ECEN 449 749: Microprocessor System Design
Department of Electrical and Computer Engineering
Texas A&M University

Homework #2

Due : Wednesday, February 28, 2018

Upload your homework solution to ecampus as a **single** file. Your homework solution should include a listing of any C code or Verilog code, along with any output obtained when the code is run. **DO NOT** upload a zip file. You will have only **ONE** attempt to upload your homework solution.

In addition, your source code (any C code or Verilog code or testbenches) should be sent via email to 449and749graders@gmail.com. Your email title should state your NAME, SECTION NUMBER and HOMEWORK NUMBER. Name your files in a way that identifies the homework number and the question (e.g. hw1-Q1b.v). For all the code that you write, please provide comments for full credit.

1. [10 points.]

   A and B are two dimensional matrices. Write a C program to add the transpose of matrix A and transpose of matrix B. For both A and B, the size of the matrix will be given along with the entries of the matrix in two input files, inA.txt and inB.txt. The first line of the input file will contain the number of rows followed by the number of columns of the matrix. The entries of the matrix are listed on the next line in row-major order. Print the output matrix C to outC.txt in the same format as input files. Provide comments in your code.
   Example: If your matrix A is

   $$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

   then inA.txt would be:
   2 3
   1 2 3 4 5 6
   Run your program on the following pairs of matrices:

   (a)

   $$A = \begin{bmatrix} 1.1 & 2.53 \\ -2.1 & -3.3 \end{bmatrix}, B = \begin{bmatrix} 4.5 & -5.67 \\ 3.73 & 0 \end{bmatrix}$$

(b)

$$A = \begin{bmatrix} 1 & -1 & 0 \\ -2 & 1 & -1 \end{bmatrix}, B = \begin{bmatrix} 2 & 1 & -1 \\ -1 & 2 & 1 \end{bmatrix}$$

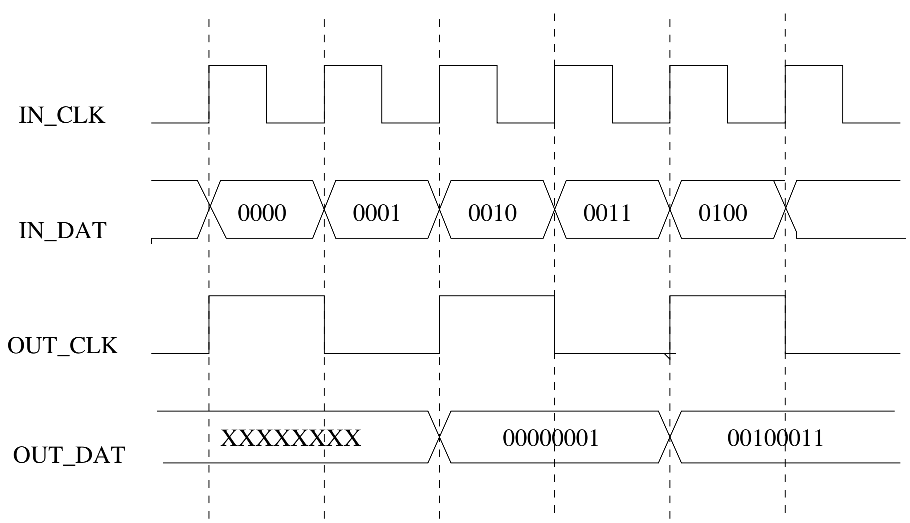
2. [10 points.]

   Construct a Verilog module for a parity generator. It has an 8 bit wide input "IN". The output "OUT" is 1 for even parity else 0. Simulate your parity checker module using a testbench, for the following inputs:

   (a) 10101010
   (b) 11111111
   (c) 10000010

3. [15 points.]

   Consider an input clock IN_CLK. Also consider an signal IN_DAT which is 4 bit wide, arriving at every positive edge of IN_CLK. Write Verilog code to generate the following: Output clock OUT_CLK and signal OUT_DAT which is of 8 bits wide, as shown in the figure below. The OUT_DAT signal packs the last 2 values of IN_DAT that the system encounters. Test your code for 16 clock cycles, with the input IN_DAT = 0000 for first clock cycle, 0001 for the second clock cycle, and so on.



Waveform for Question 3

4. [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 space 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 the string *"Hello hola how are you"*. Your program should write to file by writing to memory. Provide comments in your code.

5. [10 points.] [GRADUATE QUESTION]

   Write Verilog code to implement a LIFO (Last-In-First-Out) buffer as shown in the figure below. The LIFO buffer can store upto 8 Bytes. The input clock signal is called CLK. The $R/\overline{W}$ line is used to select between LIFO Read and LIFO Write. When $R/\overline{W}$ is high, the LIFO performs a read and when $R/\overline{W}$ is low, the LIFO performs a write. The output FULL line becomes high when the LIFO buffer is or becomes FULL. Similarly, the EMPTY line becomes high when the LIFO buffer is or becomes EMPTY. The signal RST resets the LIFO. Test your code for 16 clock cycles, with DATAIN = 00000000 for the first clock cycle, 00000001 for the second clock cycle and so on upto 8 clock cycles. The data is written into LIFO for the first 8 clock cycles. Then perform a Read ($R/\overline{W}$ = high) for the next 8 clock cycles. Plot waveforms of all the inputs and outputs.

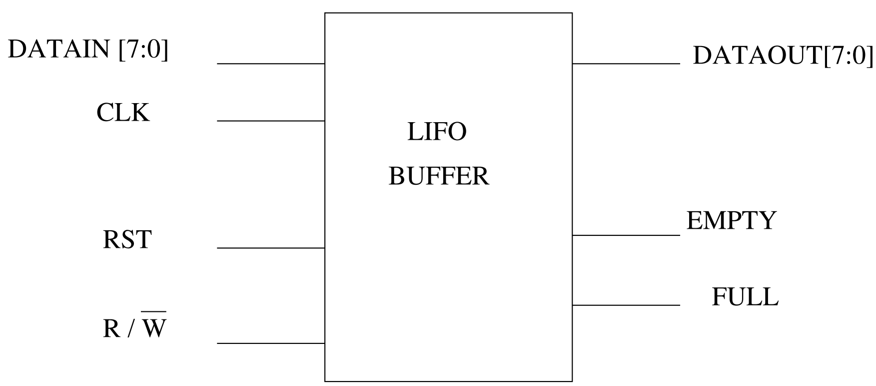


Figure for Question 5