<u>CSCI 4203 Lab Assignment 1 (posted 02/07/2017, due 03/02/2017):</u>

The main purpose of this assignment is for you to get familiar with verilog by coding two simple logic circuits.

Problem 1 : 4-bit ripple carry adder (50 points):
Part A (10 points) :
An ALU (arithmetic and logic unit) is a combinatorial circuit used in the execution stage of modern pipelined processors. It can do all kinds of arithmetic operations such as addition, multiplication, division, shift, logical operations and so on. We'd like to implement a simplified version of a 1-bit ALU, which only performs AND, OR and addition, as shown in figure B.5.6 (on page B-29) of the textbook. Please implement a behavioral model of this 1-bit ALU using the given module skeleton. You can read the section "A 1-bit ALU" in Appendix B.5 for a detailed description of the figure.

Part B (40 points) :
An n-bit adder can be constructed by chaining n 1-bit full-adders together. The carry-out output of each 1-bit full adder is connected to the carry-in input of the next full adder, except for the last full adder in the chain.
Please implement a 4-bit ripple carry adder using four 1-bit ALU units. A reference implementation of the ripple carry adder is available in the class slides Verilog.Tutorial.v2.6-up.2017.pdf (pg 3 bottom right)

Problem 2 : 4-bit Carry-Lookahead adder (50 points):
A carry lookahead adder is an optimized n-bit full adder implementation which achieves faster addition than an n-bit ripple carry adder by using multiple 'levels of abstraction'. The simplest carry look ahead adder uses a single layer of abstraction, where it computes *propagate* and *generate* values for each bit position and uses these values to compute carry-outs at each bit position. Even faster addition speeds can be achieved using multiple layers of abstraction. Please read the the description of the carry-lookahead adder in section B.6 for a more detailed description. Then, implement a 4-bit carry lookahead adder using a single layer of abstraction in verilog using the given 6 module skeletons. 4 modules implement the carry-out calculation, one module implements the propagate and generate bit calculation, and one module is the top level module.

The equations for propagate, generate and carry-out for each bit position are printed below for your convenience :

gi = ai.bi
pi = ai + bi
c1 = g0 + (p0.c0)
c2 = g1 + (p1.g0) + (p1.p0.c0)

$c3 = g2 + (p2.g1) + (p2.p1.g0) + (p2.p1.p0.c0)$
$c4 = g3 + (p3.g2) + (p3.p2.g1) + (p3.p2.p1.g0) + (p3.p2.p1.p0.c0)$

Submission instructions :
For each of the questions, you are provided several skeleton verilog files which you need to fill in, as well as a single testbench file. You can unzip the skeleton file using the command →

```
tar -xvf   Lab1_Skeleton.tar.gz
```

Submit all the verilog files (except for the testbench) and a README in a single compressed file Lab1_studentName.tar.gz file. You can place all the verilog files in a directory Lab1_studentName and create a .tar.gz file using the command →

```
tar -czvf Lab1_studentName.tar.gz Lab1_studentName
```

Indicate the compilation instructions as well as the name of the top level module of the submission in a README file. Your solutions will be tested using ISEWebpack on the CSELabs machines and will be evaluated using the provided testbench as well as additional tests which are not available to you. Your score will be assigned based on the number of test cases successfully passed by the submitted code. Partial credit may also be awarded at the discretion of the instructor, so remember to comment your solutions well. Good luck !