

ECEN 449: Microprocessor System Design
Department of Electrical and Computer Engineering
Texas A&M University

Assignment #2

Due Friday, October 05, 2007

1. [15 points.]

Write a program to multiply two matrices A and B of unknown sizes. The sizes of the matrices will be given as input. The input matrices will be given in two files `inputA.txt` and `inputB.txt`. The first line of the input file will contain two integers: number of rows, number of columns of the matrix in the file. The entries of the matrix will be listed on the next line in a row-major order. Assume the entries of the matrix are floating point numbers. Print the product matrix C into an output file `outputC.txt` in the same format as the input files. Provide comments in your code.

2. [15 points.]

Read the manual pages on the `open()` and `mmap()` function calls. You can either use "man open" or use the web to find information about these function calls. `mmap()` allows you to map the file contents to memory address spaces and then you can write to the memory addresses to write to the file and similarly reading from memory causes data to be read from the file.

Write a program using `open()` and `mmap()` to create a file containing integers from 1 to 100. Your program should write to file by writing to memory. Provide comments in your code.

The data in memory is in binary format and hence the data in the file will also be in binary.

3. [10 points.]

Construct a Verilog module for a controller. Your code must be written for synthesizability.

The controller has 3 states: *RESET*, *WAIT_FOR_ACK* and *FAIL*. The model starts up in *RESET*. When the signal *have_data* is high, data is sent and the model moves to state *WAIT_FOR_ACK*. When *have_data* is low, we wait in *RESET*. Once in *WAIT_FOR_ACK*, when *ack* is low, we continue in that state. If *ack* is high, we move to *RESET* again. If *have_data* is high when in

WAIT_FOR_ACK, we move to the *FAIL* state. Similarly we move to the *FAIL* state when *ack* is received in *RESET*.

Provide comments in your code.

4. [10 points.]

Construct a Verilog module for a memory, along with a testbench for the same. Your model should have the following properties:

- The memory has 8 address bits as input, and is 4 bits wide.
- There is an external *CLOCK* signal, such that upon any event on *CLOCK*, the data contained in the memory location referred to by the address is read from or written to the memory, if the state of the input *Read/Writebar* is 1 or 0 respectively.
- The testbench must write a pattern of 0101 to each location of memory and read it back immediately. This should be done for each memory location. After this, the testbench must write a pattern of 1010 to each location of memory and read it back immediately. This should be done for each memory location as well. If any of the read operations fails, the testbench should provide a message indicating failure, otherwise it should provide a message stating that the tests passed.

Provide comments in your code.