SANTA CLARA UNIVERSITY	ELEN 127 Fall 2018	Jim Lewis

Laboratory #3: Register File

For lab section on October 12, 2018

I. OBJECTIVES

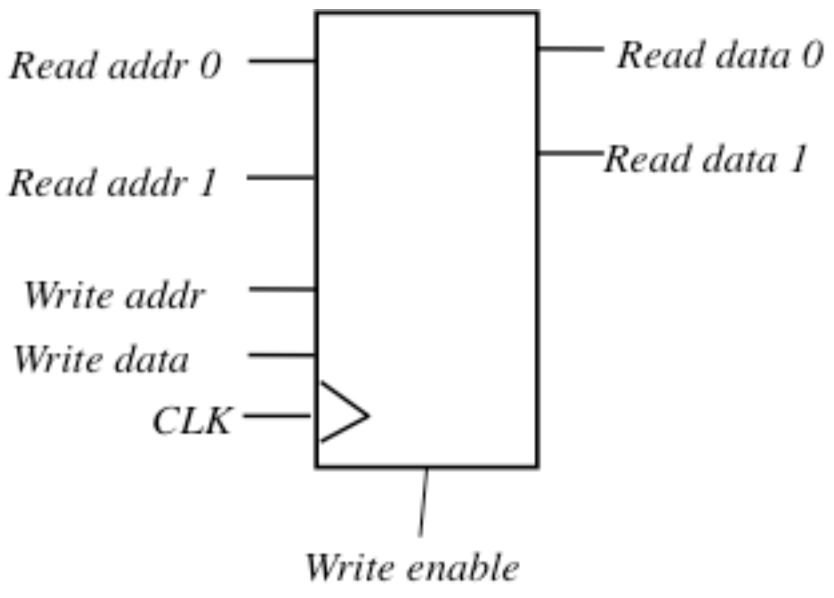
Implement a building block 8-bit register with load enable.

Implement an 8-entry register file with two read ports and one write port, along with a bypass path for writes to a value that is being read at the same time.

First exposure to the use of VCS for the purpose of simulating/testing your design.

PROBLEM STATEMENT

As a first step toward implementing a simple CPU, we need to implement a register file with 8 registers in it. The datapath will only be 8-bits wide. Because we will need an 8-bit register with load enable for other labs, we will build this as a basic building block and instantiate it multiple times within the register file.



PRE-LAB

(i) Run thru the VCS tutorial provided in the Camino file repository.

(ii) Define the interface to a Verilog module that will represent the block diagram above. You don't need to use the exact names in the diagram; you can call the ports whatever you want. But it should represent the concept, including which ports are inputs, which are outputs, and what the width of each port is (given the problem statement).

(iii) Define how your bypass logic will work. This can be just a block diagram or rough pseudocode of how the logic will actually be implemented in Verilog.

(iv) Outline a handful of important testcases that you will run against your design to make sure it's working properly.

LABORATORY PROCEDURE

You will be partnered. One of you will work on step 1, the other on step 2.

1. Implement the register file, using an 8-bit register with load enable:
   (i) Define and implement a module that takes an 8-bit input and provides an 8-bit output, where the output reflects the input after the rising edge of a clock only if the load enable is asserted.
   (ii) Using the interface you defined in part (ii) of the prelab as the starting point for your module, instantiate 8 instances of the 8-bit register you implemented in part(i), along with the bypass logic you defined in part (iii) of the prelab.

2. Implement a testbench:

   (i) Using a template that will be provided by the TA, create a testbench module that instantiates the register file and that sets up the testing constructs in the template.

   (ii) Create a unified set of testcases that merges what you identified in part (iv) of the pre-lab and the set of testcases your partner identified. Map these testcases into a set of stimuli to be applied by the testbench, using the constructs of the template.

   (iii) Once your partner has step 1 completed, simulate the entire thing and determine whether the design is functioning properly. Fix it until it does.

   (iv) Once you believe it's working properly, demonstrate your test results to the TA.

REPORT

For your lab report, include the source code for your register file and for the testbench you used to verify the behavior. In addition, include answers to the following questions.

- What problems did you encounter while testing your steps yourself?
- Did any problems arise when demonstrating for the TA? What were they? Explain your thoughts on how/why these testcases escaped your own testing.