Project Assignment #5 **DUE: Wednesday, April 23rd**

In this assignment you will be writing yet another alternative architecture for the 8 bit ALU. You will be using the same ENTITY for the 8 bit ALU that you used in steps 3 and 4. The shell of the file with the testbench (same testbench as step 4) is in file pr_step5.vhdl. There are also ps5_list.do and ps5_wave.do files.

1) The first step is to write VHDL code for three **procedures**

The first is to perform binary addition, *binadd*. This procedure could be set up to only add words of 8 bits, but can also be set up to add words of any length. Write the code to add words of any length (in VHDL terms – unconstrained) with the inputs and output having index 0 as the LSB and rightmost bit (i.e., the inputs/output will use the downto range). Also remember that you need a carry out.

The second procedure to write is a procedure to compute the two's complement of a number, *neg*. Again this should take inputs of any length and 0 is the index of the LSB and rightmost bit. This can be a stand alone procedure or use the add procedure.

The third procedure is a subtract procedure, *binsub*. It takes two bit_vector inputs and a borrow in, producing the difference and the borrow out. Either write a binary subtraction procedure or use the *binadd* and *neg* procedures to do the subtraction. But remember to be careful, it is very easy to get the subtraction wrong using the *binadd* and *neg* procedures.

2) With the procedures done, write the architecture of the ALU. Once again it will be an architecture that contains only a single process statement, i.e., one process statement.

The procedure bodies will be declared in the declaration section of the process of this alternative architecture. Note that when you declare a procedure in the declarative section of a process, only a procedure body is declared and, in fact, can be declared.

Now write the remainder of the process to complete this second alternative architecture. Use a case statement per the following setup:

PROCESS (sensitivity list)

- declaration of binadd, binsub, and neg procedures
- and declaration of OPERSW
- --

```
BEGIN  -- of the process
     •
     •
OPERSW :=  P & K & R;  -- the P,K,& R control signals
CASE  OPERSW  IS
     WHEN "110011111100" =>  Zout<= A;  Cout <= '0'; --opA
     •
     •
     •
     WHEN "001111000110" =>  neg(A, Ztemp, CoutTemp);
                             Zout <= Ztemp; Cout <= CoutTemp;
     WHEN OTHERS         =>  NULL;
END CASE;
     •
     •
```

3) Compile and simulate this description of the architecture. Be sure all tests are run (17 us).

Turn in:

a) copy of the "listing" of the VHDL code
b) copy of a file listing the results of the simulation. This listing should be the same as the results of step 4 except possibly for timing.
c) copy of waveform showing the complete simulation - one page (use Zoom->Full Size)

NOTES: Hint to make your life easier. Get the architecture working for the logic functions, then for addition, neg, and finally for subtraction. You will get error spikes on the addition, neg, and subtraction at the point you are just doing the logic functions but that will be as expected.

For the subtraction do a few 4 or 6 bit examples by hand, construct truth tables and then develop the logic equations.