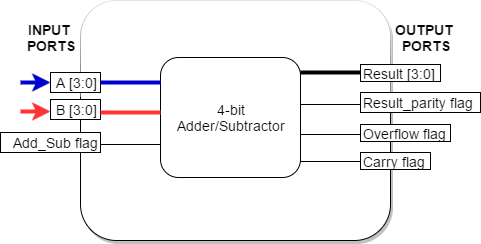
**Digital Design (CSCE 2114)**

**Lab 6 - Part II**

**Objective:**

1. Learn how to write VHDL code to implement some simple combinational circuits.
2. Learn how to simulate your design before implementing it on the FPGA to verify that it works.

**Exercise**

* Implement a 4-bit signed Add/Sub with overflow, carry, and parity flag bits
  + In Quartus, create a new project named lab6\_c .Click on File -> New and select VHDL File.
  + Write the VHDL code to implement a 4-bit adder/subtractor for signed numbers illustrated in the figure below. It should do both addition and subtraction based on the *add\_sub* input. 
  + The port description is as follows:
    - **A, B**: 4-bit inputs.
    - **Result**: 4-bit output. Either (A+B) or (A-B) depending on add\_sub input.
    - **Add\_sub**: If this input is 1, then the Result should be the sum of inputs. Otherwise, the result is A-B.
    - **Result\_parity flag**: This bit indicates whether the output “Result” is an odd or even number. It should go high only when “Result” is an odd number.
    - **Carry flag**: This flag is high when the correct unsigned answer exceeds 15 (or F).
    - **Overflow flag**: It should go high only when the operation leads to an overflow, and therefore the result is wrong. Remember that with 4 bits you can show signed numbers from -8 to +7, so any data outside this range will set the *overflow* flag to high.
  + After finishing your code, compile your project and correct your errors
  + Add a waveform file and add all inputs and the output nodes to your waveform list. Change the radix of A, B, and Result to signed decimal. Set the rest of your nodes to binary. Set up your simulation with the following test bench:
    - Set Add\_sub flag value to a clock signal with 200 ns and keep the 50% duty cycle
    - Apply these inputs to A and B with a 100ns interval.
      * A= -5, 3, 6, 7, 4, -3, -2, -7, -6, 0
      * B= -5
  + Save your waveform file and run your simulation and check the outputs.
  + Show your work to the TA
  + Save a screenshot of both your VHDL Code and your waveform results and include it in your lab report. Remember that in your lab report you should justify why overflow goes high or low for each case. Close the project.

**Hints:**

1. Please, refer to “[VHDL Notes](http://www.csce.uark.edu/~jparkers/CSCE2114-spring2017/index.html)”, “[VHDL Data Types](http://www.csce.uark.edu/~jparkers/CSCE2114-spring2017/index.html)”, and “[VHDL Operators](http://www.csce.uark.edu/~jparkers/CSCE2114-spring2017/VHDL-Operators.pdf)” on class website for VHDL examples and syntax.
2. To test whether or not your Result is an odd or even number, you may consider testing odd/even of your operands first. From calculus, you may remember that adding (or subtracting) two odd (or even) numbers always gives you an even result. You get an odd result if your operands parity is different.
3. When you write A+B in VHDL, the compiler will produce a full adder.
4. To create the output carry bit, think about adding a leading 0 to A and B and make a 5-bit temporary variable to hold the sum. The most significant bit of this 5-bit sum is your carry flag.
5. XOR and XNOR Truth tables:

