Tutorial on Simulation using Aldec Active-HDL Ver 2.0

by Shashi Karanam, Kishore Kumar Surapathri
**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. Start >> VLSI Tools >> Active-HDL 8.2
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 “labdemo” as your workspace.

3. Select Create an Empty Design, 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 “andgate” 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. You can add already existing VHDL source files. To create a new VHDL source file using the source code wizard **GO TO 9**
9. To create a new VHDL source file using the source code wizard right click on **Add New File > New > VHDL source**.

10. Click **Next** on the new source file wizard.
This wizard will create a source file with initial VHDL code using the design specifications you will enter in the following wizard dialogs.

The generated source file will contain the entity declaration, port declarations and empty architecture body.

- Add the generated file to the design

Clear this check box if you do not want to add the file generated by the wizard to the current design.
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. For now type in “andgate” to go ahead with this tutorial.

![New Source File Wizard - Name](image)

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 “andgate” gate given at the end of this tutorial.
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.

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.
Select the design unit for which you want to generate a testbench. The wizard will generate appropriate source files and a macro file for the testbench.

Entity:
andgate

Architecture:
arch

Testbench Type:
- Single Process
- WAVES Based
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).
Enter the testbench specification.

Type the name of the testbench entity:
andgate_tb

Type the name of the testbench architecture:
TB_ARCHITECTURE

Type the name of the testbench source file:
andgate_TB.vhd

Type the name of the folder for testbench files:
TestBench
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.
NOTE: Add your test conditions to the Testbench, Hit **F11** to compile.

```vhdl
library ieee;
use ieee.std_logic_1164.all;

-- Add your library and packages declaration here ... 
entity andgate_th is 
end andgate_th;

architecture TB_ARCHITECTURE of andgate_th is 
-- Component declaration of the tested unit
component andgate
  port(
    a : in STD_LOGIC;
    b : in STD_LOGIC;
    c : out STD_LOGIC);
end component;

-- Stimulus signals - signals mapped to the input and output ports of tested entity
signal a : STD_LOGIC;
signal b : STD_LOGIC;

-- Observed signals - signals mapped to the output ports of tested entity
signal c : STD_LOGIC;

-- Add your code here ...
begin
  -- Unit Under Test port map
  WUT : andgate 
  port map ( 
    a => a, 
    b => b, 
    c => c
  );

  -- Add your stimulus here ...
end TB_ARCHITECTURE;
```
Getting Started with the Simulation

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

```
begin
  -- Unit Under Test port map
  UUT : andgate
    port map (    
      a => a,
      b => b,
      c => c
    );

  -- Add your stimulus here ...

  process
  begin
    a <= '0';
    b <= '0';
    wait for 10 ns;
end process
```

**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.
begin

-- Unit Under Test port map
UUT : andgate
     port map(
       a => a,
       b => b,
       c => c
     );

-- Add your stimulus here ...

process
begin
  a <= '0';
  b <= '0';
  c <= '0';
  a <= '1';
  b <= '1';
  c <= '1';
  a <= '0';
  b <= '0';
  c <= '0';
  a <= '1';
  b <= '1';
  c <= '1';
end process;

end;
Hit F5 to run or select Simulation > Run For. You should never use Run to simulate the testbench as this would run indefinitely.
Please explore by yourself various options of the Wave Window Toolbar.
Creating/Editing VHDL Files

andgate.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity andgate is
  port(
    a : in STD_LOGIC;
    b : in STD_LOGIC;
    c : out STD_LOGIC
  );
end andgate;

architecture arch of andgate is
begin
  -- enter your statements here --
  c <= a and b;
end arch;

-- End of automatically maintained section

andgate_tb.vhd

library ieee;
use ieee.std_logic_1164.all;

-- Add your library and packages declaration here ...
entity andgate_tb is
end andgate_tb;
architecture TB_ARCHITECTURE of andgate_tb is

  -- Component declaration of the tested unit
  
  component andgate
  
  port(
    a : in STD_LOGIC;
    b : in STD_LOGIC;
    c : 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;

  -- Observed signals - signals mapped to the output ports of tested entity

  signal c : STD_LOGIC;

  -- Add your code here ...

begin

  -- Unit Under Test port map

  UUT : andgate
  
  port map ( 
    a => a,
    b => b,
    c => c
  );

  -- Add your stimulus here ...

  process
  
  begin
    a <= '0';
b <= '0';
wait for 10 ns;
a <= '1';
b <= '0';
wait for 10 ns;
a <= '0';
b <= '1';
wait for 10 ns;
a <= '1';
b <= '1';
wait;
end process;
end TB_ARCHITECTURE;

classification TESTBENCH_FOR_andgate of andgate_tb is
  for TB_ARCHITECTURE
    for UUT : andgate
      use entity work.andgate(arch);
    end for;
  end for;
end TESTBENCH_FOR_andgate;