CSCE 230 Project Part V

Extending I/O and adding J-type instructions

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 allow more simple mistakes, e.g. wire connections. VHDL may be harder to implement but you will make less simple mistakes. If you copy any 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 fifth part of the project is to add to the already completed processor, extensions to I/O memory interface to handle the remaining I/O devices (hex1,hex2,hex3, and red leds). You will also extend the processor design to handle J-type instructions (j, jal, and li), which are discussed in the Project Overview document. The bits 19..16 or 3..0 (Your choice) should handle the destination register for li. The design process will follow the steps below:

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

Steps 1-3

With an already basic I/O memory interface, you will need to expand it to handle the rest of the I/O devices. The following table gives the assignment of the upper four bits for each I/O device:

1000 – Slider Switches (SW)

0100 – Push Buttons (KEY)

0010 – 7-segment display (HEX0)

0001 – Green LEDS

0011 – Red LEDS

0101 – 7-segment display (HEX1)

0110 – 7-segment display (HEX2)

0111 – 7-segment display (HEX3)

The only component that needs modified in order to make these additional I/O devices work is the IO memory interface. Your goal is to device a way to handle these extra I/O devices, without having to change anything outside that component. As a hint, think about the other time you tried to address something, and what component helped.

In addition to the I/O devices, you will need to implement J-type instructions. For J and Jal, the main component you will want to modify is the IAG. For load immediate, you will have to decide where to interject the 16 bit immediate value into your datapath. These three instructions require a little modification, but are relatively simple to implement.

Step 4

In the control unit, you will simply have to implement the ability to check for J-type instructions and to set the corresponding control flags. One very important thing to note is that J-type instructions do not have COND bits. This means they always execute.

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 a 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 I/O memory interface to include the additional I/O devices, and add to the control unit to execute J-type instructions, as well as any modification to the datapath to handle these instructions.

Tasks you must perform

- Look over and develop an understanding any provided component(s)
- Add the additional I/O devices to your I/O memory interface
- Add to the control unit to J-type instructions and any modification to the datapath.
- Hand-assemble instruction to test these new I/O devices and new J-type instructions.
- Create a test script(s) (.do file(s)) and run a simulation(s) of the processor to make sure the new I/O devices work and J-type instructions work.

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.

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.