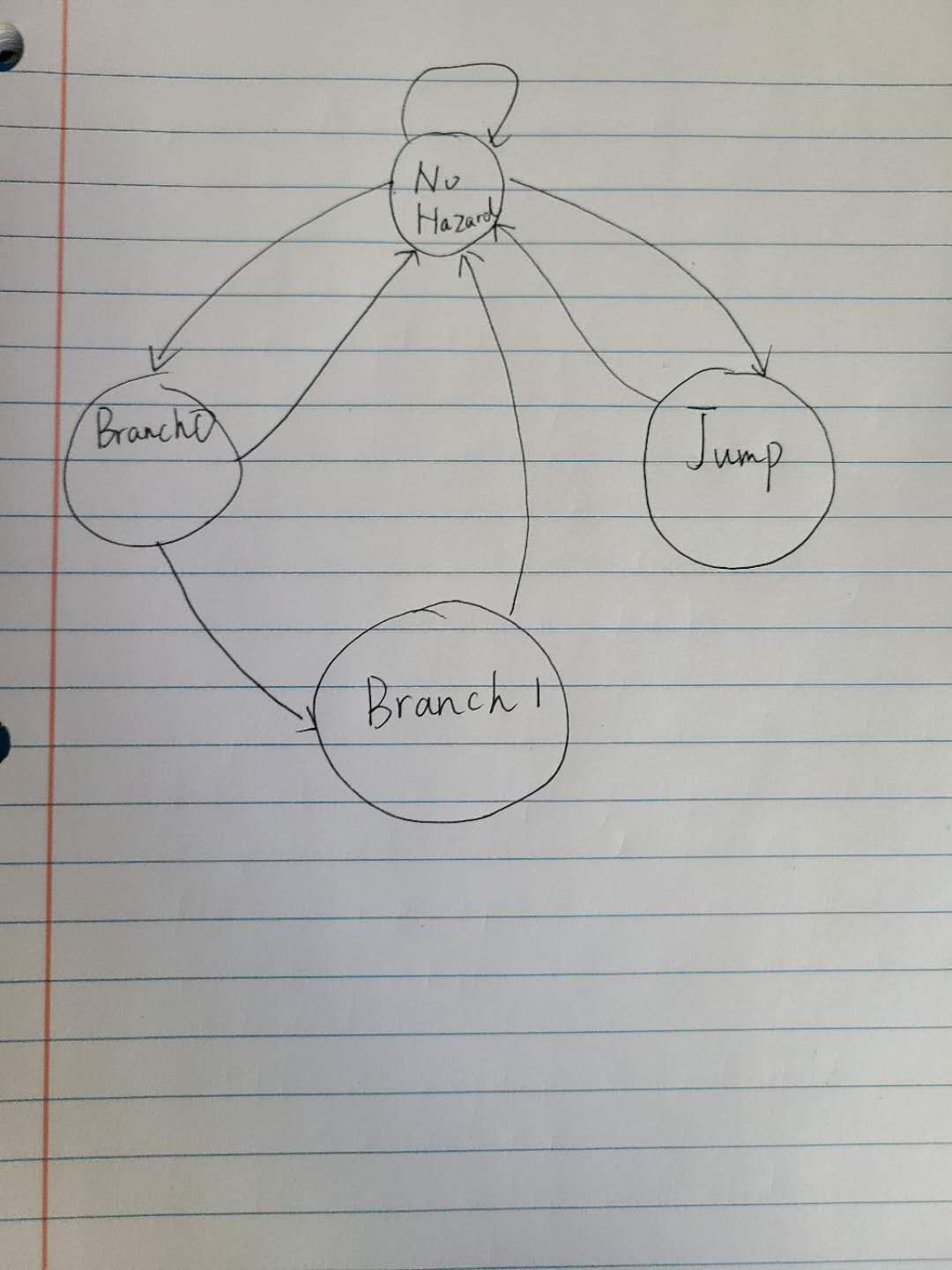
end

end

endmodule

The forwarding unit is used to control whether data forward to ALU from EX stage or MEM stage. If Rs and Rt match with previous RW addr, and immed/shamt are not set. The data is brought back.

## Hazard Unit



This unit is used to detect hazards and output 3 signals (PCWrite, IFWrite, Bubble). Within the unit, a 4-state machine is implemented.

No Hazard state: normal instructions

Branch0 state: if branch is set, enter this state. If not taken, return to No Hazard state. If branch taken, enter Branch1 state.

Branch1 state: If branch taken, then an extra stall have PCWrite and IFWrite enabled to fetch the new PC.

Jump state: When jump is set, enter this state. A stall is induced.

## Pipelined Processor

In this unit, signals connecting to Hazard unit and Forwarding unit have to mark which stage they are at. As given the Hazard units sits in the ID stage, every EX stage register or signal read is has appended a ‘EX’ and everything from the mem stage has appended a ‘MEM’. The same thing was done with the forwarding unit, that according to its signals, it selected between the outputs of the mem stage or the reg stage.

## Test Result

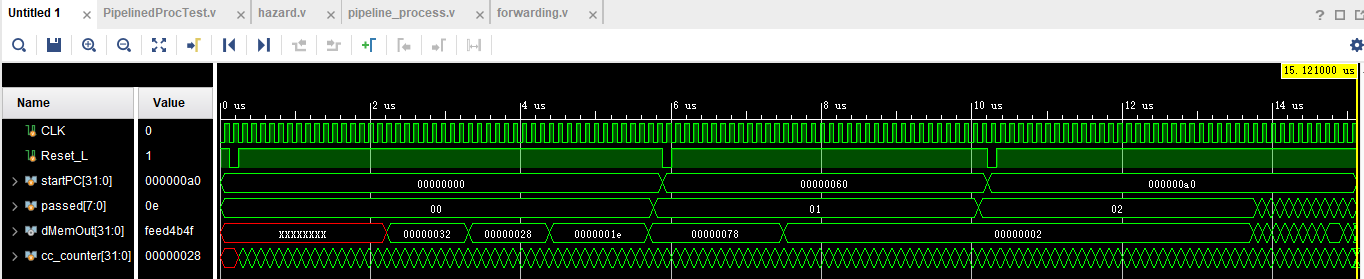


Fig 4.1 Pipelined processor waveform

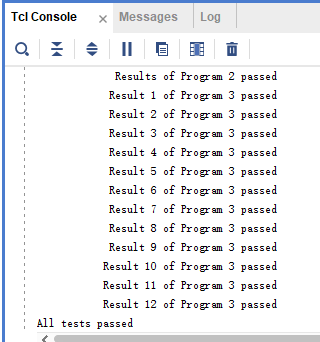


Fig4.2 Log

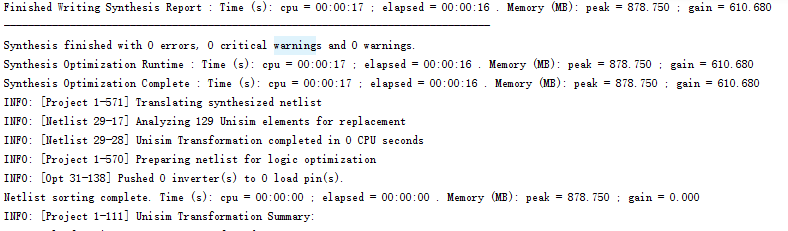
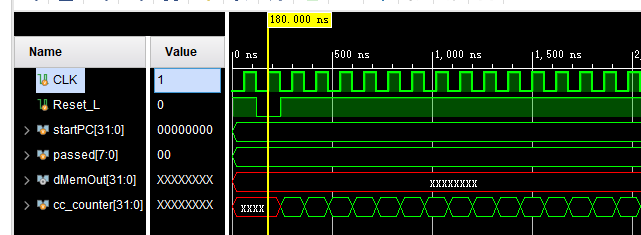


Fig 4.3 Synthesis Report

## Questions

1. This method is called static branch prediction. Instead of enabling a stall in branch0 and output PCWrite and IFWrite, it’s supposed to handle them inside if the branch is not taken. If taken, a flush signal is issued to reset all signals, and fetch new PC.



According to the figure above, the estimated clock is 120ns. Because of stalls, memory, jump and branch are critical to clock rate.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Instructions | Single Cycle | Pipeline running time(ns) | Pipeline running cycles | CPI |
| Program1 | 33 | 33 | 5761 | 48 | 1.45 |
| Program2 | 11 | 11 | 4440 | 37 | 3.36 |
| Program3 | 37 | 37 | 5160 | 43 | 1.16 |
| Total | 81 | 81 | 15121 | 128 | 1.58 |

1. For the same clock rate 120ns. Means 8.33M clock cycles per second. For single cycle, 8.33/1=8.33MIPS. For pipelined processor, 8.33/1.58=5.27MIPS. Under this case, single cycle processor is faster. But actually pipelined processor clock rate is supposed to be higher than single cycle ones. Finally pipelined should be faster.