Khanh Nguyen
UIN 525000335
ECEN 449-503
Dr.Khatri

Homework 2

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.

Answer:

C code:

```
#include <stdio.h>
#include <stdlib.h>

void add(float A[][10], float B[][10], float C[][10], int r, int c)
{// This function adds two matrixes
   int i, j;// i-index for row, j-index for columns
   for(i=0; i<r; i++)
   {
      for(j=0; j<c; j++)
      {
         C[i][j] = A[i][j] + B[i][j];
      }
   }
}

void transpose(float arr[][10], float temp[10][10], int r, int c)
{// This function transposes matrix
   int i, j;// i-index for row, j-index for columns
   for(i=0; i<c; i++)
   {
     for(j=0; j<r; j++)
      {
```

```
            temp[i][j] = arr[j][i];// rows will become columns and vice versa
        }
    }
}

void print(float arr[][10], int r, int c)
{//This function prints the array to the console, just for easy checking
    int i, j;// i-index for row, j-index for columns
    for(i=0; i<r; i++)
    {
        for(j=0; j<c; j++)
        {
            printf(" %.2f ", arr[i][j]);
        }
        printf("\n");
    }
}

void load(char fileName[], float arr[][10], int *r, int *c)
{//This function loads data from txt file
    int i, j; // i-index for row, j-index for columns
    FILE *fp;
    fp = fopen(fileName,"r");
    fscanf(fp, "%d %d", r, c); //read first line containing rows and columns
    for(i=0; i<*r; i++){
        for(j=0; j<*c; j++)
        {   //load numbers into an array from second line
            fscanf(fp, "%f", &(arr[i][j]));
        }
    }
    fclose(fp);// cloase file
}

void write(float arr[][10], int r, int c)
{// This function writes result to file
    int i, j;// i-index for row, j-index for columns
    FILE *fp;
    fp = fopen("outC.txt", "w");
    fprintf(fp, "%d %d\n", r, c); //write first line containing rows and columns
    for(i=0; i<r; i++)
    {
```

```
      for(j=0; j<c; j++)
      {   //write the result array to file second line
         fprintf(fp, "%.2f ", arr[i][j]);
      }
   }
   fclose(fp); // close file
}

int main()
{
   int rA, cA, rB, cB, rC, cC;

   float A[10][10], B[10][10], C[10][10];

   load("inA.txt", A, &rA, &cA);
   load("inB.txt", B, &rB, &cB);

   rC=rA;// set dimensions for matrix C
   cC=cA;

   float transposed_A[10][10];
   float transposed_B[10][10];

   printf("\n Array A: \n");
   print(A, rA, cA);

   printf("\n Array B: \n");
   print(B, rB, cB);

    if( (rA != rB) || (cA != cB))// check if dimensions matched
   {
      printf("\n Dimensions not matched!\n");
      return -1;
   }
         transpose(A ,transposed_A , rA, cA);
         transpose(B ,transposed_B , rB, cB);
   add(transposed_A, transposed_B, C, cC, rC);

   printf("\n Array C: \n");
   print(C, cC, rC);
   write(C, cC, rC);
```

```
    printf("\nCheck outC.txt for result\n");
    return 0;
}
```

Result: (It's just for easy checking when it's printed on the console, please check the outC.txt file for output)

Pair a:

```
[k41nt]@hera3 ~/ECEN449/hw2/q1> (16:25:15 02/28/18)
:: gcc hw2_q1.c

[k41nt]@hera3 ~/ECEN449/hw2/q1> (16:25:21 02/28/18)
:: ./a.out

 Array A:
 4.50  -5.67
 3.73  0.00

 Array B:
 1.10  2.53
 -2.10  3.30

 Array C:
 5.60  1.63
 -3.14  3.30

Check outC.txt for result
```

Pair b:

```
[k41nt]@hera3 ~/ECEN449/hw2/q1> (16:25:23 02/28/18)
:: gcc hw2_q1.c

[k41nt]@hera3 ~/ECEN449/hw2/q1> (16:28:32 02/28/18)
:: ./a.out

 Array A:
 1.00  -1.00  0.00
 -2.00  1.00  -1.00

 Array B:
 2.00  1.00  -1.00
 -1.00  2.00  1.00

 Array C:
 3.00  -3.00
 0.00  3.00
 -1.00  0.00

Check outC.txt for result
```

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

Answer:

Verilog code:

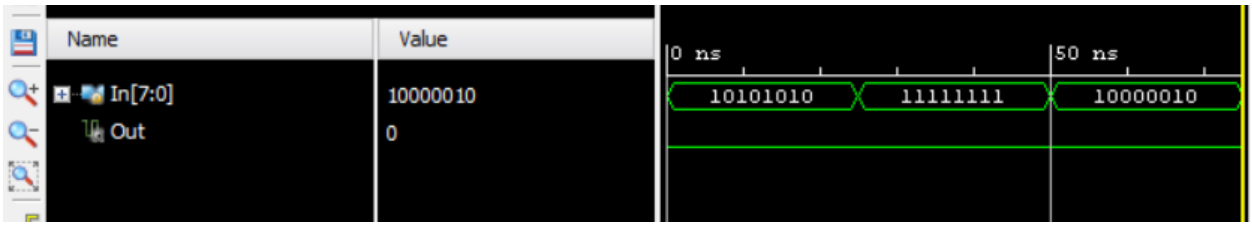
```
module hw2_q2(Out, In);
input [7:0] In;
output reg Out;
integer i = 0;
integer cnt = 0;

always@(In)
begin
for(i = 0; i < 8; i =i+1)
   begin
      if(In[i] == 1)//count the number of 1
         cnt = cnt + 1;
   end
   if((cnt % 2) == 1) //If the number of 1s is odd, parity is 1
      Out = 1;
   else
      Out = 0;//parity is 0 if the 1's are even
   end
endmodule
```

Test bench:

```
module hw2_q2_tb();
reg [7:0] In;
wire Out;
hw2_q2 uut(.In(In),
          .Out(Out)
          );
initial
   begin
      In = 8'b10101010;//test 1
```

```
    #25 In = 8'b11111111;//test 2
    #25 In = 8'b10000010;//test 3
    #25 $stop; // stop after third test
  end

initial
  begin
    $monitor($time," Input = %b, Parity check = %d", In, Out);
  end
endmodule
```

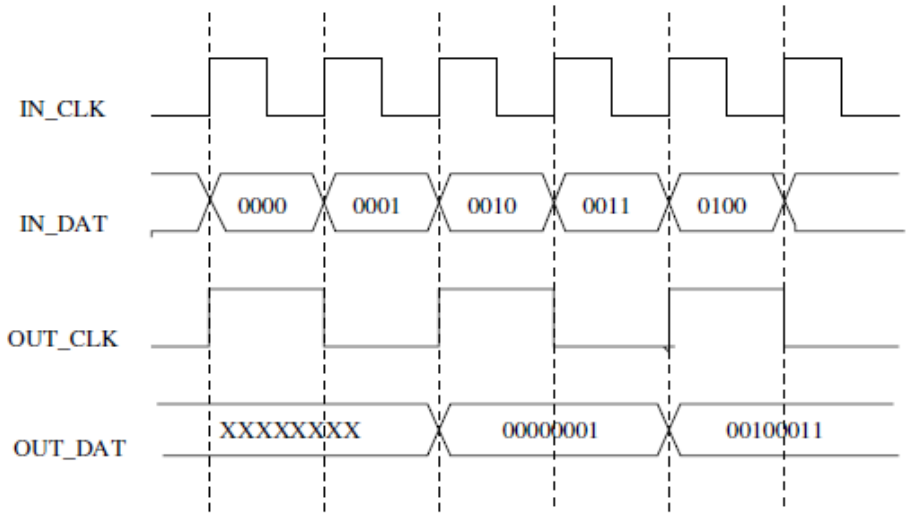
Results:




```
# run 1000ns
                   0 Input = 10101010, Parity check = 0
                  25 Input = 11111111, Parity check = 0
                  50 Input = 10000010, Parity check = 0
INFO: [USF-XSim-96] XSim completed. Design snapshot 'hw2_q2_tb_behav' loaded.
```

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

Answer:

Verilog code:

```
`timescale 1ns / 1ps
`default_nettype none

module hw2_q3(CLK, IN, CLK_OUT, OUT);
	input wire CLK;
	input wire [3:0] IN;
	output reg [7:0] OUT;
	output reg [3:0] CLK_OUT;

	initial CLK_OUT = 0;
	initial OUT = 8'bXXXXXXX;

	always@(posedge CLK)//CLK_OUT changes at pos edge
	begin
		CLK_OUT <= ~CLK_OUT;
	end

	always@(posedge CLK_OUT)//OUT changes at CLK_OUT
	begin
```

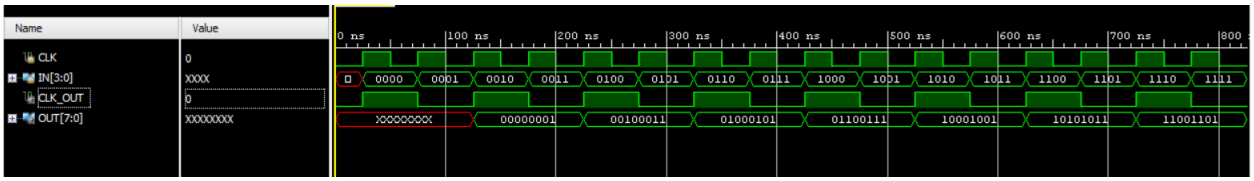
```
            #100 OUT<={IN-4'b0010,IN-4'b0001};
        end

endmodule // hw2_q3
```

Test bench:

```
`timescale 1ns / 1ps
`default_nettype none
module hw2_q3_tb;
        //Inputs
        reg CLK;
        reg [3:0] IN;
        //Outputs
        wire [7:0] OUT;
        wire CLK_OUT;

        //Instantiate the Unit Under Test (UUT)
        hw2_q3 uut(
                .CLK(CLK),
                .IN(IN),
                .CLK_OUT(CLK_OUT),
                .OUT(OUT)
        );

        //generate 20MHz clock signal
        always
                #25 CLK = ~CLK; //since 2MHz is 50ns per cycle
                                                //=>#25 half cycle
        initial begin
                CLK=0;
                #24.999 IN=4'b0000; //first input has to start before #25
                #50 IN=4'b0001;
                #50 IN=4'b0010;
                #50 IN=4'b0011;
                #50 IN=4'b0100;
                #50 IN=4'b0101;
                #50 IN=4'b0110;
                #50 IN=4'b0111;
                #50 IN=4'b1000;
                #50 IN=4'b1001;
                #50 IN=4'b1010;
```

```
        #50 IN=4'b1011;
        #50 IN=4'b1100;
        #50 IN=4'b1101;
        #50 IN=4'b1110;
        #50 IN=4'b1111;
    end

    endmodule // hw2_q3_tb
```

Result:



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.

Answer:

C code:

```
#include <stdio.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>

int main( int argc, char *argv[] )
{
   char msg[]="Hello hola how are you";
   int i;
   int msg_size = sizeof(msg);
   int file_end;
```

```
    // Open a file for writing,create if not exist
    int fd = open( "Output_q4.txt", O_RDWR | O_CREAT | O_TRUNC , S_IRUSR | S_IWUSR | S_IRGRP |
S_IROTH | S_IWGRP | S_IWOTH);
    if (fd == -1)   {
        perror("Error: file cannot be opened!");
        exit(0);
    }
    // write to Null
    file_end = lseek(fd, msg_size - 2 , SEEK_SET);
    write( fd , "", 1);
    // mapping file
    void *map = mmap(0, msg_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if (map == MAP_FAILED)    {
        close(fd);
        perror("Error: cannot mmap to file");
        exit(0);
    }
    // write message to file
    strcpy( map , msg);
    printf("Message written to file successfully!\n");
    // free mmapped memory
    if (munmap(map, msg_size) == -1) {
        perror("Error: cannot ummapped memory!");
    }
    close(fd);
    return 0;
}
```

Result: Please check Output_q4.txt for result