King Abdullah University of Science and Technology

CS282: Computer Architecture and Organization

Spring 2013

Project Part 3

Due Monday, 6th May, 2013 at 11:59PM

Description

This is the third and final part of the 5-stage pipeline design. Here, you are expected to build essential components for your designed MIPS processor to ensure valid operations. The problem behind the wrong results the programmer from part 2 was getting was because your design was not handling the data hazards. You will fix such problem in this part of the project by building a forwarding unit and add it to your design. However, since the forwarding unit cannot solve all the data hazards like the case of "load" instruction, you will also build a stalling unit to stall the CPU in such case.

In this part, you were given the implementation of the CPU file where all the stages are completed except the EX stage. The file is named now ***CPUv2.v***, and new wires definitions were added.

You will also provide a short description for a branch prediction in an attempt to eliminate some control hazards.

1. Forwarding Unit (45 points)

Forwarding is a useful way to eliminate some data hazards and greatly enhance the execution time of the five stage pipeline without incurring a severe hardware or timing penalty.

This will be accomplished in two parts:

1. **Part 1: Implementing The Module**
   You are expected to correctly interpret how forwarding works in this MIPS processor and design your module in a single 'always' block.
   The input and output ports have already been provided in the, for your convenience.
   You are required to design ***Forward.v***.

2. **Part 2: Integration**
   In this part, you will integrate the module in the new MIPS CPU file ***CPUv2.v***. You are required to modify the old design to accommodate the forwarding unit you have

just designed by following the new design illustrated in '***CS 282 – Project Part 3 – DatapahV2.pdf***'. The forwarding unit also has been already instantiated in the CPUv2 file, for your convenience.

You only need to redesign the EX stage in "***CPUv2.v***". All other the stages are implanted as specified in part 1. Thus, you don't need to modify these stages. Also, notice the new added signals and wires declaration at the top of the CPUv2 file to be used in the EX stage.

2. Load Stalling Unit (45 points):

This entails the introduction of another module that determines whether you need to stall after a 'load' instruction.

This will be accomplished in two parts:

A. Part 1: Implementing The Module

You are expected to correctly interpret how the detection and stalling for the load instruction works in this MIPS processor and design your module in a single 'always' block. The input and output ports have already been provided in the template, for your convenience. You are required to design ***Load_Stall.v***.

Hints:

1- Signals (*RDaddr_EX*, *RSaddr_ID*, and *RTaddr_ID*) are used to detect the data dependence.
2- Signals (*WE_EX*, *DMC_EX*, and *WBmux_EX*) are used to detect that there is a load instruction.
3- In order to stall, the above two conditions must be true.

B. Part 2: Integration

The module has been already instantiated in the new MIPS CPU file ***CPUv2.v***. You are only required to modify the old design of the control unit to communicate with the load stalling unit you have just designed in order to stall the CPU at the right time.

When a stall is required (i.e Stall output signal from ***Load_Stall.v*** is 1), two things should be performed to stall the CPU:

(1) The PC register and the IFID pipeline registers should freeze. This is done by disabling the capturing of new inputs to these registers (EN signals are set to zero).
(2) A bubble is inserted in the pipeline. This is done inside the control unit by setting all the control signals to zeros.

A new modified module for the Control Unit ***"Control_UnitV2.v"*** has been given to you that only adds the signal "Stall" as an input.

To successfully complete this part, you are only required to add the solution you did for the control unit in part 2 to this file and integrate it with the Stall signal. So, whenever the Stall signal is 0, normal control signals are generated as in part 2. If Stall signal is 1, then set all control signals to zeros (i.e A bubble is inserted).

3. Branch Prediction(10 points):

Now, you are going to "virtually" design a branch predictor for your processor and have it integrated with your existing system components. "Virtually" means that you are going to design and integrate the branch predictor in your head while describing it verbally on papers. Branch prediction is introduced to our system to help mitigate the control hazards exist due to pipelining. The branch predictor we are going to use here is the 2-bit saturating counter explained in the class. The following points should guide you through your discussion. You can use whatever illustration tools you need to in-detail explain your design:

- Describe in basic logic terms the design algorithm of the branch predictor. You can use pseudo code here for illustration purposes.
- Describe what happens to the branch decision and branch address calculation and at what stage of the pipeline they should be place?
- Finally, after integration of the branch predictor with your full system, take us through a "virtual tour" describing what happens when executing a branch instruction i.e. beq. In your tour, discuss what goes on at each stage of the pipeline trying to cover all possible cases i.e. branch is taken or not taken.

Design\Compile Tools

You are required to use the free ***Xilinx WebPACK 12.1*** to compile and synthesize your work. Refer to part 1 of the project for more details of how to install and use the software to compile your modules.

In order to compile and synthesize the CPUv2.v file, you will need the simple components from part 1. For PC, use the ***"PCv2.v"*** file given to you with this part. For other modules, use the ones you designed in part 1.

Submission Guidelines:

1- You are required to fill the necessary lines of code and put all files into a single ZIP folder:
CS 282 – Project Part 3 – Member1LastName Member2LastName.zip

2- This assignment should be submitted through the Blackboard module no later than Monday, 6 May, 2013 at 11:59PM. Only one submission from any group member is required.

3- 24-hour lifelines may only be used if BOTH team members have not already used their lifelines. Late submissions will not be accepted.

4- If you have any questions, please post them to the discussion forum on Blackboard so that everyone can benefit. Please do not post any code in any form.

5- Non-adherence to submission guidelines will lead to loss of 10% of the total score on this part of the project.

Early submission (5 points): If your group submits the assignment at least 24 hours before the deadline *and the solution is fully correct*, your group will receive a bonus of 5 points. Note that extra credit can allow your score to go above 100%.