EXAM 3 - ECE 287

## Instructor: Peter Jamieson

## Directions:

This is a take home exam that is due on the last day of the exam week.

You are to build the following design in Quartus and implement on a DE2 board with the provided module instantiation. You will need to submit the following:

* This printout with your name on it and any special instructions to run it on the DE2-115 board.
* The answers to the last 2 questions for your design solution on this printout.
* An electronic file containing the Verilog design (one file) via Canvas

## Details:

* There is 1 design question to be answered.
* There are 2 additional questions

## Design Question 1

You are to build the C code onto the DE2-115 or DE2 board (with provided wrapper) that was given to you and implement it in hardware. Each exam is unique and has some of the following operations as described below.

You need to build your module in the provided wrapper that helps you input the 4 values and output the 2 outputs. You also need to send a “done” signal when your code is complete and start on a “start signal” (see the wrapper on Canvas).

**Notes:**

* **Looping and conditionals must be implemented as finite state machines.**
* **Do not optimize the C code.**
* **If you have an infinite loop in the code, change it so that it works, and submit the C code changes**

FUNCTION Descriptions assume x and y are positive numbers and a, b, c, and d, are integers (positive and negative numbers):

* mod(x,y) - the modulo operator can divide one number by another and the result is the remainder. For example, 5 % 2 = 1 … 5 divided by 2 has a remainder of 1. **Note** you cannot use the % symbol in Verilog to implement this functionality and need to build it with gates (can’t use lpm modules either). The code will look like: mod(5, 2) which is 5 % 2.
* div(x,y) – the divider operator can divide one number by another and the result is the remainder. For example, 5 / 2 = 2. **Note** you cannot use the / symbol in Verilog to implement this functionality and need to build it with gates (can’t use lpm modules either). The code will look like: div(5, 2) which is 5 / 2.
* mult(x,y) – the multiplier operator can multiply one number by another and the result is the remainder. For example, 5 \* 2 = 10. **Note** you cannot use the \* symbol in Verilog to implement this functionality and need to build it with gates (can’t use lpm modules either). The code will look like: div(5, 2) which is 5 / 2.
* exp(x,y) – the exponent operator does x^y where 2^3 = 8. You can implement this function with the aid of the \* symbol in Verilog but not ^ or an exponent lpm.
* min(a,b,c,d) – finds the minimum value of four values. For example min(1, -2, 3, 4) would return -2.
* max(a,b,c,d) – finds the minimum value of four values. For example max(1, -2, 3, 4) would return 4.
* sum\_max\_min(a,b,c,d) – finds the greatest number and smallest number and adds them together. For example, sum\_max\_min(1, -2, 3, 4) would return 2.
* pop\_count(a) – pop\_count counts the number of binary 1’s in the decimal number. For example pop\_count(7) returns 3.
* abs(a) – absolute value takes the absolute value of a number. For example, abs(-3) returns 3.
* palindrome(a) – assumes that a is an 16-bit number and returns the palindrome (invert the values around a mirror in the center). For example, palindrome(16’b1000100100000010) returns 16’b0100000010010001.
* nib\_swap(a) – assumes that a is an 16-bit number and swabs the least significant nibble (4-bits) with the most significant nibble. For example, nib\_swap(16’b0000111100111100) returns 16’b1100111100110000.
* rot\_left(x,y) – assumes that x is an 16-bit number and rotates x by a bits. Rotate left means that the most significant bit gets put into the least significant spot and all other bits shift 1 to the left. For example, rot\_left(1,1) returns 2, rot\_left(1,3) returns 4, rot\_left(1,8) returns 1
* is\_odd\_parity(a) – returns a 1 if there is an odd number of binary 1’s and 0 if there is an even number. For example, is\_odd\_parity(2) returns 1.
* is\_even\_parity(a) – returns a 1 if there is an even number of binary 1’s in a. For example, is\_even\_parity(2) returns 0.
* is\_factor\_of(x,y) – returns a 1 if y is a factor of x. For example, is\_factor\_of(4,2) is 1 and is\_factor\_of(4,3) is 0.
* is\_prime(x) – returns 1 if x is a prime number. For example, is\_prime(7) is 1.
* is\_positive(a) – returns 1 if a is positive. For example, is\_prime(1) is 1.

**Additional Questions:**

1. If clk is a 50MHz clock, what is the operating frequency of the circuit and how much time will it take in seconds for your program to complete on your hardware implementation for a given reasonable input set when the start of the program happens?
2. What is the size of your circuit (do not include the wrapper) and how much of the FPGA is that on the DE2-115 board?

## ECE 287 - Take home exam Rubric

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Topic**  **(Weight)** | **Unacceptable**  **(0)** | **Marginal**  **(1)** | **Acceptable**  **(2)** | **Mastery**  **(3)** | **Points** |
| **Design**  **(8.5 marks)** | Little or no grasp of problem. Incapable of producing a successful solution. | Some understanding of problem.   * No compilation of complex design. * Missing some functions and/or controlling state machine. * No citation of borrowed solutions for some part of design | Overall sound understanding of the problem and constraints.   * Has some functions implemented and finite state machine in Verilog. | Clear and complete understanding of design goal and constraints. All functions working and citations included. |  |
| **Extra Question (a)**  **(1 mark)** | No answer or incomplete. | Serious deficiencies in question solution. | Mostly complete and correct solution to additional question. | Complete solution. |  |
| **Extra Question (b)**  **(0.5 marks)** | No answer or incomplete. | Serious deficiencies in question solution. | Mostly complete and correct solution to additional question. | Complete solution. |  |

TOTAL = \_\_\_\_\_\_