CSCE 230 Project Part IV

Adding I/O and ARM-like Conditional Execution — Due April 23rd/24th

In general, you are allowed to use block diagrams, VHDL, or a mix of both to complete any of these assignments. Each will come with its own advantages/disadvantages. Block diagrams are easier to realize but more simple mistakes, i.e. wire connections. VHDL may be harder to implement but less simple mistakes. If you copy and code from other sources, internet/books, you must cite your source or receive a zero for the project. Each team member must work together and should only turn in one assignment/report/check off by the TA.

Overview

Your objective for this fourth part of the project is to add to the already completed processor, extensions to the lw and sw instructions to handle the input/output with the help of additional components and information provided to you. You will also extend the processor deign to handle an ARM - like conditional execution of instructions , which is described in pages 11 - 12 of the Project Overview document, as well as in section D.9 of the textbook

1. Block-level datapath design (or VHDL design)
2. Design of inidividual Components
3. Datapath integration
4. Control Unit Design
5. System integration and validation

Steps 1-3

Now that you already have a basic processor with load and store capability, only a little bit of logic is needed to connect the I/O memory. You will be given IO_MemoryInterface.bdf as one of the I/O files to extend lw and sw to deal with I/O. In addition to the mem_write control signal, the upper four bits of the memory addresses are used to determine which I/O device is being addressed. The upper four bits are assigned to the I/O devices as follows:

1000 – Slider Switches (SW)

0100 – Push Buttons (KEY)

0010 – 7-segment display (HEX)

0001 – Green LEDS

Since the mem_write control signal will be 1 for sw instructions regardless of whether you want to write to memory or perform an output operation, additional logic is needed for the mem_write input to the data memory . When any of the upper four bits of the memory address are a 1, then mem_write should be a 0 for the data memory. Think of the logic to determine if either of those bits are 1 and the mem_write is 1.

Similarly, y_select will be 01 for lw instructions, so the upper four bits of the address should be used to select which memory output (data memory or I/O memory) will be an input to mux_Y . Think of what object you can add here to choose between the output of the memory unit and the output of the IO

In addition to clock and reset, you need to add inputs and outputs for each of the four I/O devices and assign them to pins on the board. To do this, right - click on a pin and select Locate→Locate in Pin Planner. Refer to the tables in Pin_Connections_FPGA_Board.pdf. Note, that if you map your reset input to a push button, then push button has a value of 1 when not pressed and 0 when pressed . Also, the Hex display is opposite as well. A 1 turns a bit off, while a zero will turn it on.

For conditional execution you should already have a way of storing the conditional flags with your PS register. The only thing that should change is when you want to update the PS register. You can use the flag_enable to determine when to update your PS. This flag should be set based on the input of the S bit.

Step 4

The control unit already has inputs for cond and the four flags. The easiest way to implement conditional execution is to add another internal signal (lik e the wmfc signal) to represent whether or not the instruction should be executed. You should check the condition and the flags during the down cycle of the clock, not the upcycle. Add an if statement before the if(rising_edge(clock)) to check for falling _edge(clock). In that if statement, you want to check the flags to match the condition passed in, and change the enable signal you created.

You may have been wondering about the S bit. I already mentioned it above, but this is what you will use to determine if you want to update your PS. You will have to add code in your control unit that sets the flag_enable bit if the S bit is set. this should be done for R and D type instructions, except JR.

Step 5

A **MAJOR** component of each part in the project is the last step. **Make sure you reserve sufficient time for this step** as your validation results will help us understand the success of your implementation.

You will also test your implementation thoroughly for correctness and timing performance. Also create and program that uses the I/O and download your processor to your board and see that it works as expected. To program the board, make sure all your pins are assigned, then go to Tools→Programmer, then make sure that your board is listed under hardware and click Start.

Assignment

As described in the above sections, modify your datapath to include the I/O memory interface and the additional logic for it to correctly function, and add to the control unit to execute instructions conditionally .

Tasks you must perform

- Download the provided component(s) from Piazza
- Look over and develop an understanding any provided component(s)
- Add the I/O memory interface to your datapath.
- Add to the control unit to implement conditional execution.
- Hand-assemble instruction to test I/O and instructions that have conditions to execute.
- Create a test script(s) (.do file(s)) and run a simulation(s) of the processor to make sure the I/O works and that instructions can be executed conditionally.

Testing and simulation

Before verifying your enhanced processor with the two extensions made in this part by a test program, consider doing basic component - level testing of the modified parts of the design. Then do a functional simulation of the test programs for comprehensively testing all the instructions implemented thus far. As before, add all inputs and outputs and look for errors as your test program runs. I

IMPORTANT TIPS:

- Test each I/O device individually
- Get one condition to work before adding the rest
- In the .mif file, leave the first address space empty like before.
- Get I/O to work in simulation before trying it on the board.
- When assigning pins to inputs, be sure to assign the clock to one of the clocks on the board, and the reset to key0.

In your timing simulation, strive to test your processor for correct operation at the highest possible clock speed. As you processor grows more complex, it is very likely that you will have to slow down this clock speed of the system to produce the correct results by giving signals enough time to propagate. The test cases for the timing simulation should be the same as in the functional simulation.

Report

You must create a three page (minimum, not including images) technical report detailing the processor you have created so far. Include its current abilities (which instructions is can perform), a speed overview (how fast it can run), and the components inside of it. You should also briefly talk about your groups experiences with connecting everything. Use normal margins, normal text size, and normal formatting.

This report should be structured. This means you should include a title page as default. You should also have headers and sections as necessary. If you include images, make sure to reference them in your document. An included image, that isn't talked about, serves no purpose in a report. This report will be graded much more rigorously. If you have questions regarding how this should be structured, look over your comments from the first report or second report. If questions remain, then talk to Zach or Yaoxin.

A few examples of how points will be taken off follows. These are just a few of the possible reasons you could lose points.:

- No title page
- No headers/sections where needed
- Erratic flow. Make sure when you change topics when writing to making it logical and easy to follow. Avoid jumping from one topic to another without a smooth transition
- No images included to further describe your processor
- Images included that are not talked about or have no relevancy

Grading and TA check-off

The grading breaks down from the table below:

Points	Part
20	Design File (.qar)
10	Simulation
15	Check off and demonstration
30	Technical report

The design file, simulation, and report are due by Midnight of April 2nd/3rd depending on your lab. The Check off is due by the time Lab ends. **NOTE:** If you are in the 4:30-6:20 lab on Thursday and do not receive a check off by 6:30 you will be considered late as to make it fair for the other two lab sections.

Submit

You must submit the files electronically to webhandin by 11:59PM on April 23rd/24th. The naming convention should be as follows:

- "Project-team-number"_Part4.qar for the project archive file. The simulation file should be located inside of the archive.
- "Project-team-number"_Part4.(.doc/.tex) for the project report. This report should follow the best practices of writing technical reports. For tips looks at the links provided on Piazza.

Hints

A few hints to help you along.

1. Your I/O should be able to work on any of the clock speeds given by the board. However, this may not be true depending on your setup. Make sure to test your processor on both if you can't get them to work.
2. Make sure to leave the first instruction in the .mif file empty. Placing an instruction there tends to cause problems.
3. Reset is active high, so if you connect key0 to your reset input, make sure to invert it so that your processor actually runs.
4. You can assign pins to a bus input. You do not need an individual node input for each pin.