Cpr E 281 FINAL PROJECT

ELECTRICAL AND COMPUTER ENGINEERING
IOWA STATE UNIVERSITY

FINAL Project

Objectives

The main objective of the final project is to teach you how to put together all of the class material that you have learned so far in order to program the Altera DE2 board to carry out an independent activity. You will have to design a circuit to realize one digital machine that accepts input from switches and outputs to LEDs and 7-segment displays.

Project Selection

Feel free to choose any one of the following three projects. Your grade will not depend on the project that you pick, as long as you implement the entire project.

If you don't like any of the provided projects, then you also have the option of specifying your own project. If there was something that you always wanted to do in digital design, now is your chance. However, your proposed project MUST be of appropriate complexity as determined by your lab TA. The minimum constraints are that y*our project MUST: 1) include a state machine, 2) include a register file, 3) a combinational logic, and 4) be demonstrated on the ALTERA boards provided in the lab.*

Notes on the Three Provided Projects

The first two project descriptions outline all necessary details for these projects. If you choose to implement one of these projects, **you must follow** all details listed in the project description. Your implementation, at a minimum, **should satisfy** all implementation features and constraints as specified in this document. This means that if you do not implement a certain feature or reduce the complexity of the project you will lose points as indicated in the grading **RUBRIC**. You are allowed to work beyond the basic functionality as long as the implementation improves upon the specified design. It would be helpful to discuss any such changes with your TA.

Think about the final project as a really long homework. You may have to go to the lab outside of your regularly scheduled lab time in order to complete it. To get a grade for this assignment you need to present your final project to your TA during dead week (in your regular lab time). No presentation, no grade! Also, there will be ZERO credit for any design step in the grading rubric that does not work or produces the wrong output or is not of the given specifications. All components of your design MUST WORK and produce the output as described in the specs to receive credit.

Additional Notes on Verilog Usage

While it is acceptable to use Verilog in the final project, usage of Verilog outside of the functional circuitry that we have used in class is not allowed. Examples: usage of division and exponentiation circuitry (including the /, %, and ** operators), implementation of an algorithm completely within Verilog, thus subverting the register file requirement, usage of loop structures to implement redundant architecture that should be reused with an FSM, etc. If you have any questions about particular usage, please ask your TA if your code is suitable.

Project #1: Circuit for checking if a list of numbers is sorted

See the grading RUBRIC below before starting with the design.

In this project you will implement a finite state machine and a circuit that allows a user to enter a list of numbers into a register file, and then press a button to determine if the list is sorted in increasing order. This circuit for this project will have two modes that are described below.

Initialization Mode: During this stage the numbers are loaded one by one into the register file. Two values will be provided by the user through the switches on the Alterra board: the address and the number. The address specifies in which register the number should be stored. A LOAD button or switch is used to load each value into the register file. Once all the numbers are loaded into the register file, a different switch is used to change modes. (Note that the loading may be out of order.)

Checking Mode: A VALIDATE button or switch needs to be pressed to begin the validation process. The machine should iterate through the registers and compare each with its neighbor in the address space. If the contents of the two registers are not sorted in increasing order, then the machine should stop the iteration and indicate with the LEDs or in some other way the position of the exception. That is, the address or location of the first register that has a lower value than the preceding register.

Your circuit must have a register file with eight 4-bit registers. The values stored in the registers must be visualized on the eight 7-segment displays. The rest of the datapath should contain the comparator circuit that is used for checking and any other components that may be necessary.

RUBRIC

Each of the following steps must be in its **own design file**. Demonstrate each step individually to receive credit.

a. Design and demonstrate a register file with one write port and two read ports. Show that you can write and read from the register file, including reading from two registers during the same clock cycle. **(20 points)**

b. Design and demonstrate the comparator circuit. **(20 points)**

c. Design and demonstrate one of the modes of the finite state machine. **(20 points)**

d. Put together all of the individual components from Parts a, b, and c and demonstrate that the complete task with both modes can be performed. **(20 points)**

e. Submit a written final report. **(20 points)**

Project #2: Stack Arithmetic

See the grading RUBRIC below before starting with the design.

For this circuit, you will need to implement a simple finite state machine to control a stack of 4-bit unsigned integers. If you are not familiar with the stack abstract data type, check http://en.wikipedia.org/wiki/Stack_(abstract_data_type) . You have to design this stack so that its contents can be used to perform some simple arithmetic. The user must be able to perform four operations:

- **Push** - A value is added to the top of the stack.
- **Pop** - The top-most value is removed from the stack.
- **Pop with Add** - The top two values on the stack are popped from the stack, added together, and the result is pushed back onto the top of the stack.
- **Pop with Subtract** - The top two values on the stack are popped from the stack, used to perform subtraction, and the result is pushed back onto the top of the stack. (The first value popped is subtracted from the second value popped.)

To keep things simple, the maximum stack depth will be four. Your circuit should display the complete contents of the stack on HEX3 through HEX0. To be clear, HEX3 should display the first value pushed onto the stack, HEX2 should display the second value, and so on. The seven-segment display should have no lit LEDs if the stack is empty.

Use the switches on the board to specify the numbers that are pushed. Switches can also be used to specify the type of operation that should be performed.

Some notes about error handling:

- **Stack Overflow** - If a push is performed, but there is no more room on the top of the stack, the contents of the stack should not be changed. Instead, an error should be indicated with an LED or some other way to notify the user. This indicator should remain set and lit until the board is reset.
- **Stack Underflow** - If any of the three pop operations are performed without enough values on the stack to perform that operation, then the contents of the stack should not be changed. Instead, another error should be indicated but through a different visual indicator. This error signal should remain set and lit until the board is reset.

Some hints:

- You may find it convenient to implement the stack using a 4-bit register file with 4 registers, two read ports, and one write port.
- If there is only one value on the stack, only HEX3 should display a value. The other HEX displays should not have any lit LEDs.
- Select which switches or keys you will use to enter the numbers and specify the operations. Make sure to document that in your final report.

RUBRIC

Each of the following steps must be in its **own design file**. Demonstrate each step individually to receive credit.

a. Demonstrate that you can load values into the stack using the **PUSH** operation. Display the stack contents on the 7-segment displays. This can be done by using the appropriate switches to push some random initial values into the register file. **(20 points)**

b. Demonstrate the **POP** operation by popping the values pushed in Part a. **(20 points)**

c. Design and demonstrate **POP with ADD/SUBTRACT**. **(20 points)**

d. Put together all of the individual components from Parts a, b, and c and demonstrate the complete stack machine with **error handling**. **(20 points)**

e. Submit a written final report. **(20 points)**

Project #3: Do something cool with the i281 CPU

Some possible ideas that would be considered "cool" include:

- Improve or simplify the design in some non-trivial way.
- Pick a popular algorithm, implement it in assembly, and execute it on the CPU.
- Extend the OP-CODES by adding a new instruction, e.g., a new branch or jump.
- Rewrite the assembly to machine code compiler in Python or some other language.
- Find a bug in the hardware design that is reproducible.
- Extend the data and code memory size and write a two-player version of PONG.
- Anything else that you think would be cool or awesome.

This project is deliberately left open ended, so that it does not constrain your creativity.

Because it is so open ended, however, this project requires instructor or TA approval. Please write a short 2-3 paragraph description of what you want to do and send it to your instructor and TAs for approval. The purpose of this is to make sure that you don't try to do something that is too time consuming or too out of range for the skills that you currently have.

Project #4: Specify Your Own Project

See the grading RUBRIC below before starting with the design.

Propose a project of your own design. Once again, the minimum constraints are that your project MUST: 1) include a state machine, 2) include a register file, 3) a combinational logic, and 4) be demonstrated on the ALTERA boards provided in the lab.

If you choose this option, you **MUST** discuss your intended design with your lab TA to see if the project would be appropriate for this class. This discussion can be over e-mail. Please contact your lab TA.

If the lab TA or Instructor evaluates the project and recommends that it is not of appropriate difficulty (either too easy or too hard), then the project proposal will be denied and you'll have to modify it or propose a new one.

RUBRIC
Each of the following steps must be in its **own design file**. Demonstrate each step individually to receive credit.

a. Design and demonstrate the register file for your design. **(20 points)**

b. Design and demonstrate the state machine for your design. **(20 points)**

c. Design and demonstrate the combinational circuit. **(20 points)**

d. Put together all of the individual components from Parts a, b, and c and demonstrate the complete project. **(20 points)**

e. Submit a written final report. **(20 points)**