EELE 475
HARDWARE AND SOFTWARE ENGINEERING FOR EMBEDDED SYSTEMS

Lab #3
Test and Verification of a Simple FIR Filter

Assignment Date: 9/15/2015

Due Date: 9/29/2015

The goal of Lab 3 is to create a simple FIR filter in VHDL and then test it using the Matlab Verification Toolbox.

The equation of the FIR filter you will create is:

$$y[n] = \sum_{i=0}^{N} b_i x[n-i]$$

where the parameters of the filter have the following constraints:

- **N=2**.
- **Input x[n]** – (W=16, F=8). The input values will have the following data type: Signed fixed-point, word length = 16-bits, fraction length = 8-bits. (W=16, F=8).
- **Coefficients** b_i - (W=16, F=8). The coefficients will be constants with the following data type: Signed fixed-point, word length = 16-bits, fraction length = 8-bits.
- **Output y[n]** - (W=?, F=16). The output data type must ensure that no overflow occurs, i.e. make the word length long enough. Make the fraction length = 16-bits.

Create a VHDL component that performs this calculation. Your component should have the following interface:

Entity Description:

```
entity fir is
   port (
      clk      : in     std_logic;
      x        : in     std_logic_vector (15 downto 0);
      y        : out    std_logic_vector (15 downto 0)
   );
end fir;
```

To implement math in VHDL, you may want to look at the following set of slides:
http://www.synthworks.com/papers/vhdl_math_tricks_mapld_2003.pdf
(which can also be found on D2L).

Setting up Quartus II's Simulation Libraries to be used by ModelSim

(via Matlab's CosimWizard).

1. **In Quartus II:**
   a. Open the EDA Simulation Library Compiler (Tools->Launch Simulation Library)
      i. Under *Tool Name*, Select ModelSim
      ii. Under *Executable Locations*, browse to C:\modeltech64_10.1c\win64 (or where ModelSim resides)
      iii. Under *Library families*, add the Cyclone II (using >)
      iv. Under *Library Language*, select VHDL.
      v. Under *Output Directory*, browse to a folder you have created (e.g. \simlib)
      vi. Push the button Start Compilation.
2. **In Matlab's CosimWizard,** following additional step needs to be performed (See the document *Matlab VHDL CoSimulation* for example steps).
   a. Under the *HDL Compilation* step, add the following line in the *Compilation Commands* window after the line *vlib work*:
   vmap altera_mf c:/..../simlib/vhdl_libs/altera_mf

   Note 1: For the vmap command, the directories need to be separated by forward slashes ('/') rather than back slashes ('\'), since it follows a Linux path specification rather than Windows.
   Note 2: Replace /…./ with the appropriate path on your computer system to the simulation library.

Matlab VHDL CoSimulation

using the Cosimulation Wizard

and the Matlab System Object

Cosimulation Steps:

Files Needed:

fir.vhd (your code to verify, and any dependent .vhdl files)

fir_tb.m (an example Matlab test bench, which is supplied, but you will likely need to modify it to match what you are explicitly doing in hardware).

1. Create a working directory, we will use in this example **\fir_verification**
2. Place VHDL code to be tested (i.e. **fir.vhd,** and any dependent .vhd files) into \fir_verification
3. Change Matlab's working directory to \fir_verification
4. At the Matlab command prompt, run the cosimulation wizard, i.e. >>**cosimWizard**
    a. In the ***Cosimulation_Type*** panel select:
        i. HDL cosimulation with: **Matlab System Object**
        ii. HDL simulator: **ModelSim**
        iii. Select one of the two options
            1. *Use HDL simulator executable on the system path* (this assumes that the path variable has already been setup).
            2. Otherwise select *Use the HDL simulator executables at the following location* and set the HDL simulator installation path (for computers in the digital lab) to: C:\modeltech64_10.1c\win64
        iv. Click Next

    b. In the **->HDL Files** panel:
        i. Add: fir.vhd
        ii. Add: <all other VHDL files that are used by fir.vhd>
        iii. Click Next

    c. In the **->HDL Compilation** panel:
        i. Click Next (i.e. accept the defaults)

    d. In the **->HDL Modules** panel:
        i. Name of HDL module to cosimulate with: **fir**

ii. Click Next

e. In the **->Input/Output Ports** panel:
   i. Click Next (i.e. accept the defaults)

f. In the **->Output Port Details** panel:
   i. Data Type: **Unsigned** (for y) (we will convert the data later)
   ii. Fraction Length: **0** (for y)
   iii. Click Next

g. In the **->Clock/Reset Details** panel:
   i. Active Edge: **Rising** (for clk)
   ii. Click Next

h. In the **->Start Time Alignment** panel:
   i. HDL time to start cosimulation (ns): **0** (Note: if there is a reset, you will want to start after the reset has been de-asserted)
   ii. Click Next

i. In the **->System Obj. Generation** panel:
   i. HDL Simulator sampling period (ns): **10** (Note: just keep this the same as the clock period, i.e. accept the default.)
   ii. Click Finish

5. Matlab creates a bunch of files in \fir_verification and will open the following two files in the Matlab Editor:
   a. **launch_hdl_simulator_fir.m**
   b. **hdlcosim_fir.m**

6. Get the cosimulation ready to run by running the following command at the Matlab prompt: >> **launch_hdl_simulator_fir**
   This will open up ModelSim and create a connection to it.

7. Run the cosimulation by running the function **fir_tb.m** in the Matlab Editor. The **fir_tb** requires the following lines:
   a. `fir_hdl = hdlcosim_fir;` This calls the function hldcosim_fir.m that was created by the cosimWizard, which sets up the simulation object.
   b. `output_vector1 = step(fir_hdl,input_vector1);` The step() function invokes the simulator and passes data to the component under test and receives data from the output. The arguments to the step() function (other than the system object, which is fir_hdl in this case), must

be fixed-point objects created by the fixed-point constructor fi(). The word length must match in the word length of the VHDL input signal. The outputs are fixed-point objects as well. If there are multiple I/O in the component, use: **[out1, out2, out3] = step(rsqrt_hdl, in1, in2, in3);**

c. The rest of the fir_tb function is just to create the appropriate input vectors and then test the outputs (with the appropriate clock cycle latency accounted for).

Instructor Verification Sheet

Turn this sheet in along with your VHDL code to get credit for the lab.

Lab 3

Test and Verification of a Simple FIR filter

Name : _____

Demo: Show Matlab verifying your design over a range of input values.

Verified: _____ Date:_____