EEL4713 Assignment #5
Spring 2012
Assigned: 3/15/12
Checkpoint 1 Demo: 3/29/12 in Lab
Checkpoint 2 Demo: 4/5/12 in Lab
Demo: 4/12/12 in Lab
Due: 4/14/11 @ 11:55pm Via Sakai

Section 1: Setup

There is no setup in this assignment.

Section 2: Textbook Questions

Questions from chapter 4: 4.12.1-3, 4.20.1-2, 4.24.1-3

Section 3: Laboratory

In this laboratory you will be modifying your single cycle processor from Assignment 4 in order to implement a pipelined processor. Your processor must support the same 29 instructions required for the single cycle processor, detect potential hazards, and resolve them by forwarding when possible or stalling when not. Please read chapter 4 of your textbook for insight on how this can be done.

Begin your design by partitioning your single cycle processor into 5 stages: instruction fetch (IF), instruction decode (ID), execution (EX), memory access (MEM), and write back (WB). Your goal should be to equally divide the propagation delay of your single cycle implementation into the 5 stages. Consult Figure 4.33 in your textbook for inspiration on how you should partition your design. Create four registers to place in between your pipeline stages that will transfer needed data from stage to stage. It is possible to eliminate the "mclk" signal by using the registered input of the altsyncram component as a register between the pipeline stages. Eliminating the "mclk" signal should greatly increase your maximum clock rate and is highly recommended.

At this point your processor should be able to execute dependency free code without any problems. Test your processor before moving on. Next, create a table that describes dependencies that may occur when a certain instruction is followed by another instruction and what action you should take in order resolve the dependency (i.e. forwarding or stalling). Below are a few entries from such a table that will help you understand what your table should contain:

	R	I	L	S	JR	LUI	B	JAL
R	1a: exmem.rd=idex.rs(1) =idex.rt(2) 2a: memwb.rd=idex.rs(3) =idex.rt(4) 3a: memwb.rd=ifid.rs(5) =ifid.rt(6)				1a: idex.rd=ifid.rs(stall) 2a: exmem.rd=ifid.rs(8) 3a: memwb.rd=ifid.rs(5)			
I	1a: exmem.rt=idex.rs(1) =idex.rt(2) 2a: memwb.rt=idex.rs(3) =idex.rt(4) 3a: memwb.rt=ifid.rs(5) =ifid.rt(6)							
L		1a: ifid.rt=pre.rs(stall) 2a: memwb.rt=idex.rs(3) 3a: memwb.rt=ifid.rs(5)						
S	do nothing							
JR			do nothing					
LUI	1a: exmem.rt=idex.rs(11) =idex.rt(12) 2a: memwb.rt=idex.rs(13) =idex.rt(14) 3a: memwb.rt=ifid.rs(15) =ifid.rt(16)							
B					do nothing			
JAL		1a: exmem.31=idex.rs(21) 2a: memwb.31=idex.rs(23) 3a: memwb.31=ifid.rs(25)						

(1). forward exmem.aludata to rs of alu
(2). forward exmem.aludata to rt of alu
(3). forward WBdata to rs of alu
(4). forward WBdata to rt of alu
(5). forward WBdata to rs of idex reg
(6). forward WBdata to rt of idex reg
(8). forward exmem.aludata to rs of idex reg
(11). forward ui from exmem to rs of alu
(12). forward ui from exmem to rt of alu
(13). forward ui from memwb to rs of alu
(14). forward ui from memwb to rt of alu
(15). forward ui from memwb to rs of idex reg
(16). forward ui from memwb to rt of idex reg
(21). forward PC+4 from exmem to rs of alu
(23). forward PC+4 from memwb to rs of alu
(25). forward PC+4 from memwb to rs of idex reg

The first entry says that when an Rtype instruction is followed by another Rtype (both of which are not jump register instructions), the instructions are 1 instruction apart, and rd of the instruction from the EXMEM register is equal to either the rs or rt of the instruction from the IDEX register you should perform actions (1) or (2) to correct the hazard which is forward the alu data from the EXMEM register to the rs or rt inputs of the alu. The first entry also contains the information of what to do when the Rtype instructions are 2 or 3 instructions apart. Depending on how your processor is implemented, your table may have a few key differences.

Checkpoint 1: show your completed table and demonstrate the functionality of your pipelined processor without hazard detection and data forwarding no later than the first DEMO date.

Create a hazard detection unit that detects the hazardous instruction combinations that cannot be resolved by forwarding. This unit in combination with other hardware should cause the pipeline to stall in these situations.

Checkpoint 2: demonstrate the ability of your pipeline to detect and resolve these hazards by stalling no later than the second DEMO date.

Create a data forwarding unit that resolves all other dependencies by forwarding. Make sure to take into account what happens when multiple dependencies are detected at the same time. This unit in combination with other hardware should be the last step in completing your pipeline implementation.

Separate from the pipeline, create a clock cycle counter that will count the number of clock cycles it takes to complete a program. The inputs should be reset and clk and the output should be the current clk count. On reset the count should go to 0.

Checkpoint 3: demonstrate the full functionality of your pipelined processor with hazard detection and data forwarding no later than the third DEMO date. You must successfully execute both the lab4test and lab4demo programs as well as a third program provided on demo day. If you successfully complete checkpoints 1-3 no later than the first DEMO date you will receive 30 points of extra credit in addition to the points dedicated for the checkpoints and successful operation of the demo programs. If you successfully complete checkpoints 1-3 no later than the second DEMO date but later than the first you will receive 15 points of extra credit. No extra credit will be given for completion on the third DEMO date.

For your report, include a detailed section where you go through the design process (this should include the table you created). Include a section where you decode both the lab4demo and lab5demo programs into MIPS assembly code along side the functional simulation output (in table form) of your processor. Make

sure you annotate your output to show that your processor executed the program correctly. Since the demo programs do not fully test the operation of your pipelined processor, provide additional simulation output from programs of your own creation that demonstrate the correct operation of your processor for hazards not included in the demo programs. Provide a detailed diagram of all interconnects in you system (similar to Figure 4.65 in your textbook). Make sure to include VHDL code for newly created components. Your report should still follow the report guidelines discussed in assignment 2.

Timing analysis of your processor should be done with the CycloneII EP2C8T144C8 FPGA (which is the FPGA in you EEL4712 board if you want to test your design on actual hardware) as your target device. Part of your DEMO grade will be based on the maximum clock rate listed in the timing analyzer report. The points will be assigned on a linear scale for frequencies ranging between 25MHz and 50MHz. Frequencies above 50MHz will receive extra credit. Anything below 25MHz will receive a zero. In your report make sure to discuss what design choices led to your clock rate and what can be done to improve the speed of your processor.

Note: in addition to submitting your reports and design files electronically via Sakai, you must demonstrate your components functionality in Lab on the DEMO dates listed on the first page of this Assignment. Failure to submit your design files will cause you to receive a zero on the design sections of the Assignment. Please make sure that you only send your VHDL code, BDF files, mif files,
and/or symbol files and don't send any other files generated by Quartus.