17.02.2014

EEE 491 Lab Assignments 2

Fast Fourier Transform (FFT) in VHDL

Last week you have element-wise multiplied two arrays in both MATLAB and VHDL. This week you will calculate the FFT of this multiplication again using both MATLAB and VHDL.

1. Please make sure that your design from the previous lab session works properly, i.e. the values written in the RAM checks with the values you have calculated with the modified MATLAB code. The correct modification that you should have written last week is as in the following line:

   ```
   mult_out = floor(s1FPGA.*s2FPGA/2^8)
   ```

   Remember you have ignored the least significant 8 bits and this corresponds to dividing the multiplication by 2^8 in the integer domain. The “floor” function is used to obtain an integer number after the division.

2. Using MATLAB, calculate the 128-point FFT of the multiplication. Please continue working on Lab1.m that you had been provided last week by doing the modification above and add lines for taking FFT. You will use "fft" and "fftshift" functions of MATLAB. Plot the absolute value of the FFT such that the abscissa shows the frequency (not array index). Interpret the location of the peaks.

3. Now you will implement FFT in VHDL: Continue working on your project from last week. You will write another process which will read the values from the RAM (remember RAM is filled with multiplication values) and will calculate the 128-point FFT. Note that your new process will start after the multiplication is done and so after the RAM is filled with the multiplication results.

4. The implementation of the FFT will be done by using IP core. In order to add an FFT IP core, select "New source" and then select "IP Core". Find "Fast Fourier Transform" under Digital Signal Processing -> Transforms -> FFTs. You may choose Radix-2 Lite, Radix-2 or even Radix-4 architecture. For the selected architecture, please check how many multiplier and block RAM is used from the resource estimation tab on the left bottom of the FFT IP Core GUI. The target clock frequency will be 50 MHz. The FFT IP Core has 8 bits real and imaginary input, 16 bits real and imaginary output (please choose “Unscaled” under “Scaling Options”) and do the 128-point FFT operation. The output ordering can be in the form of “bit reversed order” or “natural order” (We will explain the differences and implications of these forms). Please click the “Datasheet” and read the appropriate places in the document to understand how this IP core works. We will give brief information about the FFT block at the beginning of the lab.

5. Design your new top module such that it has 24-bits output port for the absolute value of the FFT result from the IP core and a single bit output port (fft_data_ready) which indicates that the FFT data is ready. Other ports from the previous lab (clk and mult_out) should also remain. Note that the IP core gives the real and imaginary parts FFT as two separate 16-bits output signals (xk_re and xk_im). Take the most significant 12 bits of these signals so that the multiplication result becomes 24-bits which is suitable for your output.

Test of the Design

6. Last week you have written the multiplication results to RAM and then check the contents of the RAM in the simulation environment to be sure that your design works properly. Another method for the verification of the result would be writing the mult_out signal to a text-file again in the simulation environment. For such purposes, a simulation library named TEXTIO is developed. This week we will work on TEXTIO libraries (Next week, we will work on RS232 Communication as a method for testing the design in the hardware).

7. In order to use TEXTIO, add a test-bench module to your design and associate it with the top module. You will write the absolute value of the FFT samples to a text file by observing the fft_data_ready signal. If this signal is asserted this means the FFT samples are flowing in each clock cycle and so these outputs will be written to the text file. You are given a sample test-bench file (test_FFT_block.vhd). Please examine this file thoroughly and write your own test-bench.

8. Once you complete the simulation (always make sure that you have run the simulation for enough duration), the written text file will appear in your project folder. Write a MATLAB code for importing the txt file to the MATLAB workspace (you may simply write `import fft_output.txt`). Next, plot the imported absolute value of the FFT such that the abscissa shows the frequency (not array index).

9. Finally you will compare your plots from the MATLAB and VHDL. Remember you have truncated the output of the FFT IP core in your VHDL project. Therefore make the appropriate modification to your MATLAB FFT result such that the results from MATLAB and VHDL checks (Hint: Read the first step again).