UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering

EEC180B **DIGITAL SYSTEMS II** **Fall 1999**

Lab 2: Register File Design

Objective: In this lab you will design and simulate a Register File that will be used later in the design of a pipelined processor.

I. Register File Specifications

A Register File is an integral part of a microprocessor. Registers are used to store operands for ALU operations, addresses for branch instructions and data in the form of intermediate values during a computation. The number of registers and the width of each register characterize a Register File. An *M*x*N* Register File has *M* registers, each of which is *N* bits wide. Typically, a Register File supports two operations, namely, *read* and *write*. A *read* operation involves placing the data value within a given register onto an output bus, while a *write* operation involves storing a given value into a specified register.

In this lab, you will design and simulate a Register File consisting of **eight 16-bit registers**, R0 – R7. Register R0 should be hardwired to zero. That is, reading from R0 should always give the value 0 and writing to R0 should have no effect. The external interface of the Register File is shown in Figure 1. There are two *read* buses, A and B, and one *write* bus, WB. There are also three three-bit *address* buses, A_add, B_add, and WB_add. Each of these address buses is used to specify one of the 8 registers for either reading or writing. The write operation takes place on the rising edge of the clk signal when the wr_en signal is logic 1. The read operation, however, is not clocked - it is combinational. Thus, the contents of the register specified by the A_add bus should always be on the A bus. Similarly, the contents of the register specified by the B_add bus should always be on the B bus. So, with this Register File, you can write into a register and read *two* registers *simultaneously*. It is also possible to read a single register on both of the read buses simultaneously.

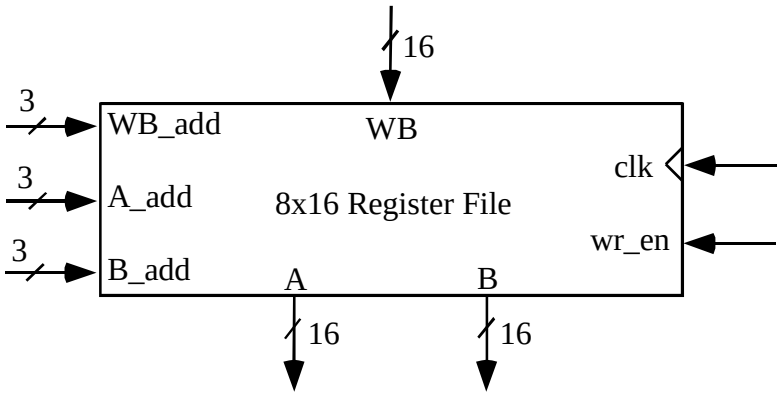

Figure 1: External Interface of the Register File

II. Lab Requirements

1. Design the Register File specified above using the Altera Max+Plus II CAD package. Create a symbol for your Register File so that it can be used in a test circuit as well as in future labs.

2. Compile your Register File for a Flex 10K device. Verify your Register File circuit by performing a timing simulation in which you write to each register and read from each register on both of the read ports, A and B. Generate a printout of your simulation wave forms.

III. Lab Write-up

Have your TA verify your timing simulation and then sign a verification sheet. For your lab report, include the following:

- Signed TA verification sheet.
- Schematic of your Register File circuit
- Simulation waveforms produced by your timing simulation