Tutorial on Simulation using Aldec Active-HDL Ver 1.0

by Shashi Karanam
Introduction

Active-HDL is an integrated environment designed for development of VHDL designs. The core of the system is a VHDL simulator. Along with debugging and design entry tools, it makes up a complete system that allows you to write, debug and simulate VHDL code. Based on the concept of a design, Active-HDL allows you to organize your VHDL resources into a convenient and clear structure.

Students can use Active-HDL to perform following tasks:
• development of the VHDL based designs,
• functional simulation of their code,
• functional simulation of the synthesized code,
• timing simulation of the hardware implementation.

Objective

This tutorial helps you to
• Create a new design or add .vhd files to your design
• Compile and debug your design
• Perform simulation

Note: This tutorial does not explain the synthesis or implementation steps.
**Start-up**

1. Double click on the Active HDL icon on the desktop; it comes up with a getting started window.

   ![Getting Started window]

   a. Select create new workspace and click OK, creates new workspace in your directory.
   b. Selecting open existing workspace gives you the option to choose from your previous Workspace’s.

2. Type the name and select location of your desired workspace and click OK. If you’re using school’s lab, it is recommended to work in your local’s drive (D drive), as your thumb drive and your network drive (K drive) are slow. For now, use “lab” as your workspace.

   ![New Workspace window]
3. Select Create an Empty Design with Design Flow, and click Next.

4. Choose block diagram configuration as **Default HDL language** and Default HDL language as **VHDL**; Leave the Target Technology blanks as not defined; Click **Next**.
5. Type the design name you want to create and Click **Next**. For this tutorial, use “hm1” as your design name.

6. Design name and design directory will be displayed proceed further by clicking **Finish**.
7. To the left workspace and new design are displayed.

8. To add existing VHDL files to your design GO TO 9A.
   To create a new VHDL source file using the source code wizard GO TO 9B.

9A. To add existing files to your design, select Add New File > Add Files to Design.
    Then, GO TO 13.
9B. To create a new VHDL source file using the source code wizard right click on Add New File > New > VHDL source.

New VHDL source can also be added by following up File > New > VHDL source.
10. Click **Next** on the new source file wizard.
11. Type the source file name you want to create; you can choose to give a different name for entity and architecture, otherwise the default name for entity and architecture will be your source file name. Click **Next**. “or3” is just a placeholder for any source file you want to create. But for now type in “or3” to go ahead with this tutorial.

![New Source File Wizard - Name]

12. Select **New** in the New Source File Wizard-Ports, start declaring your input and output ports by naming them, choosing the port direction, choose the type by indicating the array indexes. Click **Finish**. Once done, supplement any missing portions of the entity architecture, based on the example of the “or3” gate given at the end of this tutorial.

![New Source File Wizard - Ports]
13. If you have multiple designs in the same workspace you need to set an active design and this can be done by a right click on your design and select **Set as Active Design**.
Main Window Toolbars
Language Assistant helps you with the syntax of the VHDL data types.

Compiling the Design

Once the design is ready, compile it either by clicking on the toolbar as shown or hit F11 or choose Design > Compile.

Console window

Console window appears at the bottom of the page with a success message if design is free of errors.
**Generating Testbench**

**NOTE:** To add your testbench, click **Add New File>Add Files to Design** as in 9A.

Testbench for the design can be generated as **Tools > Generate Testbench**.

Select the Design Unit (Entity and Architecture) for which testbench should be generated and select testbench type as **Single Process**. Proceed by clicking **Next**.
Testbench generator wizard appears with all your input ports under UUT ports. Proceed by clicking Next.

Click Next as the tool generates the names for all the design units. You can change them (Not Recommended).
Click **Finish** to proceed, the window below shows the files generated by the Testbench. The source code for the testbench can be obtained from the below section.

![Test Bench Generator Wizard](image)

**NOTE**: Add your test conditions to the Testbench, Hit **F11** to compile.
Getting Started with the Simulation

Before simulate, select Top level unit as your testbench as shown below in the design browser.

NOTE: All files must be compiled before you select Top level unit.

Initialize simulation by selecting Simulation > Initialize Simulation.

Click Structures (circled below) to see all the ports, select them and add to the waveform as shown below.
Hit **F5** to run or select **Simulation > Run For**. You should never use **Run** to simulate the testbench as this would run indefinitely.

Wave Window Toolbar

Please explore by yourself various options of the Wave Window Toolbar.
Creating/Editing VHDL Files

or3.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity or3 is
port(
    A : in STD_LOGIC;
    B : in STD_LOGIC;
    C : in STD_LOGIC;
    Z : out STD_LOGIC
);
end or3;

architecture behave of or3 is
begin
    Z <= A or B or C;
end behave;

or3_tb.vhd

library ieee;
use ieee.std_logic_1164.all;
-- Add your library and packages declaration here ...
entity or3_tb is
end or3_tb;
architecture TB_ARCHITECTURE of or3_tb is
-- Component declaration of the tested unit _component or3
port(
    A : in std_logic;
    B : in std_logic;
    C : in std_logic;
    Z : out std_logic
);
end component;

-- Stimulus signals - signals mapped to the input and inout ports of tested entity
signal A : std_logic;
signal B : std_logic;
signal C : std_logic;
-- Observed signals - signals mapped to the output ports of tested entity
signal Z : std_logic;
-- Add your code here ...
begin
    -- Unit Under Test port map
    UUT : or3
port map (  
  A => A,  
  B => B,  
  C => C,  
  Z => Z  
);  
process  
begin  
  ----Test case 1----  
  A<='0';  
  B<='1';  
  c<= '0';  
  wait for 20ns;  
  ----Test case 2----  
  A<='1';  
  B<='1';  
  c<='0';  
  wait for 20ns;  
  ----Test case 3----  
  A<='0';  
  B<='1';  
  c<='1';  
  wait for 20ns;  
  ----Test case 4 boundary condition----  
  A<='0';  
  B<='0';  
  c<='0';  
  wait for 20ns;  
  ----Test case 5 boundary condition----  
  A<='1';  
  B<='1';  
  c<='1';  
  wait for 20ns;  
end process;  
-- Add your stimulus here ...  
end TB_ARCHITECTURE;  
configuration TESTBENCH_FOR_or3 of or3_tb is  
  for TB_ARCHITECTURE  
    for UUT : or3  
      use entity work.or3(behave);  
    end for;  
end for;  
end TESTBENCH_FOR_or3;