EE 4490 Project #2 Assignment

due date: November 15, 11:59pm

Purpose: Gain familiarity with basic computer architecture and how it can be implemented using Verilog. Use the ARMv4 processor as an example computer architecture and demonstrate the ability to adapt a moderately large hierarchal Verilog implementation of the ARM to include more instructions, using the Digilent Basys 3 FPGA board.

Given: The ARM implementation in Verilog, which implements a subset of the full ISA, is provided as the following file collection, including Verilog source code and support files, on WyoCourses. Look under Verilog Examples, UW_ARM, having the sub-folders SingleCycleUWARM and TopLevelVivadoStuff:

Filename	Description
`alu.v`	Arithmetic logic unit
`controlunit.v`	Control unit
`dmem.v`	Small RAM, using an FPGA register block
`extend.v`	Logic for immediate data extension
`imem.v`	Logic in `case` form of instruction ROM
`regfileDB.v`	Debug form of 32-bit, 15-register memory with R15 pass-thru
`SingleCycleProcessor.v`	Module instantiation, glue logic and wiring of single-cycle implementation
`tb_SingleCycleProcessor.v`	Example test bench to drive simulation of processor design
`smallProg.s`	Short source of ARM assembly instruction program to test basic processor function
`Hex27Seg.v`	Standard encoder for seven-segment displays on Basys 3
`Mux4Machine.v`	Standard time multiplexer of Basys 3 seven-segment displays
`TopLevelSCUWARM.v`	Top level module for exposing single-cycle processor with debug to the I/O of the Basys 3
`TopLevelSCUWARM.xdc`	Xilinx Vivado design constraints file specifying pin location assignments for the `TopLevelSCUWARM` debug user interface and clocking

(continue to the next page)

The user interface for the Basys 3 board, as defined by this implementation, is summarized in the following table:

Basys 3 Resource	User Interface Application
BtnC	ARM processor system clock (use to step through each instruction execution)
SW0	ARM `reset`, brings program counter back to start value, must clock once to achieve this
SW15-SW12	Select which register to display on the four seven-segment displays. Register identity is encoded as a four-bit binary nibble.
SW11	Identify which portion of the specified register to display on the seven-segment displays (Lower 16-bits if Low, Upper 16-bits if High).
Seven-Segment Displays	The hex encoding of the selected 16-bit quantity. CAUTION: R15 will be PC+8.

Procedure: Your task is to design, simulate and test in hardware (on the Digilent Basys 3 FPGA prototyping board) an implementation of the single cycle ARM processor discussed in class, in the following manner. Points listed below are associated with the score (on a 100-point scale) to be assigned to your project; take the steps in the order listed below.

1. (up to 10 points) Obtain the Verilog and design constraint files described above, place them in a working folder on your computer. Complete a simulation using iverilog. Note: your iverilog simulation should only involve the files in the sub-folder SingleCycleUWARM, not the files in the sub-folder TopLevelVivadoStuff. Provide documentation that shows you completed this step.

2. (up to 20 points) Examine the ARM program described in the assembly language source file named smallProg.s and confirm that the corresponding machine language is implemented in the embedded processor instruction memory imem.v. **Decipher what the "program does" by determining the final values of machine registers R2, R3, R4, R5** and **R15** once the "program" reaches the infinite loop at the end of the code. Provide documentation that shows you completed this step.

3. (up to 20 points) Import the design from both sub-folders SingcleCycleUWARM and TopLevelVivadoStuff into a new Xilinx Vivado project for the Basys 3 board (caution: don't let the test bench be part of the Synthesis step). Synthesize and implement the design, then download to the Basys 3. Use the debug switches and displays, as described above, to verify that you can single step the embedded ARM processor and view the register contents at each instruction execution. Provide documentation that shows you completed this step.

4. (up to 10 points) Modify the top-level interface to provide a means for viewing, on the four seven-segment displays of the Basys 3 board, the current value of the "instruction" (in machine code) being executed. Provide full source code of the Verilog files modified to achieve these changes. (**Fully document, including photos of the Basys 3!**)

5. (up to 15 points) Add an instruction: implement the CMPcd instruction of the ARM ISA, then confirm its operation with a simple program in the resulting system design.[1] Write an assembly language program *.s that includes CMPcd and run it both on the ARMSim and on your design to show it works. Include the source of the assembly language program you used for testing. (**Fully document, including photos of the Basys 3!**)

6. (up to 25 points) Add an instruction: implement the EORcdS instruction of the ARM ISA, then confirm its operation with a simple program in the resulting system design. Write an assembly language program *.s that includes EORcdS and run it both on the ARMSim and on your design to show it works. Include the source of the assembly language program you used for testing. (**Fully document, including photos of the Basys 3!**)

Turn in: For this Project, turn in electronically (as one zip-file attachment) the following items. Submit the zip file on WyoCourses. The zip file must contain:

1. A PDF file providing a succinct technical description of the work your team did in order to complete steps 4-6 above. This document contains the documentation and photos mentioned for steps 4-6 above. Furthermore, this document should include the following:

- An "RTL Schematic" of the top-level design of your implementation, and of the next level down (i.e., the SingleCycleProcessor level).
- An inventory (tabular list, for example) of the Basys 3 user-interface resources which are utilized by your design. For example, which switches, buttons and Basys 3 LEDs are utilized for the user-interface functionality.
- A brief description of the modifications to any individual Verilog modules provided to you and a description (in a black-box sense) of any new Verilog modules you created.

2. Your zip-file must also include the following (in a separate folder within the zip file):

- All Verilog source files (filename extension .v) and assembly language program source files (filename extension .s) utilized in the implementation of your project. You should also include any test benches and simulation results to demonstrate the desired behavior of your appropriate modules.
- The Xilinx Vivado design constraints file (filename extension .xdc) utilized by your project.
- The bitstream file (filename extension .bit) produced by Vivado to program the FPGA.

These specific files (and ONLY THESE FILES, no other content) should be placed in a zipped archive file with the following name:

EE4490_Proj02_Lastname1_Lastname2.zip

The two students for the team should substitute their last names in the obvious two positions of the file-name. Note that the separators in the filename are underscore characters, not spaces. This file must be submitted no later than 11:59 pm on Sunday November 15, 2020.

[1]A suffix such as cd means the instruction is subject to conditional execution specified by the most significant four bits of the machine code. The suffix S, if present, means the programmer is directing that the status flags will be updated.