FYS4220/9220

Quartus II setup and use for the Modelsim–Altera simulator

By Jan Kenneth Bekkeng, UIO

Simulations using the ModelSim-Altera Software

You can perform simulation of Verilog HDL or VHDL designs with the ModelSim-Altera software at three levels: functional, post-synthesis, and gate-level.

Performing Functional Simulation

Functional simulation verifies code syntax and design functionality.

Performing Post-Synthesis Simulation

You can perform post-synthesis simulation to verify that design functionality is preserved after synthesis.

Performing Gate-Level Timing Simulation

Gate-level timing simulation is an important step in ensuring that the device functionality is correct and meets all timing requirements following place and route.

Modelsim-Altera Simulator Setup in Quartus II

This setup makes use of the NativeLink feature in Quartus II. The Quartus II NativeLink feature eases the tasks of setting up and running a simulation, enables you to launch third-party simulators to perform simulations from within the Quartus II software, and automates the compilation and simulation of testbenches.

Setting Up the EDA Simulator Execution Path

To run an EDA simulator (e.g. Modelsim-Altera) automatically from the Quartus II software using the NativeLink feature, specify the path to your simulation tool by performing the following steps:

1. On the Tools menu, click **Options**. The **Options** dialog box appears.
2. In the **Category** list, select **EDA Tool Options**.
3. Double-click the entry under **Location of executable** beside the name of your EDA Tool Modelsim-Altera .
4. Type the path or browse to the directory containing the executables of the Modelsim-Altera simulator.
5. Click **OK**.

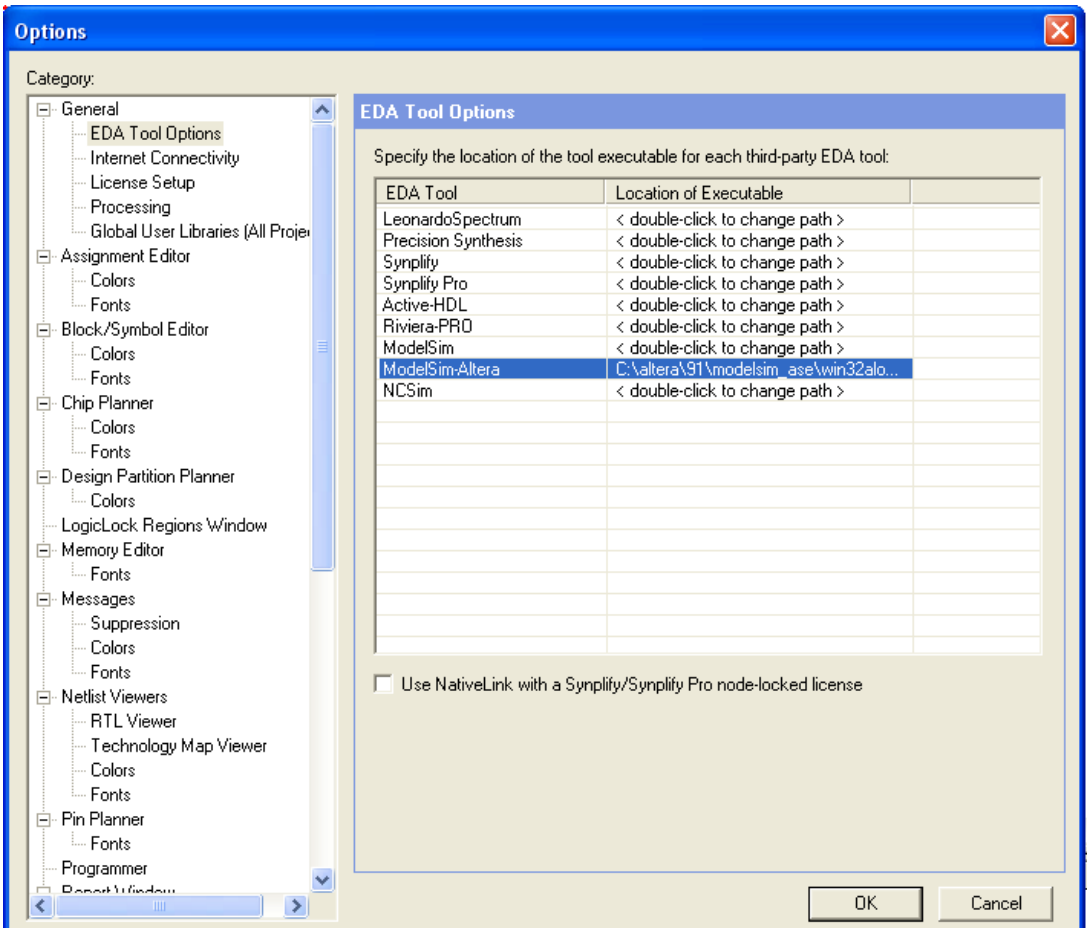

Figure 1

Configuring EDA tool and NativeLink Settings

To configure NativeLink settings, follow these steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Simulation**. The **Simulation** page appears.
3. In the **Tool name** list, select Modelsim-Altera.
4. For gate-level simulation, if you want to run simulation in Modelsim automatically after Quartus II full compilation, turn on ***Run gate-level simulation automatically after compilation***.

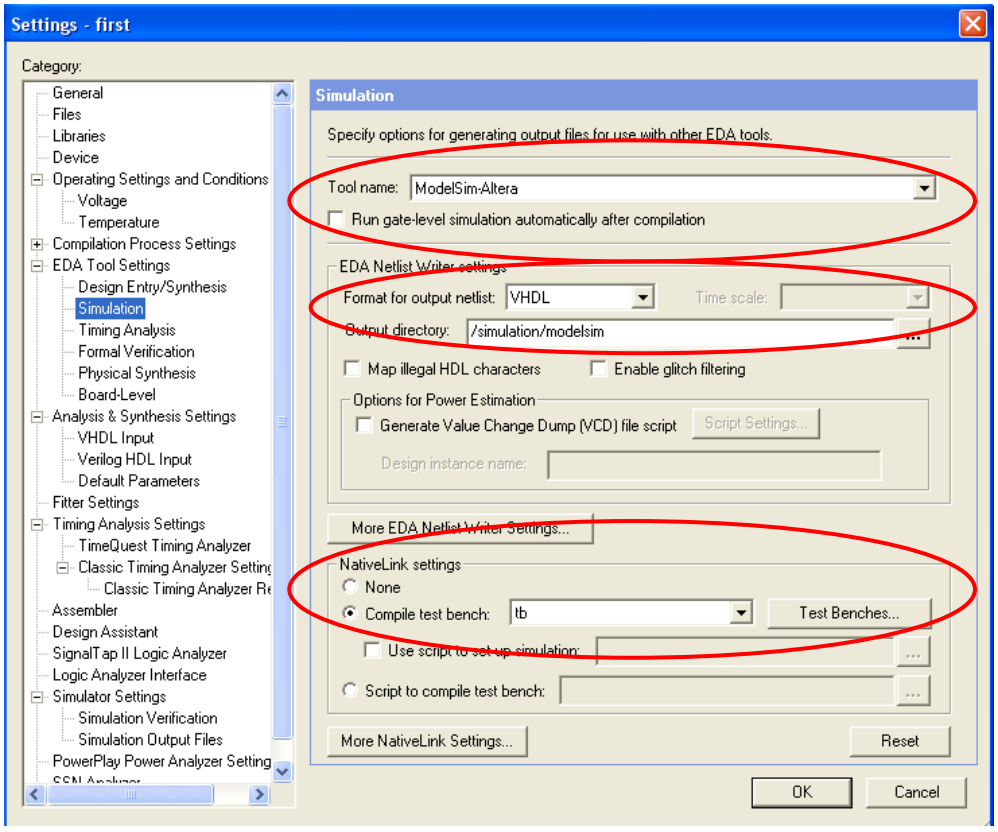

Figure 2

5. ***Format output netlist*** should be VHDL
6. ***Output directory*** can be e.g. /simulation/modelsim. Then these new directories are created under the working directory, and the different Modelsim output files (.vho, .sdo) are placed here.
7. Enter the information about the testbench file under **NativeLink settings**.
    a. Select ***Compile test bench***
    b. Click ***Test Benches***. The ***Test Benches*** dialog box appears.
    c. Click ***New***. The **New Test Bench Settings** dialog box appears.

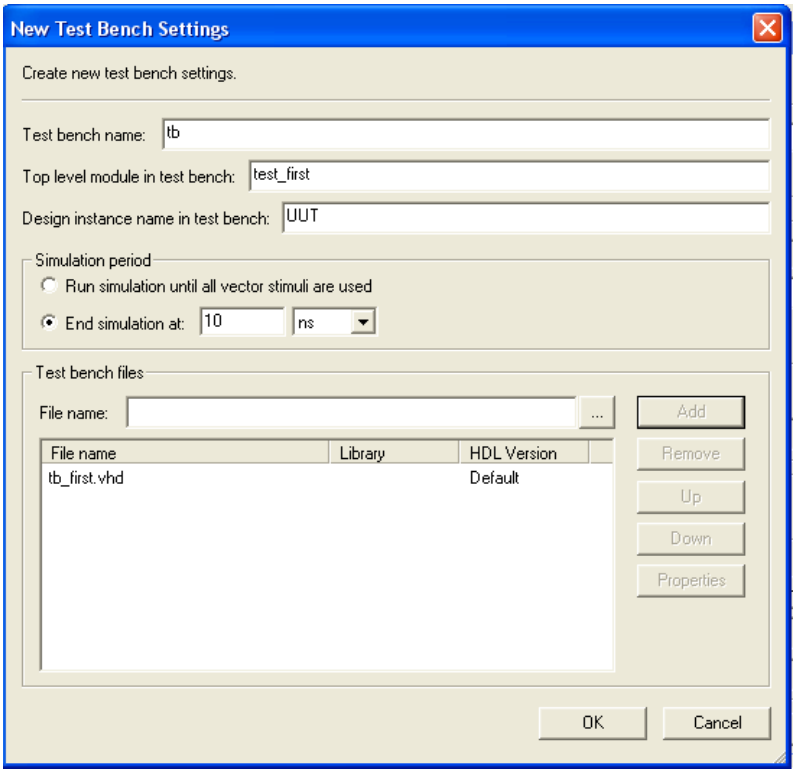

Figure 3

d. The ***Test bench name*** can be any suitable name by your choice. This name will later appear in the **Compile test bench** list.
e. The ***Top level module in test bench*** must be the name of the entity of your testbench file. (For a Quartus II-generated VHDL testbench from a file, e.g. with the name first.vhd, type *first_vhd_vec_tst*).
f. In the ***Design instance name in test bench*** box, type in the label used in front of the port mapping of the component under test in the testbench, in this example this label is UUT (short for Unit Under Test) . Note: for a Quartus II-generated VHDL testbench, type **i1**).

```
--Instantiating of  "Unit Under Test"
UUT : FIRST
port map (
.....
 );
```

g. Select ***End simulation at*** to a suitable simulation time
h. Under **Test bench files**, *browse* and ***Add*** (all of) your testbench files in the ***File name*** box.
i. Select your testbench setup from the **Compile test bench** list
j. Click ***OK***

Running RTL Functional Simulation

1. On the Processing menu, point to **Start** and click **Start Analysis & Elaboration** to perform an Analysis and Elaboration. This command collects all your file name information and builds your design hierarchy in preparation for simulation.

2. On the Tools menu, point to **Run EDA Simulation Tool** and click **EDA RTL Simulation** to automatically run the EDA simulator, compile all necessary design files, and complete a simulation.

Running Gate-Level Timing Simulation

1. On the Processing menu, click **Start Compilation** to perform Quartus II full compilation, including generation of an EDA netlist file.
2. On the Tools menu, point to **Run EDA Simulation Tool** and click **EDA Gate Level Simulation** to automatically run the EDA simulator, compile all necessary design files, and complete a simulation. **(** If you have turned on **Run gate-level simulation automatically after compilation** while configuring NativeLink settings, you can skip this step).

NB! Remember to close the Modelsim-Altera program, if started previously, before running functional or timing EDA simulations from Quartus II.

Generating a Test Bench Template from Quartus II

- If you have not already done so, open an existing project
- If you have not already done so, perform a full compilation
- Select Modelsim-Altera as the EDA simulator tool (if not already done); see Figure 4
- Select an output directory (if not already done);); see Figure 4
- Create the testbench from ***Processing – Start – Start Test Bench Template Writer*** (see Figure 4), this gives a *.vht file, see Figure 5.

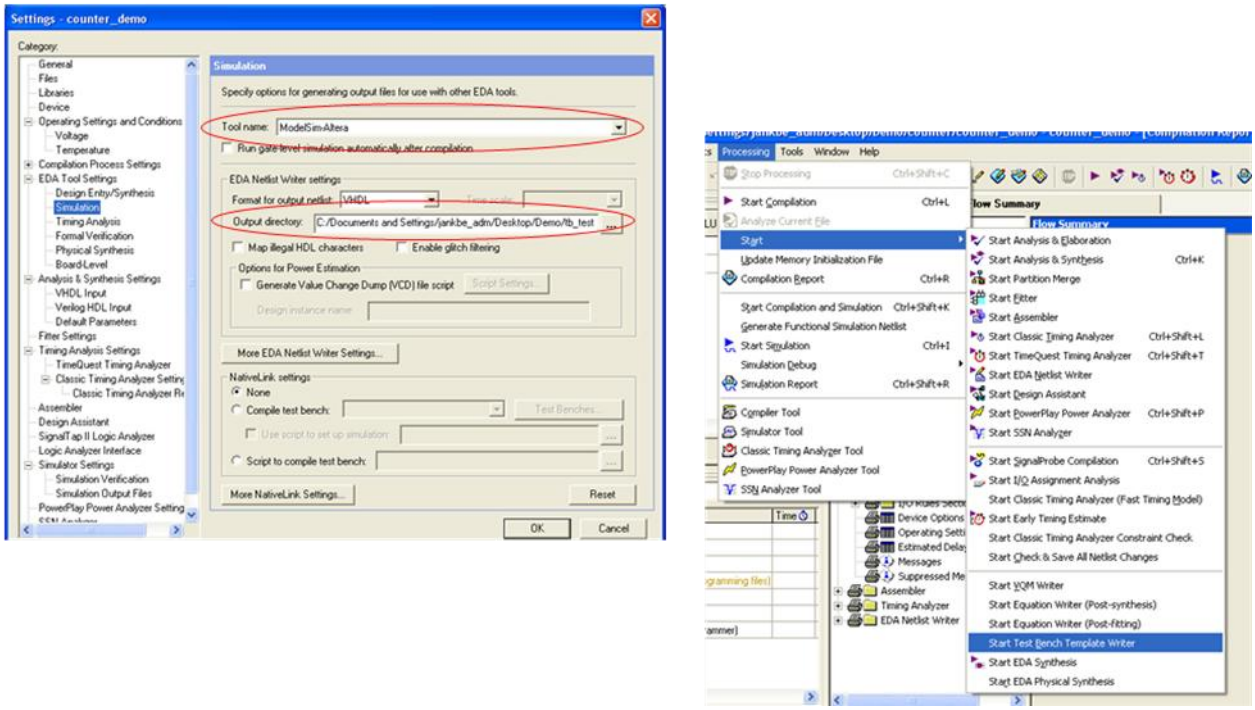

Figure 4

- Rename the file to .vhd (use ***save as***), and add ”tb_” in front to show that this is a testbench file
- E.g. **counter.vht** is saved as **tb_counter.vhd**

```
27  LIBRARY ieee;
28  USE ieee.std_logic_1164.all;
29
30  ENTITY counter_demo_vhd_tst IS
31  END counter_demo_vhd_tst;
32  ARCHITECTURE counter_demo_arch OF counter_demo_vhd_tst IS
33  -- constants
34  -- signals
35  SIGNAL clk_in : STD_LOGIC;
36  SIGNAL clk_out : STD_LOGIC;
37  SIGNAL cnt : STD_LOGIC_VECTOR(3 DOWNTO 0);
38  COMPONENT counter_demo
39      PORT (
40      clk_in : IN STD_LOGIC;
41      clk_out : OUT STD_LOGIC;
42      cnt : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0)
43      );
44  END COMPONENT;
45  BEGIN
46      i1 : counter_demo
47      PORT MAP (
48  -- list connections between master ports and signals
49      clk_in => clk_in,
50      clk_out => clk_out,
51      cnt => cnt
52      );
53  init : PROCESS
54  -- variable declarations
55  BEGIN
56          -- code that executes only once
57  WAIT;
58  END PROCESS init;
59  always : PROCESS
60  -- optional sensitivity list
61  -- (        )
62  -- variable declarations
63  BEGIN
64          -- code executes for every event on sensitivity list
65  WAIT;
66  END PROCESS always;
67  END counter_demo_arch;
```

Figure 5