ENSC 350 ModelSim Altera Tutorial

This is a quick guide get you started with the ModelSim Altera simulator. ModelSim is only a functional verification tool so you will also have to use Quartus II to complete timing analysis on your design before you can be sure it will work the DE2 hardware.

ModelSim Basic Simulation (Optional)

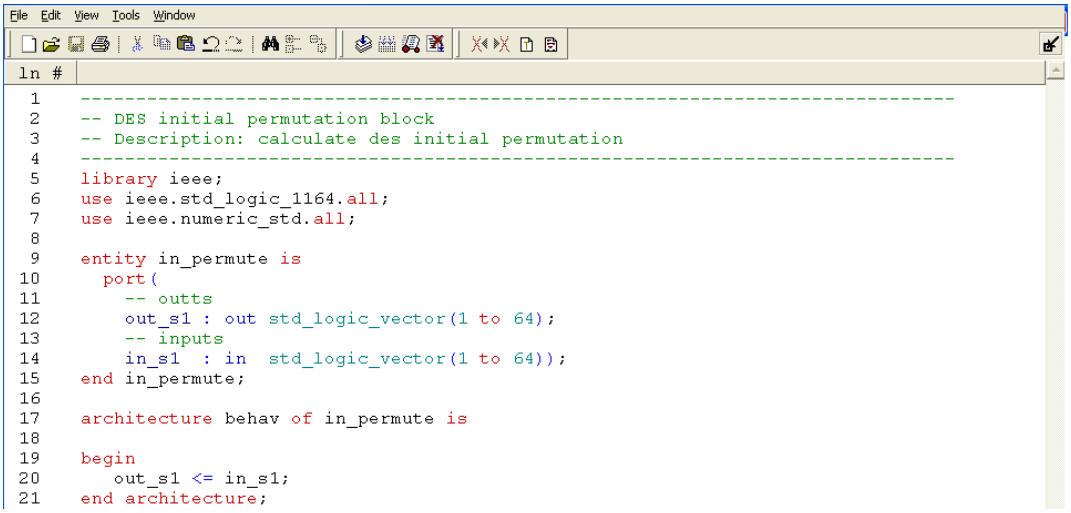
It is recommended that you complete the exercise "Basic Simulation" in chapter 3 of the ModelSim tutorial. This will give you all the background you need for lab 2.

Select **Help > PDF Documentation > Tutorial** to view ModelSim tutorials. Go to Chapter 3 "Basic Simulation". You can ignore the section "Set Breakpoints and Step through the Source". The ability to set breakpoints is very useful for designs with state machines or multiple execution paths but offers little gain when verifying 'pure datapath' designs like many of the components in lab2. This exercise should not take more than 15 minutes to complete.

ModelSim and Lab2

Setup Tutorial Directory and Sources

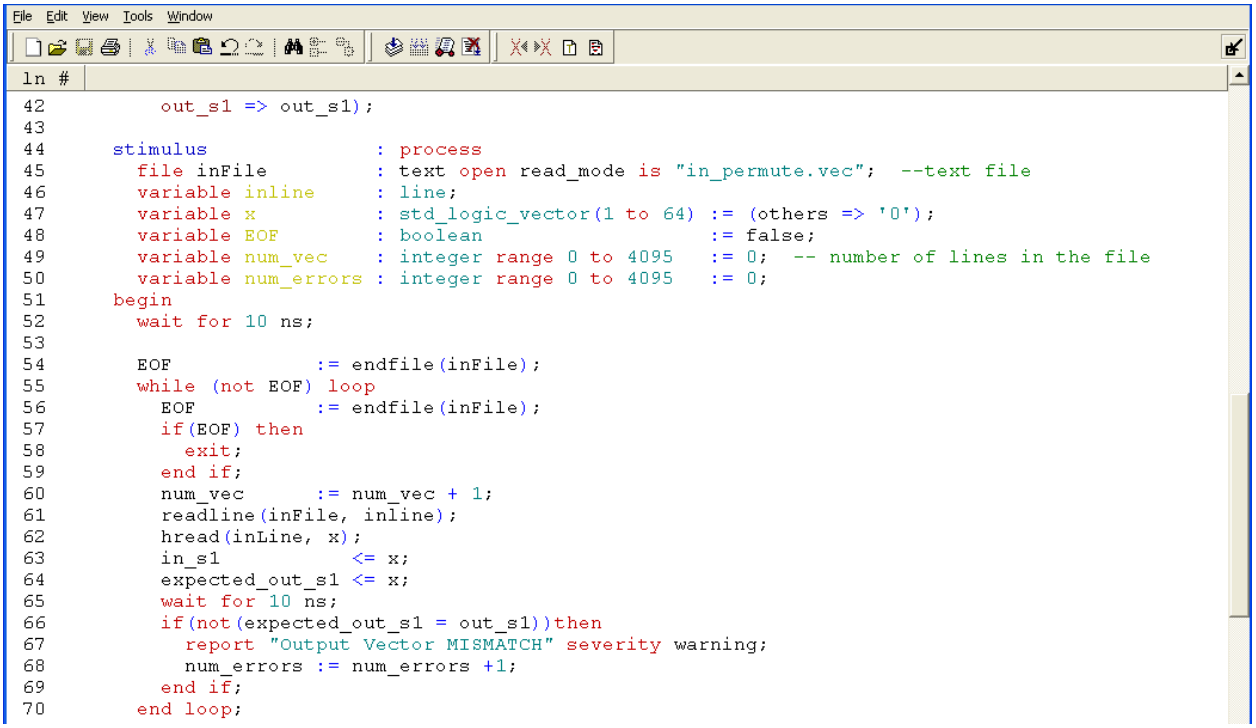
- Create a new **tutorial** folder and copy the files: '*templates/in_permute.vhd*', '*vectors/in_permute.vec*' and '*sim/in_permute_tb.vhd*' into this folder.
- Open ModelSim Altera Edition program. Select **File** > **Change Directory** and change to the tutorial directory you created.
- Open the file 'tutorial/in_permute.vhd' for editing. Select **File > Open > in_permute.vhd**. Note: You may choose to use another text editor for viewing and editing VHDL source files. Assign the out_s1 output to be in_s1. Your new code should look something like Figure 1:



```
ln #
 1   ----------------------------------------------------------------------------
 2   -- DES initial permutation block
 3   -- Description: calculate des initial permutation
 4   ----------------------------------------------------------------------------
 5   library ieee;
 6   use ieee.std_logic_1164.all;
 7   use ieee.numeric_std.all;
 8
 9   entity in_permute is
10     port(
11       -- outts
12       out_s1 : out std_logic_vector(1 to 64);
13       -- inputs
14       in_s1  : in  std_logic_vector(1 to 64));
15   end in_permute;
16
17   architecture behav of in_permute is
18
19   begin
20      out_s1 <= in_s1;
21   end architecture;
```

Figure 1: Updated in_permute.vhd

- Save your changes and close '*tutorial/in_permute.vhd*'. Select **File > Save**.
- Open the file '*tutorial/in_permute_tb.vhd'* for editing. Remove line 64 which reads the second value of the .vec file line into variable x. Also update file path for the *InFile* variable on line 45 to point to the new vector file location. Your code should now look like Figure 2:



```
ln #
42          out_s1 => out_s1);
43
44      stimulus              : process
45        file inFile         : text open read_mode is "in_permute.vec";  --text file
46        variable inline     : line;
47        variable x          : std_logic_vector(1 to 64) := (others => '0');
48        variable EOF        : boolean                   := false;
49        variable num_vec    : integer range 0 to 4095   := 0;  -- number of lines in the file
50        variable num_errors : integer range 0 to 4095   := 0;
51      begin
52        wait for 10 ns;
53
54        EOF            := endfile(inFile);
55        while (not EOF) loop
56          EOF          := endfile(inFile);
57          if(EOF) then
58            exit;
59          end if;
60          num_vec      := num_vec + 1;
61          readline(inFile, inline);
62          hread(inLine, x);
63          in_s1          <= x;
64          expected_out_s1 <= x;
65          wait for 10 ns;
66          if(not(expected_out_s1 = out_s1))then
67            report "Output Vector MISMATCH" severity warning;
68            num_errors := num_errors +1;
69          end if;
70        end loop;
```

Figure 2: Updated in_permute_tb.vhd

- Save your changes and close '*tutorial/in_permute_tb.vhd*'. Select **File > Save**.

Now your design should be assigning the out_s1 port to the value on the in_s1 port and your testbench should be assigning the expected_out_s1 and in_s1 signals to the same value. This should allow you to complete this tutorial without "Mismatch" warnings and without already having a correct in_permute.vhd implementation.

Overview of the ModelSim tool flow

Recall the ModelSim "Basic Simulation Tool Flow" shown in Figure 3.

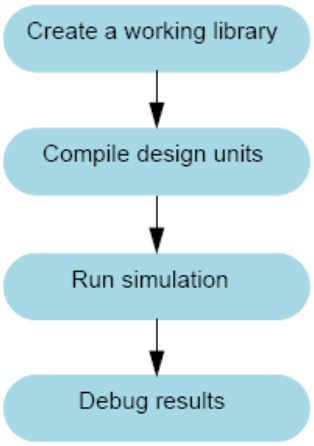


Figure 3: ModelSim Basic Simulation Flow[1]

The remainder of this tutorial will cover the first three steps of this flow.

Create a working library

- Select **File** > **New** > **Library**. In the "Create a New Library" dialog box, select "Create a new library and a logical mapping to it" and enter 'work' for both the 'Library Name' and 'Physical Library Name' options.

Compile design Units

- Select **Compile > Compile**. In the "Compile Source Files" dialog box select both in_permute.vhd and in_permute_tb.vhd and click **Compile**. Once it compiles correctly, click **Done**.

OR

- From the transcript window type :
  - vcom in_permute.vhd
  - vcom in_permute_tb.vhd

Using 'do' Files

You can add all the commands you wish to execute to a *.do* file and run this file from the transcript window.

[1] Mentor Graphics **ModelSim® Tutorial** Software Version 6.3g, May 2008

- Select **File > New > Source > Do** to open a new *.do* file. Copy compile commands above into the do file and Save it as *do_in_permute.do*.
- From the transcript window type:
  do do_in_permute

This is equivalent to compiling each file individually. ModelSim 'do' files are macro files that can contain any ModelSim command. As your structural design grows, 'do files' will become very useful for compiling multiple files and re-running simulations.

Run Simulation

Load the in_permute block tesbench entity.

- From the transcript window type :
  - vsim in_permute_tb

Add signals to the waveform viewer.

- In the 'workspace' window. Select the 'sim' tab. Right-Click on the in_permute_tb object and select **Add > Add to Wave**.

Notice the output in the transcript window: add wave -r *
You could have instead loaded your design and wave by placing the following in you '.do' file:

```
vsim in_permute_tb
add wave -r *
```

In fact a good way to learn ModelSim commands is to use the GUI and watch the resulting transcript output.

The default radix for signals of type 'std_logic_vector' is binary. In the wave window right click on the signal 'in_permute_tb/in_s1' and select **Radix > Hexadecimal** to view hexadecimal values.

Run the simulation.

- From the transcript window type :
  - run 500 ns;

This will run the testbench for 500 ns of 'simulation time'. Observe the updated waveform in the wave window. Experiment with zoom, cursors and transition finder functions.

- In the transcript window type :

- run -a

This will run the simulation to the next breakpoint. The supplied testbenches provided will ‘break’ automatically when you reach the end of a vector file.

Close the simulation

- Select **Simulate > End Simulation**.

OR

- From the transcript window type :
  - quit -sim

This will end the current simulation. Not including the -sim option will close ModelSim.