RTL digital design LAB

Launch the Xilinx Project Navigator development environment. The project directory C:\Esercitazione2\esercitazione2_fir will be directly accessible from the home screen.
The operating mechanism of the testbench provides for reading from the file the filter coefficients (file_coeff.txt) and a sequence of x_in inputs (file_in.txt) and writing to file (file_out.txt) of some decimated outputs. In general, the correctness of the implementation can be assessed by comparing the file file_out.txt with the model file_out_ref.txt. All text files are located in the directory C:\Esercitazione2\files.
When it is necessary to use the same testbench for different types of simulations (i.e. functional, post-synthesis, post-P & R) it is necessary to first instantiate the correct description (architecture) for the block.

Part I - RTL description of the block.

i) Starting from the VHDL code already available, which provides *only a partial* implementation of the block, and with reference to the system specifications, design the driving logic of the data-path control signals (identifiable with the ctrlpath FSM outputs). Also write the code relating to the implementation of registers x_odd and x_even, bearing in mind that their reset value is 0x00 and that the same value must be loaded synchronously and with maximum priority even in correspondence of a 0 level of the system enable signal.
ii) Compile the VHDL code obtained in this way and verify its correctness by functional simulation with the testbench provided (using Modelsim).
iii) Complete the testbench provided also providing a verification of the behavior of the system against the following anomalous driving condition: during set-up (loading of the coefficients) the set_a signal goes to 0 before the unit asserts the ready signal . Using the code-coverage verification tool integrated in Modelsim, evaluate the goodness of the testbench.
iv) Document what was done in points (i) - (iii) (final report at the time of the exam), highlighting any project choices, their philosophy of realization in VHDL, possible corrections made on the code against the functional compilation/simulation, the meaning of the results of the code test -coverage verification.

The environment was prepared in order to automate the simulation processes as much as possible by connecting to the Modelsim tool. In particular, to perform the functional simulation, simply select the testbench in the project tree and launch *Simulate Behavioral Model* . You will launch Modelsim which will execute the commands contained in the script command_behavioral.do. The settings are also such as to enable the display of code-coverage information since the start of Modelsim; for clarity, however, it may be initially convenient to hide them using the options in the Tool-> Code Coverage menu.