Simulation of a VHDL code for a pipelined implementation of CORDIC

1. Copy the VHDL file Cordic.Vhd for CORDIC algorithm verification from S:\TN\E\027_Digital_Kommunikationselektronik\CORDIC VHDL code\ to your own directory.
2. Create a new FPGA project by clicking on **File » New » Project » FPGA Project** and rename the new project file by clicking on **File » Save Project As**.
3. Add the VHDL file Cordic.Vhd to the project by right clicking the project file name in the **Projects** panel and selecting **Add Existing to Project …**.
4. Open Cordic.Vhd by double clicking the file name Cordic.Vhd.
5. Click on **Design » Create VHDL Testbench**. A testbench Test_cordic.VHDTST is created and opened as follows.

```
-------------------------------------------------------------
-- VHDL Testbench for cordic
-- 2006 10 23 11 57 33
-- Created by "EditVHDL"
-- "Copyright (c) 2002 Altium Limited"
-------------------------------------------------------------

Library IEEE;
Use     IEEE.std_logic_1164.all;
Use     IEEE.std_logic_textio.all;
Use     STD.textio.all;
-------------------------------------------------------------

-------------------------------------------------------------
entity Testcordic is
end Testcordic;
-------------------------------------------------------------

-------------------------------------------------------------
architecture stimulus of Testcordic is
    file RESULTS: TEXT open WRITE_MODE is "results.txt";
    procedure WRITE_RESULTS(
        angle: std_logic_vector(7 downto 0);
        clk: std_logic;
        datax: std_logic_vector(11 downto 0);
        datay: std_logic_vector(11 downto 0);
        res: std_logic;
        x_n: std_logic_vector(11 downto 0);
        y_n: std_logic_vector(11 downto 0)
    ) is
        variable l_out : line;
    begin
        write(l_out, now, right, 15);
        write(l_out, angle, right, 9);
        write(l_out, clk, right, 2);
        write(l_out, datax, right, 13);
        write(l_out, datay, right, 13);
        write(l_out, res, right, 2);
        write(l_out, x_n, right, 13);
        write(l_out, y_n, right, 13);
        writeline(RESULTS, l_out);
    end procedure;

    component cordic
```

```
        port (
            angle: in std_logic_vector(7 downto 0);
            clk: in std_logic;
            datax: in std_logic_vector(11 downto 0);
            datay: in std_logic_vector(11 downto 0);
            res: in std_logic;
            x_n: out std_logic_vector(11 downto 0);
            y_n: out std_logic_vector(11 downto 0)
        );
    end component;

    signal angle: std_logic_vector(7 downto 0);
    signal clk: std_logic;
    signal datax: std_logic_vector(11 downto 0);
    signal datay: std_logic_vector(11 downto 0);
    signal res: std_logic;
    signal x_n: std_logic_vector(11 downto 0);
    signal y_n: std_logic_vector(11 downto 0);

begin
    DUT:cordic port map (
        angle => angle,
        clk => clk,
        datax => datax,
        datay => datay,
        res => res,
        x_n => x_n,
        y_n => y_n
    );

    STIMULUS0:process
    begin
        -- insert stimulus here
        wait;
    end process;

    WRITE_RESULTS(
        angle,
        clk,
        datax,
        datay,
        res,
        x_n,
        y_n
    );

end architecture;
-------------------------------------------------------------------

-------------------------------------------------------------------
```

6. In the testbench, insert the some statements before the STIMULUS0:**process** and some statements in the process as follows:

```
   clk <= not clk after 1 ns;
    res <= '0', '1' after 2 ns;

    STIMULUS0:process
    begin
        -- insert stimulus here
        angle <= B"00100000";
```

```
        datax <= B"010000000000";
        datay <= B"000000000000";
        wait for 2 ns;
        angle <= B"00110000";
        datax <= B"011000000000";
        datay <= B"000000000000";
        wait for 2 ns;
       end process;
```

The statement "`clk <= not clk after 1 ns`;" defines the behavior of a simulated clock signal. The time 1 ns is one half of the time period for the clock. The time 1 ns is chosen for convenience and is not a realistic value for the FPGA. The frequency of the clock may be changed by changing the time to other values. Notice that this statement can only be used for simulation. The statement "`res <= '0', '1' after 2 ns`;" sets the res signal to '0' at the beginning and sets it to '1' after 2 ns. In the process STIMULUS0, the values of input signals are changed after 2 ns. You can add other input signal values in the process.

7. Add an initial value to the signal clk as follows:

```
signal clk: std_logic := '0';
```

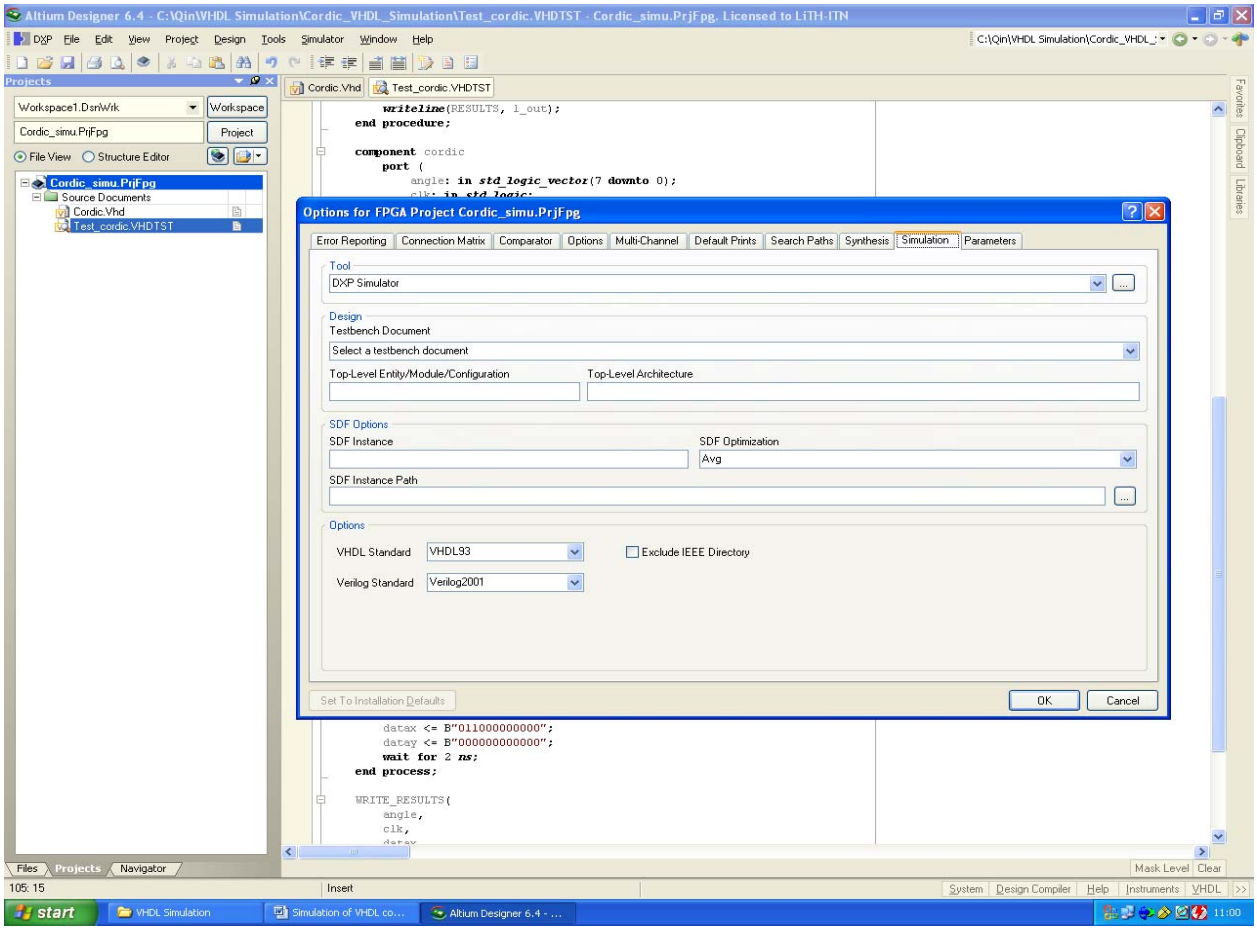
   After the changes, the testbench looks like the following:

```
--------------------------------------------------------------
-- VHDL Testbench for cordic
-- 2006 10 23 11 57 33
-- Created by "EditVHDL"
-- "Copyright (c) 2002 Altium Limited"
--------------------------------------------------------------

Library IEEE;
Use     IEEE.std_logic_1164.all;
Use     IEEE.std_logic_textio.all;
Use     STD.textio.all;
--------------------------------------------------------------

--------------------------------------------------------------
entity Testcordic is
end Testcordic;
--------------------------------------------------------------

--------------------------------------------------------------
architecture stimulus of Testcordic is
    file RESULTS: TEXT open WRITE_MODE is "results.txt";
    procedure WRITE_RESULTS(
        angle: std_logic_vector(7 downto 0);
        clk: std_logic;
        datax: std_logic_vector(11 downto 0);
        datay: std_logic_vector(11 downto 0);
        res: std_logic;
        x_n: std_logic_vector(11 downto 0);
        y_n: std_logic_vector(11 downto 0)
    ) is
        variable l_out : line;
    begin
        write(l_out, now, right, 15);
        write(l_out, angle, right, 9);
```

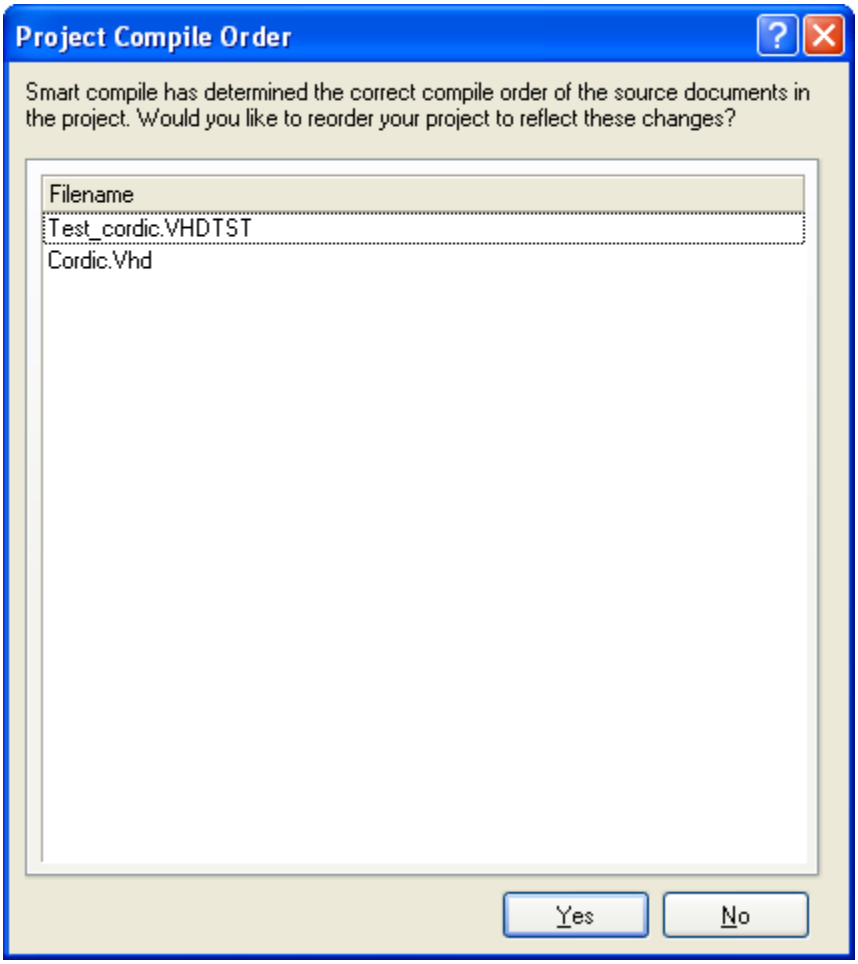
```
        write(l_out, clk, right, 2);
        write(l_out, datax, right, 13);
        write(l_out, datay, right, 13);
        write(l_out, res, right, 2);
        write(l_out, x_n, right, 13);
        write(l_out, y_n, right, 13);
        writeline(RESULTS, l_out);
    end procedure;

    component cordic
        port (
            angle: in std_logic_vector(7 downto 0);
            clk: in std_logic;
            datax: in std_logic_vector(11 downto 0);
            datay: in std_logic_vector(11 downto 0);
            res: in std_logic;
            x_n: out std_logic_vector(11 downto 0);
            y_n: out std_logic_vector(11 downto 0)
        );
    end component;

    signal angle: std_logic_vector(7 downto 0);
    signal clk: std_logic := '0';
    signal datax: std_logic_vector(11 downto 0);
    signal datay: std_logic_vector(11 downto 0);
    signal res: std_logic;
    signal x_n: std_logic_vector(11 downto 0);
    signal y_n: std_logic_vector(11 downto 0);

begin
    DUT:cordic port map (
        angle => angle,
        clk => clk,
        datax => datax,
        datay => datay,
        res => res,
        x_n => x_n,
        y_n => y_n
    );
     clk <= not clk after 1 ns;
     res <= '0', '1' after 2 ns;
    STIMULUS0:process
    begin
        -- insert stimulus here
        angle <= B"00100000";
        datax <= B"010000000000";
        datay <= B"000000000000";
        wait for 2 ns;
        angle <= B"00110000";
        datax <= B"011000000000";
        datay <= B"000000000000";
        wait for 2 ns;
    end process;

    WRITE_RESULTS(
        angle,
        clk,
        datax,
        datay,
        res,
        x_n,
        y_n
    );
```

```
end architecture;
-------------------------------------------------------------------

-------------------------------------------------------------------
```

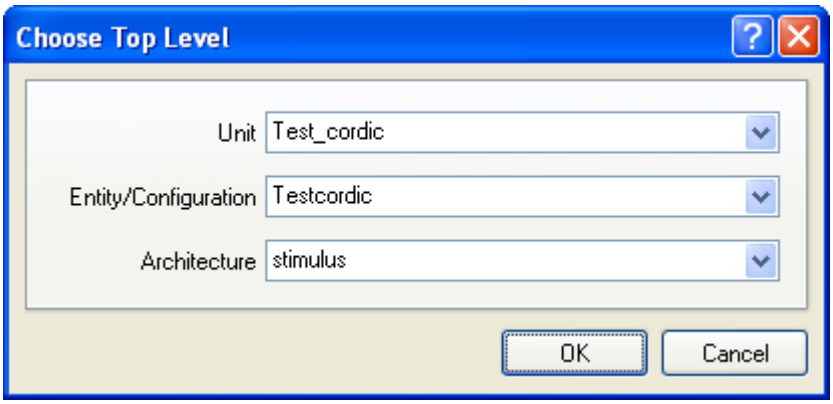
8. Select the simulation tool and the testbench document by right clicking the project file name in the **Projects** panel and selecting **Simulation** tab from within the **Project Options** dialog.



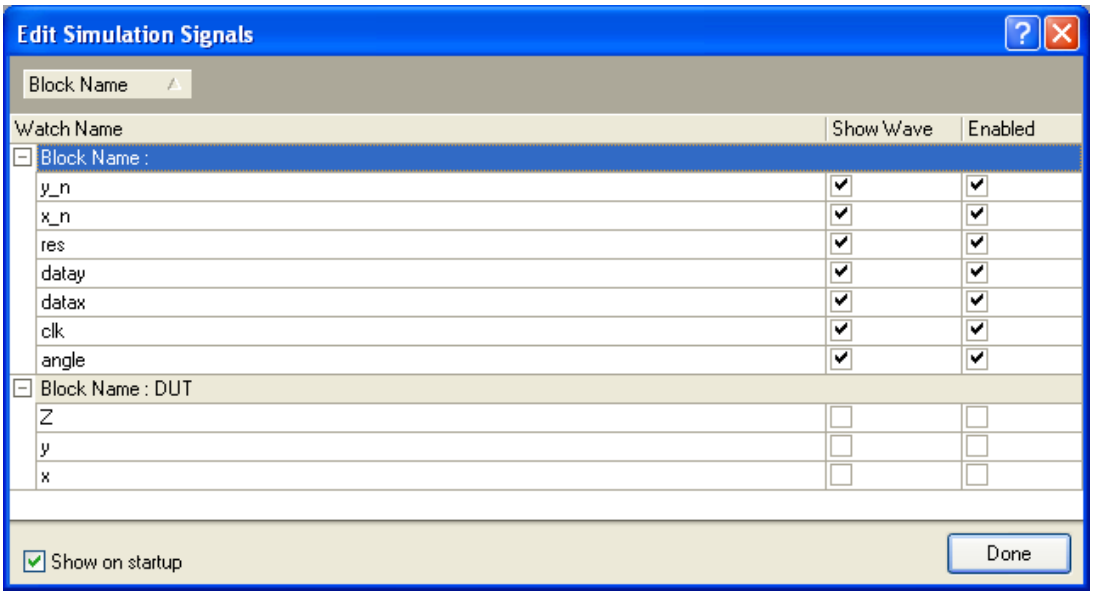
9. Select the simulation tool **DXP Simulator** from the drop-down list for **Tool**.
10. Select the testbench document "Test_cordic.VHDTST" from the drop-down list for **Testbench Document** and click the **OK** button.
11. Initiate a simulation session by selecting **Simulator » Simulate** from the menu.
12. When you first run a simulation from a testbench, the following window will be shown. Whilst performing this process, you may see an error appearing in the **Messages** panel with the message: "Unbounded instance DUT of component Cordic". Do not be concerned as this is normal when you first run a simulation.



13. If the correct compile order is shown as in the above window, click the **No** button. The following window will be shown.

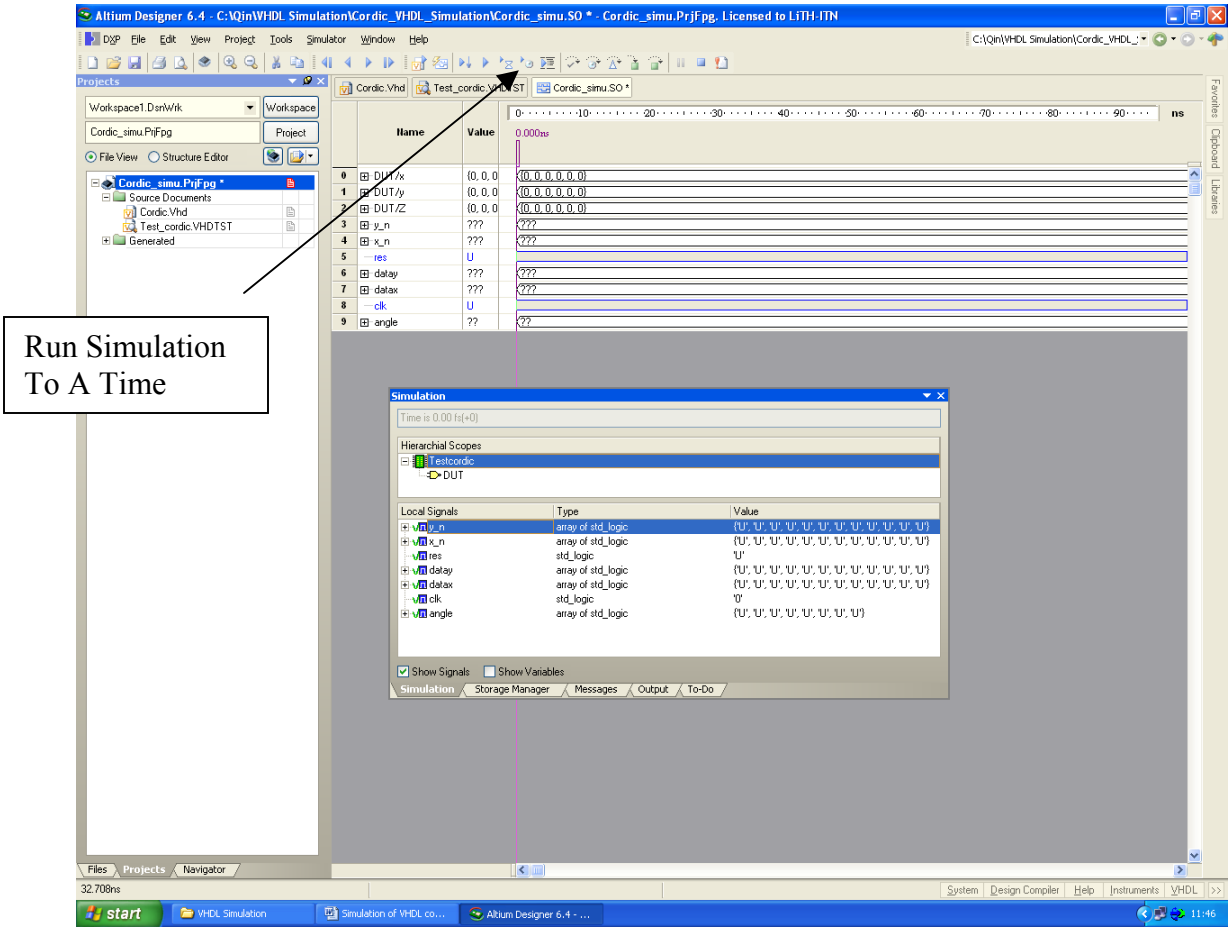


14. Click the **OK** button. The following window will be shown.

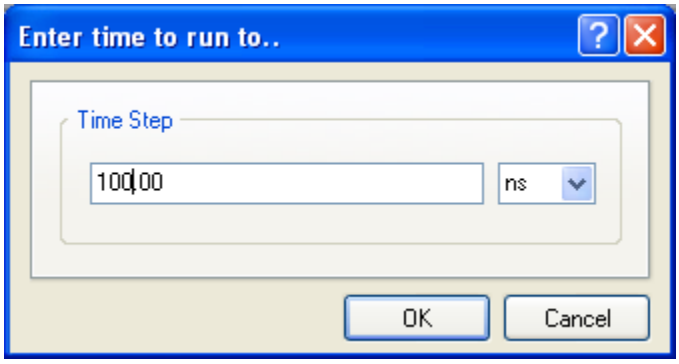


15. You can add signals to the display of the simulation by setting a tick for the signals in **Show Wave**. The signals should also be **Enabled**. Click on the **Done** button. If you need to change the signals for the display later, this window can be accessed by clicking **Simulator » Signals**.

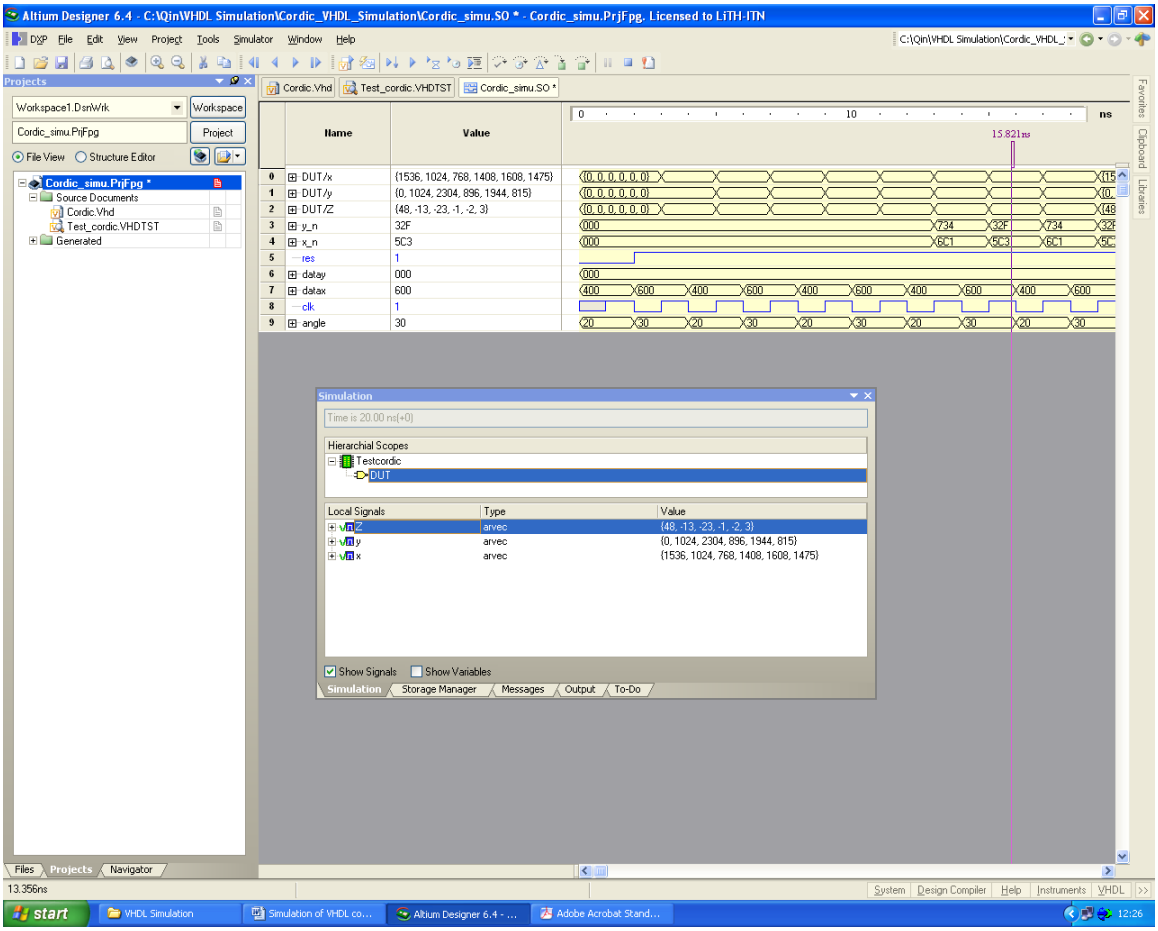
16. After the above step, the following window is shown. The following text explains details of some functions.
    - The “plus” icon next to the bus name indicates a bus signal. Clicking on this icon will expand the bus into its individual signals for closer inspection.
    - The time cursor (indicated by the purple vertical bar) can be dragged along the time axis via the mouse. The current position of the cursor is provided in the time bar across the top of the display. The values of signals under the time cursor will be shown in the column **Value**.
    - Zooming in or out is achieved by pressing the Page Up or Page Down keys respectively.
    - The display format of the individual signals can be altered via the menu item **Tools » Format and Radix**.



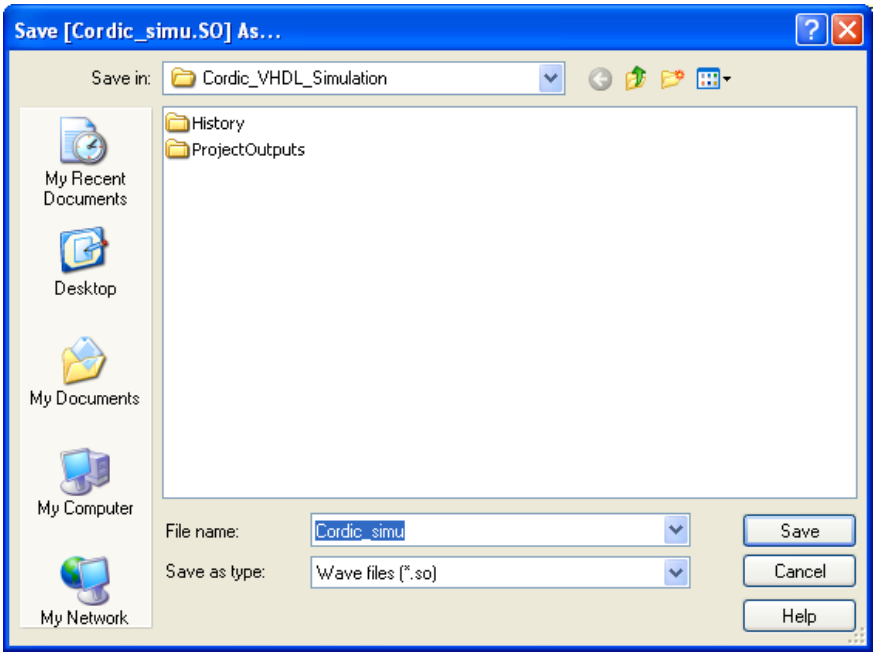
17. Run the simulation to a time by clicking the button **Run Simulation To A Time**. The following window will be shown. You may change the time to 20 ns and click the **OK** button.



18. After the simulation, you can change the zoom and the time cursor to check the signal values.



19. Click the Save button in the menu, the following dialog window will be shown. You can save the simulation result in a wave file.



You can read the other simulation commands from Chapter 7 of the document "FPGA Design Training Module" in the file Training Module 5 FPGA Design.pdf in the directory S:\TN\E\027_Digital_Kommunikationselektronik\Altium Manuals and Tutorials\.

20. You can reset the simulator by clicking **Simulator » Reset** from the menu. Notice that you may only reset the simulator once. If it does not work after resetting the simulator. You must click **Simulator » End** to terminate the simulation. Then you should click **Simulator » Simulate** for further simulations.

21. You can try to change the statements in the stimulus process of the testbench and obtain other simulation results.

Notice that one can also simulate the VHDL code in your project previously created in Lab2. After opening your project, you should open a schematic document and select **Tools » Convert » Create VHDL Testbench** from the menu.

Reference:

"FPGA Design Training Module" in the file Training Module 5 FPGA Design.pdf in the directory S:\TN\E\027_Digital_Kommunikationselektronik\Altium Manuals and Tutorials\.