

Simulating a Design Using ModelSim VHDL Compiler and Simulator

Dr. Aleksandar Milenkovic
Electrical and Computer Engineering
The University of Alabama in Huntsville
E-mail: milenka@ece.uah.edu

In this tutorial you will learn to edit, compile, and simulate VHDL models. You will use the ModelSim compiler/simulator from Model Technology to simulate an example of the binary to hexadecimal converter.

To use the ModelSim VHDL compiler/simulator proceed as follows:

With the mouse, first double click on the ModelSim icon that is on the desktop.



Close the “Welcome to the ModelSim” popup window and it will open up the following ModelSimSE command window .

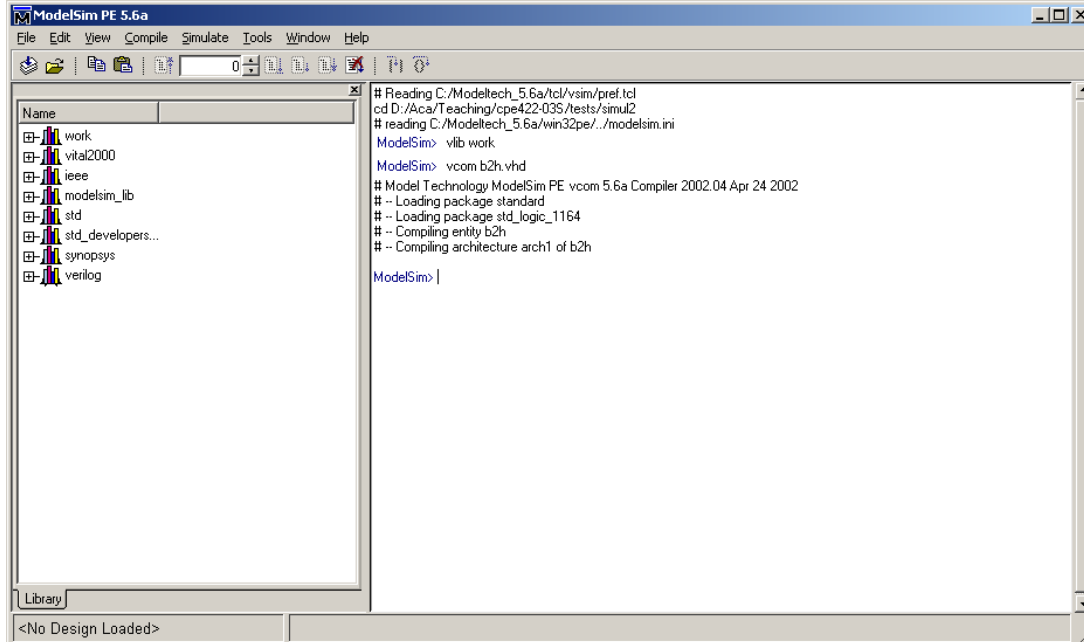


Figure 1. ModelSim command window.

You will then want to change to the folder that is associated with your account and to create a new directory for this design.

```
% cd c:\smith\cpe422 <press enter/return key> # move to your directory here
% mkdir simul2 <press enter/return key> # create directory for this lab
% cd simul2 <press enter/return key> # move to this directory
```

Each time we start a new design we must create a VHDL library that will hold our compiled designs, just as the IEEE library contains the std_logic_1164 package. The library name **work** is commonly used. Remember, *this only has to be done when you first create a new project directory*. From the command line type:

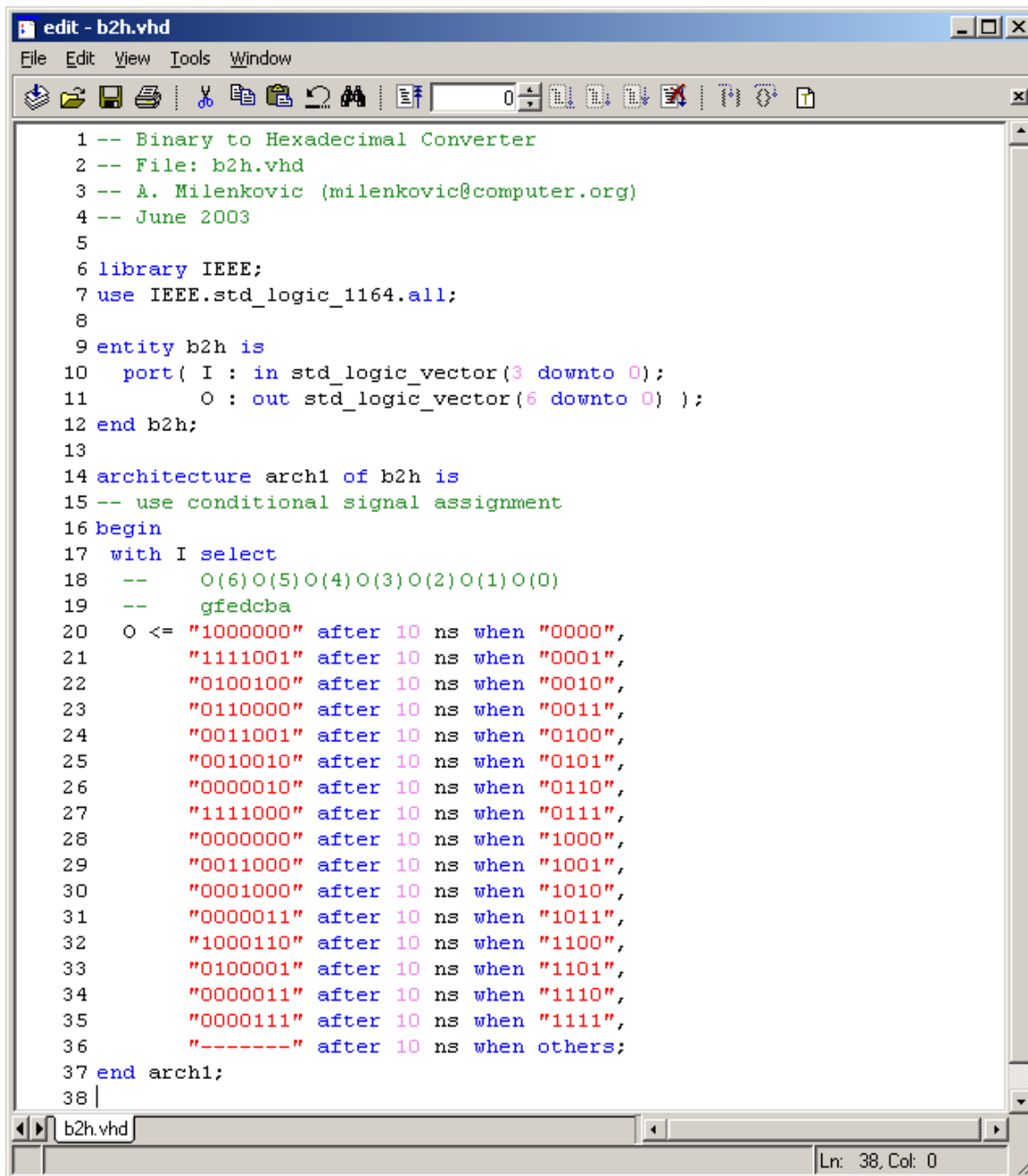
```
% vlib work <press enter/return key>
```

This creates a library called work in the current directory.

Your next step is to type in the following VHDL code for the converter. You may use any text editor or the ModelSim's edit utility. To open the ModelSim editor click on **File->New->Source-VHDL**.

Type in the VHDL code for the binary to hexadecimal converter (see Figure 2).

To save file select the following: **File->Save as** and give the name of the file (e.g., b2h.vhd).

The image shows a screenshot of a text editor window titled "edit - b2h.vhd". The window contains VHDL code for a binary-to-hexadecimal converter. The code includes a header with a title, author, and date, followed by library declarations for IEEE and std_logic_1164. The main entity "b2h" has two ports: "I" (input) and "O" (output), both of type std_logic_vector. The architecture "arch1" uses conditional signal assignment to map 8-bit binary inputs to their corresponding hexadecimal outputs. The mapping is as follows: 0000 to gfedcba, 0001 to gfedcba, 0010 to gfedcba, 0011 to gfedcba, 0100 to gfedcba, 0101 to gfedcba, 0110 to gfedcba, 0111 to gfedcba, 1000 to gfedcba, 1001 to gfedcba, 1010 to gfedcba, 1011 to gfedcba, 1100 to gfedcba, 1101 to gfedcba, 1110 to gfedcba, 1111 to gfedcba, and others to gfedcba. The code is line-numbered from 1 to 38. The status bar at the bottom indicates "Ln: 38, Col: 0".

```
1 -- Binary to Hexadecimal Converter
2 -- File: b2h.vhd
3 -- A. Milenkovic (milenkovic@computer.org)
4 -- June 2003
5
6 library IEEE;
7 use IEEE.std_logic_1164.all;
8
9 entity b2h is
10 port( I : in std_logic_vector(3 downto 0);
11       O : out std_logic_vector(6 downto 0) );
12 end b2h;
13
14 architecture arch1 of b2h is
15 -- use conditional signal assignment
16 begin
17 with I select
18 --   O(6)O(5)O(4)O(3)O(2)O(1)O(0)
19 --   gfedcba
20 O <= "1000000" after 10 ns when "0000",
21      "1111001" after 10 ns when "0001",
22      "0100100" after 10 ns when "0010",
23      "0110000" after 10 ns when "0011",
24      "0011001" after 10 ns when "0100",
25      "0010010" after 10 ns when "0101",
26      "0000010" after 10 ns when "0110",
27      "1111000" after 10 ns when "0111",
28      "0000000" after 10 ns when "1000",
29      "0011000" after 10 ns when "1001",
30      "0001000" after 10 ns when "1010",
31      "0000011" after 10 ns when "1011",
32      "1000110" after 10 ns when "1100",
33      "0100001" after 10 ns when "1101",
34      "0000011" after 10 ns when "1110",
35      "0000111" after 10 ns when "1111",
36      "-----" after 10 ns when others;
37 end arch1;
38
```

Figure 2. B2h VHDL code.

To compile a VHDL module

Execute the following command from the ModelSim command window:

```
% vcom b2h.vhd <press enter/return key>
```

where *vcom* invokes the ModelSim vhd compiler and *b2h.vhd* is the name of your VHDL source file.

If you do not have any errors you will see a report similar to the following:

```
# Model Technology ModelSim PE vcom 5.7d Compiler 2003.05 May 11 2003
# -- Loading package standard
# -- Loading package std_logic_1164
# -- Compiling entity b2h
# -- Compiling architecture arch1 of b2h
```

If there are errors this process should be repeated as necessary by editing and altering the VHDL source file.

To simulate a VHDL module

Once the VHDL model is successfully compiled into the work library, the simulator is invoked to test it.

Start the simulator by selecting **Simulate->Simulate**.

The Simulate window is displayed showing you all the designs that are currently available. Select the b2h entity in the work directory to display all the architectures that have been associated with the entity. You should have a window like the one below. Click on the b2h entity and click the **Load** button.

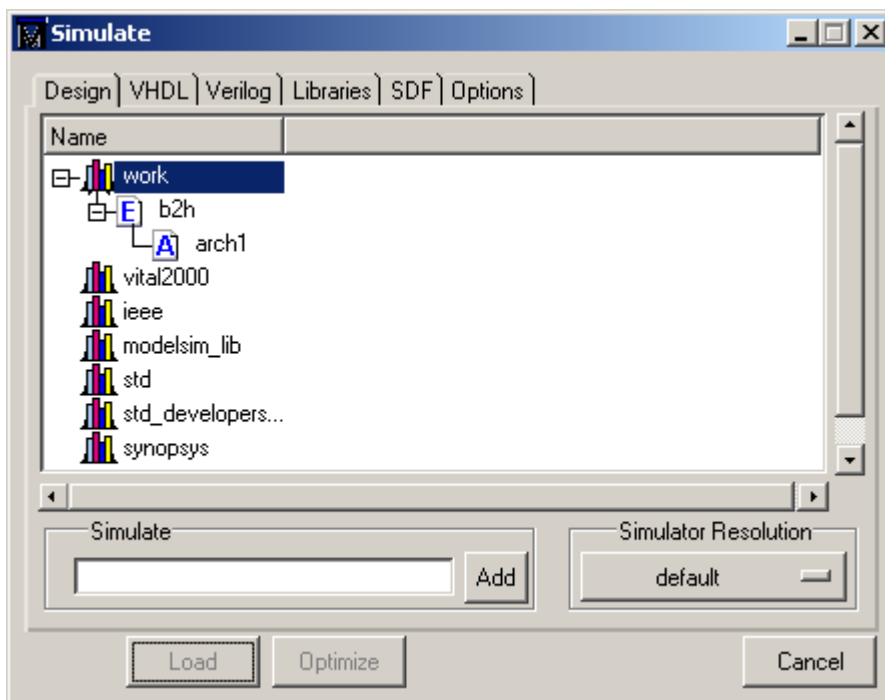


Figure 3. Load design.

Now we need to force the inputs of the design to see if it behaves properly. We will use VHDL testbenches to force inputs on labs and our projects later in the semester. For now we will use the force

commands build into the simulator. ModelSim command files are an easy way to implement a series of commands. The ModelSim command interpreter is actually a Tcl interpreter with ModelSim specific functions added. There are dozens of ModelSim commands but the ones we are interested in are **force**, **run**, and **add wave**. Note that any of the commands in a command file can be interactively entered at the VSIM command prompt in the ModelSim window.

For this tutorial, we will directly type in commands to apply forces to the inputs. Type in the following:

```
% force I "0000" 0, "0001" 20, "0010" 40 # input I is "0000" at 0 ns,  
"0001" at 20 ns, "0010" at 40 ns
```

Although there are many options for viewing simulation state, we will use the signals and the wave view here. From the ModelSim pulldown menu bar, start by selecting **View ->Signals**. The following signal window will appear:

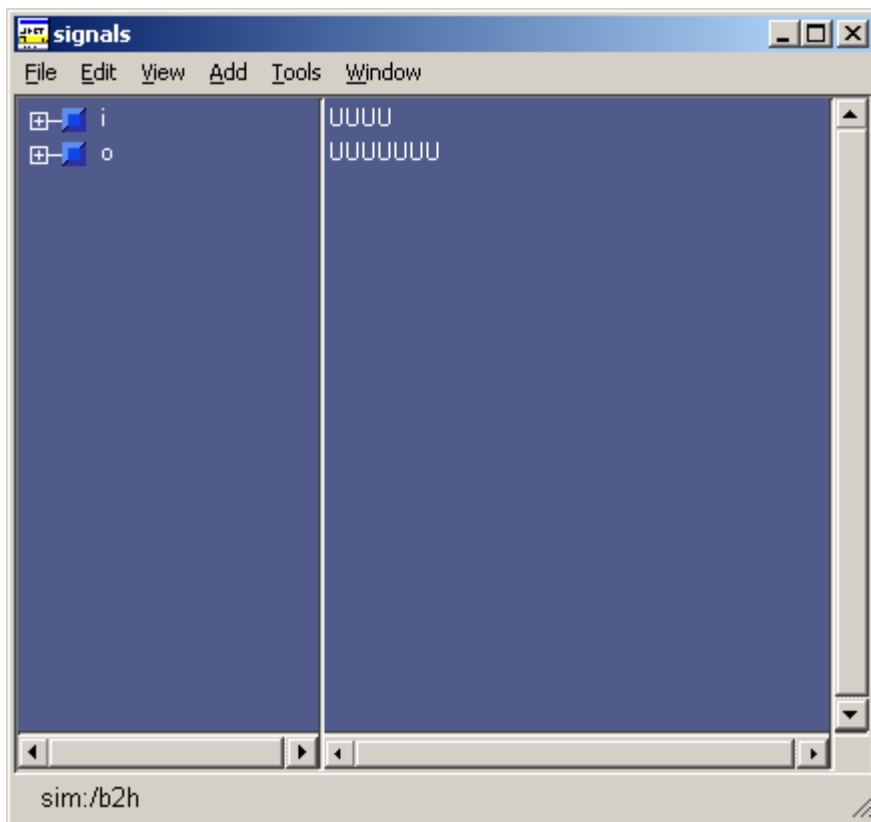


Figure 4. View signals.

To view all the signals in the model select **Add->Wave->Signals in Region** from the signal window. A waveform window will appear containing all of the signals in the design. We could have also use the **add wave** Tcl command as part of a command file. The display is now prepared for our simulation. You may wish to adjust window sizes and positions so that all signals are visible in the Wave window. Once you feel comfortable with these views, you are encouraged to explore others that are also available. The following window will appear.

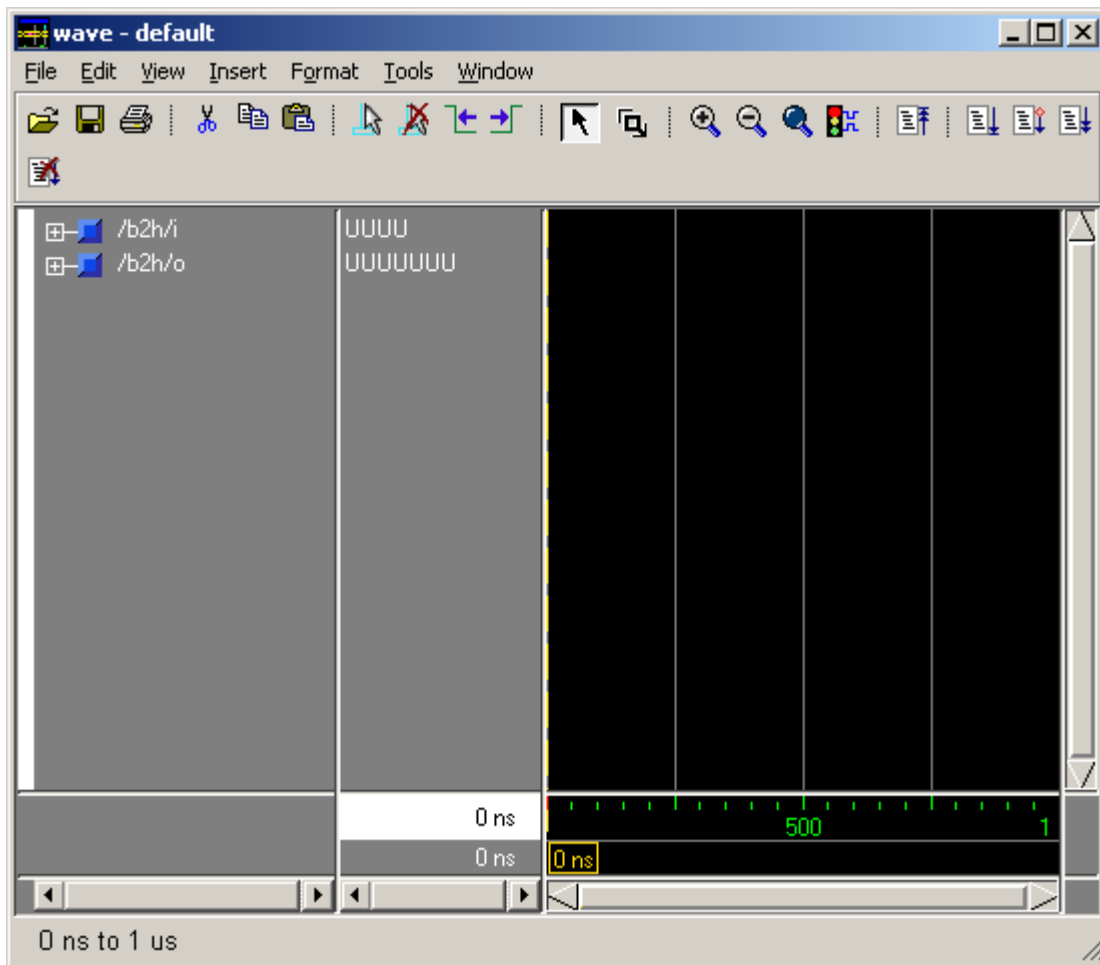


Figure 5. Wave signal window.

To run simulation type in the following in the command window:

```
% run 100 ns
```

The waveform window should look like the one below. Select **Zoom -> Zoom Full** to expand the wave display. You may have to adjust the area used to display the waveform names to see the complete name and its current std_logic value. Click and hold the mouse on the vertical time cursor and move it across the waveforms then release the cursor. The values beside the signal name show the current signal value with respect to the time cursor. Additional cursors can be added by selecting **Cursor -> Add Cursor** from the wave window.

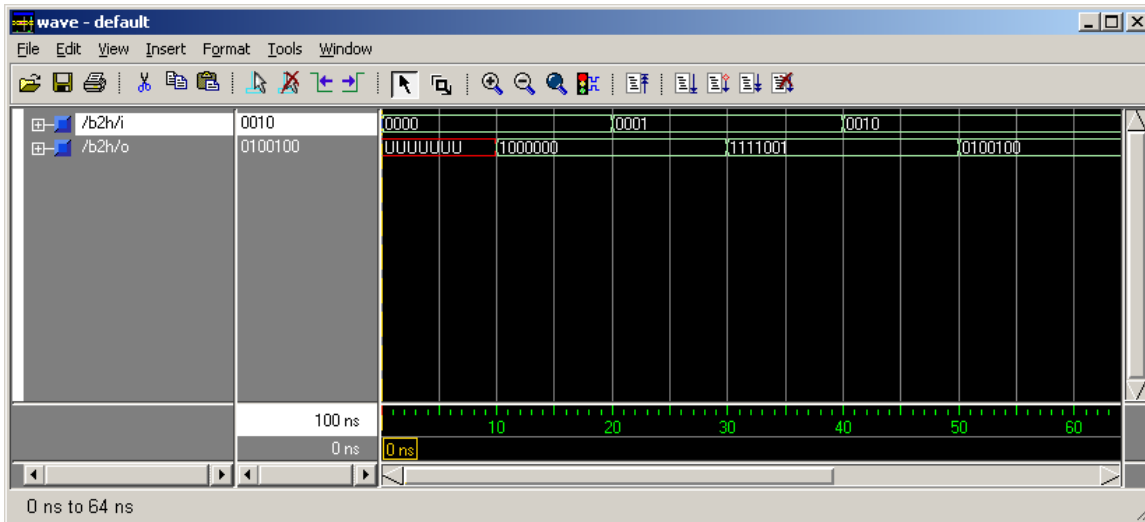


Figure 6. Simulation results.

Observe the results. Verify that your design is working properly!

Simulation results can also be observed by selecting **Add->List->Signals in Region** from the signals window.

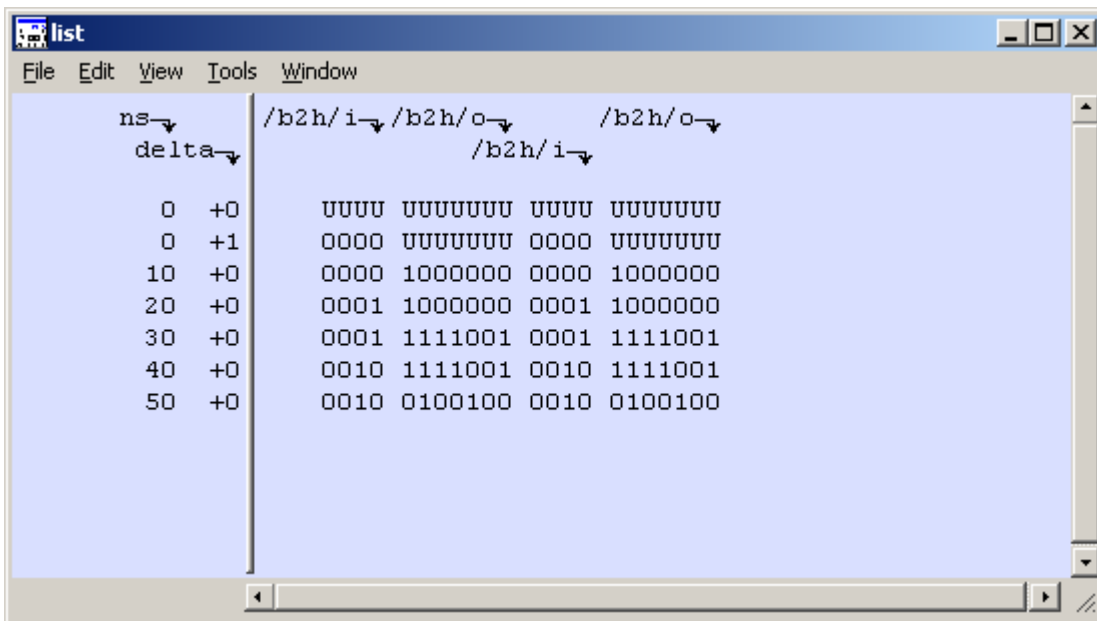
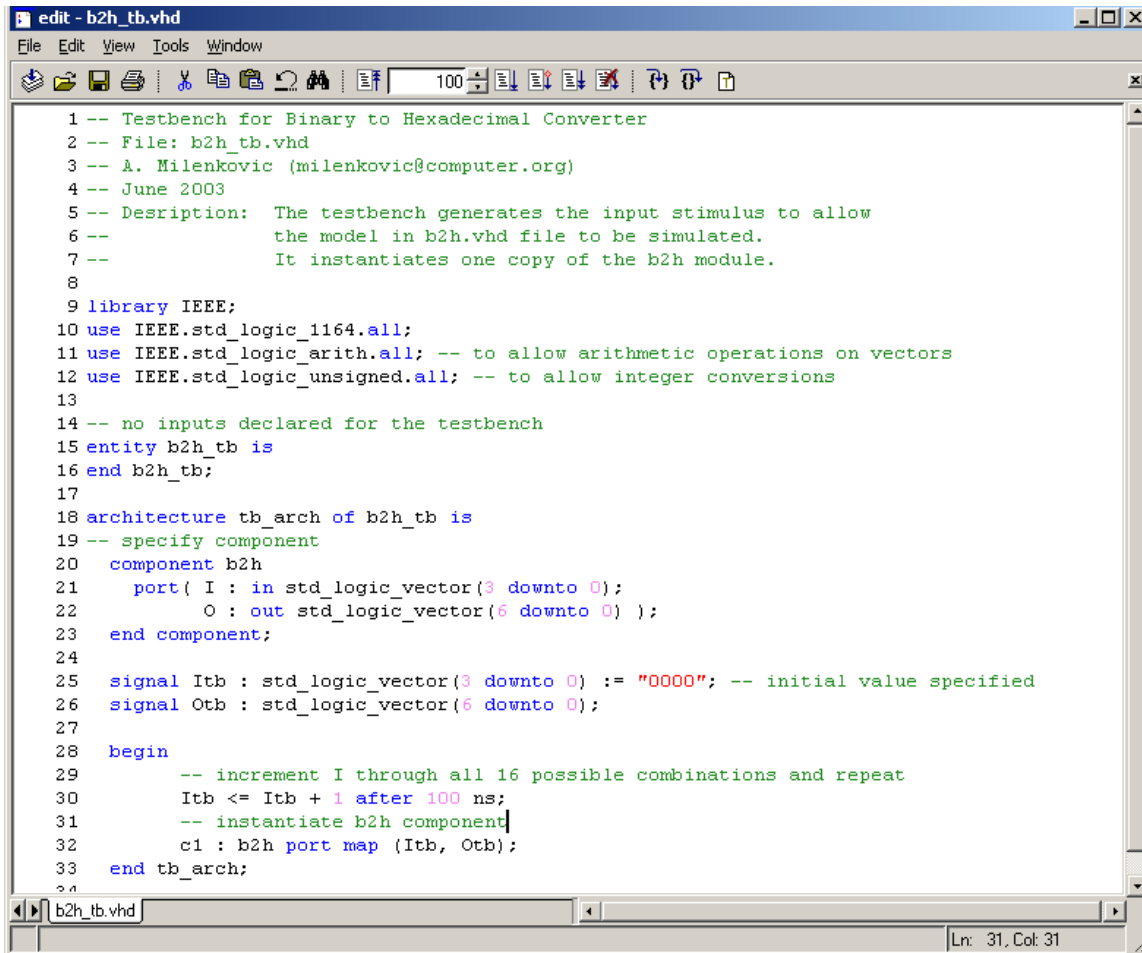


Figure 7. List signals view.

To create a VHDL testbench

A convenient way to enter stimulus is to create a separate test bench file which is written in VHDL to drive the inputs of the module that is to be tested. In this arrangement, the original VHDL model which is being simulated is instantiated as a component within the test bench file. A file which will

accomplish this purpose for the binary converter example, called t2h_tb.vhd, is shown in Figure 2.



```
1 -- Testbench for Binary to Hexadecimal Converter
2 -- File: b2h_tb.vhd
3 -- A. Milenkovic (milenkovic@computer.org)
4 -- June 2003
5 -- Description: The testbench generates the input stimulus to allow
6 --               the model in b2h.vhd file to be simulated.
7 --               It instantiates one copy of the b2h module.
8
9 library IEEE;
10 use IEEE.std_logic_1164.all;
11 use IEEE.std_logic_arith.all; -- to allow arithmetic operations on vectors
12 use IEEE.std_logic_unsigned.all; -- to allow integer conversions
13
14 -- no inputs declared for the testbench
15 entity b2h_tb is
16 end b2h_tb;
17
18 architecture tb_arch of b2h_tb is
19 -- specify component
20   component b2h
21     port( I : in std_logic_vector(3 downto 0);
22           O : out std_logic_vector(6 downto 0) );
23   end component;
24
25   signal Itb : std_logic_vector(3 downto 0) := "0000"; -- initial value specified
26   signal Otb : std_logic_vector(6 downto 0);
27
28   begin
29     -- increment I through all 16 possible combinations and repeat
30     Itb <= Itb + 1 after 100 ns;
31     -- instantiate b2h component
32     c1 : b2h port map (Itb, Otb);
33   end tb_arch;
```

Figure 8. VHDL Testbench for the b2h.

To simplify things, the test bench file should be saved and compiled under the same folder (directory) as the model file, which is to be simulated. To do this for the binary converter example one would simply type

```
% vcom b2h_tb.vhd <press enter/return key>
```

Start the simulator (Simulate->Simulate) and load the b2h_tb entity. Select the view you want and run simulation for 1600 ns. The list view is shown below.

ns	delta	/b2h_tb/itb	/b2h_tb/otb
0	+0	0000	UUUUUUUU
10	+0	0000	10000000
100	+0	0001	10000000
110	+0	0001	11110001
200	+0	0010	11110001
210	+0	0010	01001000
300	+0	0011	01001000
310	+0	0011	01100000
400	+0	0100	01100000
410	+0	0100	00110001
500	+0	0101	00110001
510	+0	0101	00100010
600	+0	0110	00100010
610	+0	0110	00000010
700	+0	0111	00000010
710	+0	0111	11110000
800	+0	1000	11110000
810	+0	1000	00000000
900	+0	1001	00000000
910	+0	1001	00110000
1000	+0	1010	00110000
1010	+0	1010	00010000
1100	+0	1011	00010000
1110	+0	1011	00000011
1200	+0	1100	00000011
1210	+0	1100	10001100
1300	+0	1101	10001100
1310	+0	1101	01000001
1400	+0	1110	01000001
1410	+0	1110	00000011
1500	+0	1111	00000011
1510	+0	1111	00001111
1600	+0	0000	00001111

Figure 9. List view for b2h simulation using testbench.