Using the ModelSim-Intel FPGA Simulator by Drawing Waveforms

For Quartus Prime 18.0

1 Introduction

This tutorial introduces the simulation of VHDL code using the Graphical Waveform Editor in the *ModelSim-Intel FPGA* simulator. We assume that you are using *ModelSim-Intel FPGA Starter Edition version 18.0*. This software can be downloaded and installed from the *Download Center for Intel FPGAs*. In this download center, you can select release *18.0* of the *Quartus Prime Lite Edition*, and then on the `Individual Files` tab choose to download and install the *ModelSim-Intel FPGA Starter Edition* software. We assume that you are using a computer that is running the Windows operating system. If you are using the Linux operating system then minor differences to the instructions would apply, such as using a / filesystem delimiter rather than the \ delimiter that is used with Windows.

This tutorial is intended for introductory use of ModelSim by students who are just beginning to learn about the relevant topics. For students who have gained some familiarity with logic design and VHDL code we recommend the approach presented in the tutorial *Using the ModelSim-Intel FPGA Simulator with VHDL Testbenches*.

Contents:

- Design Project
- Creating Waveforms for Simulation
- Simulation
- Making Changes and restarting the simulation

Requirements:

- ModelSim-Intel FPGA Starter Edition software
- A computer running either Microsoft* Windows* (version 10 is recommended) or Linux (Ubuntu, or a similar Linux distribution). The computer would typically be either a desktop computer or laptop, and is used to run the ModelSim software.

Optional:

- Intel Quartus® Prime software
- A DE-series development and education board, such as the DE1-SoC board. These boards are described on Intel's FPGA University Program website, and are available from the manufacturer Terasic Technologies.

2 Getting Started

The ModelSim Simulator is a sophisticated and powerful tool that supports a variety of usage models. In this tutorial we focus on only one design flow: using the ModelSim software as a *stand-alone* program to perform *functional* simulations, with simulation inputs specified by *drawing waveforms*, and with simulator commands selected by using the *menu items* that are available in ModelSim's graphical user interface (GUI). Other possible design flows for using ModelSim include *invoking* it from within the Intel Quartus Prime software, performing *timing* simulations, specifying simulation inputs by using a *testbench*, and selecting simulator commands via *script* files. These flows are not described here, but can be found in other documentation that is available on the Internet.

This tutorial is aimed at the reader who wishes to simulate circuits defined by using the VHDL language. An equivalent tutorial is available for the user who prefers the Verilog language.

3 Design Project

When using the ModelSim GUI to execute simulation commands, it is convenient to work in the context of a ModelSim *project*. A project includes the design files that specify the circuit to be simulated. We will first create a folder to hold the project used in this tutorial. Create a new folder for this tutorial with a name such as *ModelSim_Tutorial* and then make a subfolder named *Majority* for this example project.

To illustrate the simulation process, we will use a very simple logic circuit that implements the majority function of three inputs, x_1, x_2 and x_3. The circuit is defined by the expression

$$f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$$

In VHDL, this circuit can be specified as follows:

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY majority IS
    PORT (  x1, x2, x3  : IN    STD_LOGIC;
            f           : OUT   STD_LOGIC);
END ENTITY;

ARCHITECTURE Behaviour OF majority IS
BEGIN
    f <= (x1 AND x2) OR (x1 AND x3) OR (x2 AND x3);
END Behaviour;
```

Enter this code into a file called *majority.vhd*, in the folder *ModelSim_Tutorial\Majority*. You can create the file using any text editor of your choosing (a text editor is also available within the ModelSim GUI). Open the ModelSim simulator. In the displayed window select File > New > Project, as shown in Figure 1. A `Create Project` pop-up box will appear, as illustrated in Figure 2. Specify the name of the project; we chose the name *majority*. Use the Browse button in the `Project Location` box to specify the location of the folder that you created for the project. ModelSim uses a working library to contain the information on the design in progress; in the `Default Library Name` field we used the name *work*. Click OK.

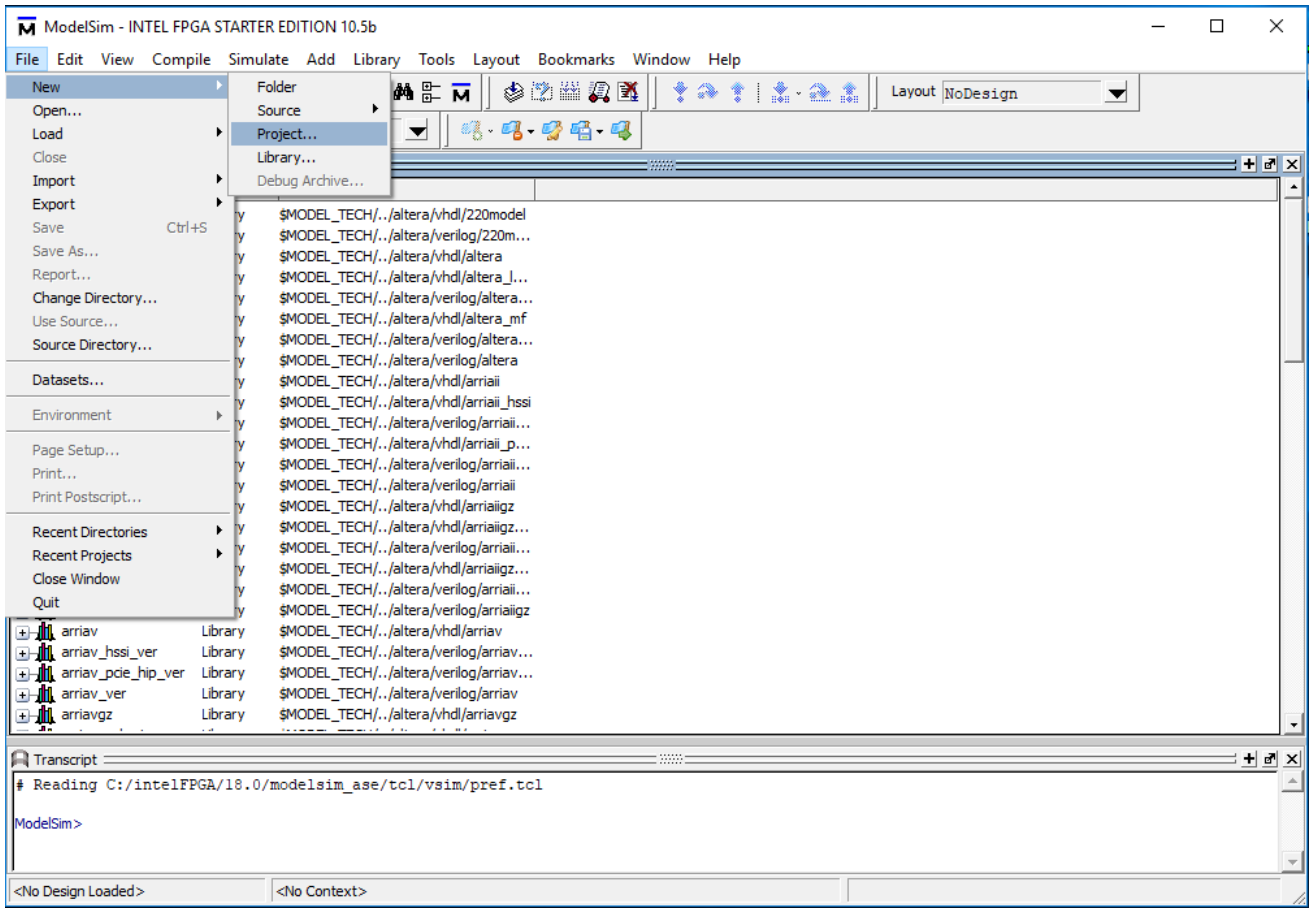

Figure 1. The ModelSim window.

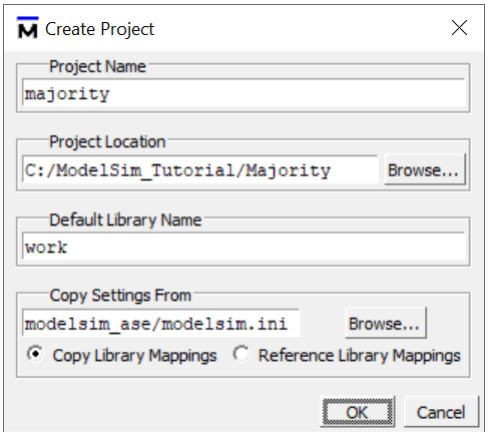

Figure 2. Created Project window.

In the pop-up window in Figure 3, click on Add Existing File and add the file *majority.vhd* to the project as shown in Figure 4. Click OK, then close the window from Figure 3.

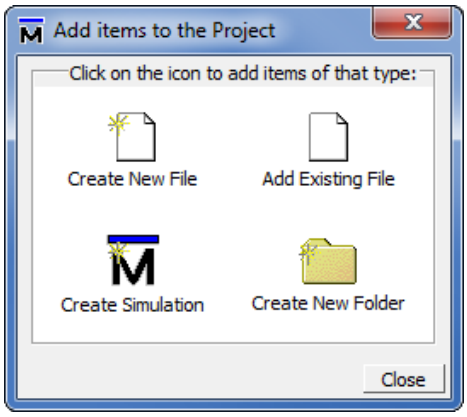

Figure 3. Add Items window.

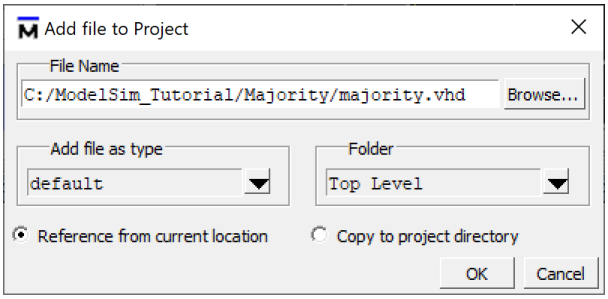

Figure 4. Add File window.

At this point, the main Modelsim window will include the file as indicated in Figure 5, with a question mark in the `Status` column. Now, select Compile > Compile All. As illustrated in the figure, the ModelSim GUI will indicate in the `Transcript` window (at the bottom) that the code in the *majority.vhd* file was successfully compiled, and a corresponding check mark will be displayed in the `Status` column. If you made any errors when typing the code, fix them accordingly and then compile again. The VHDL code is now ready for simulation.

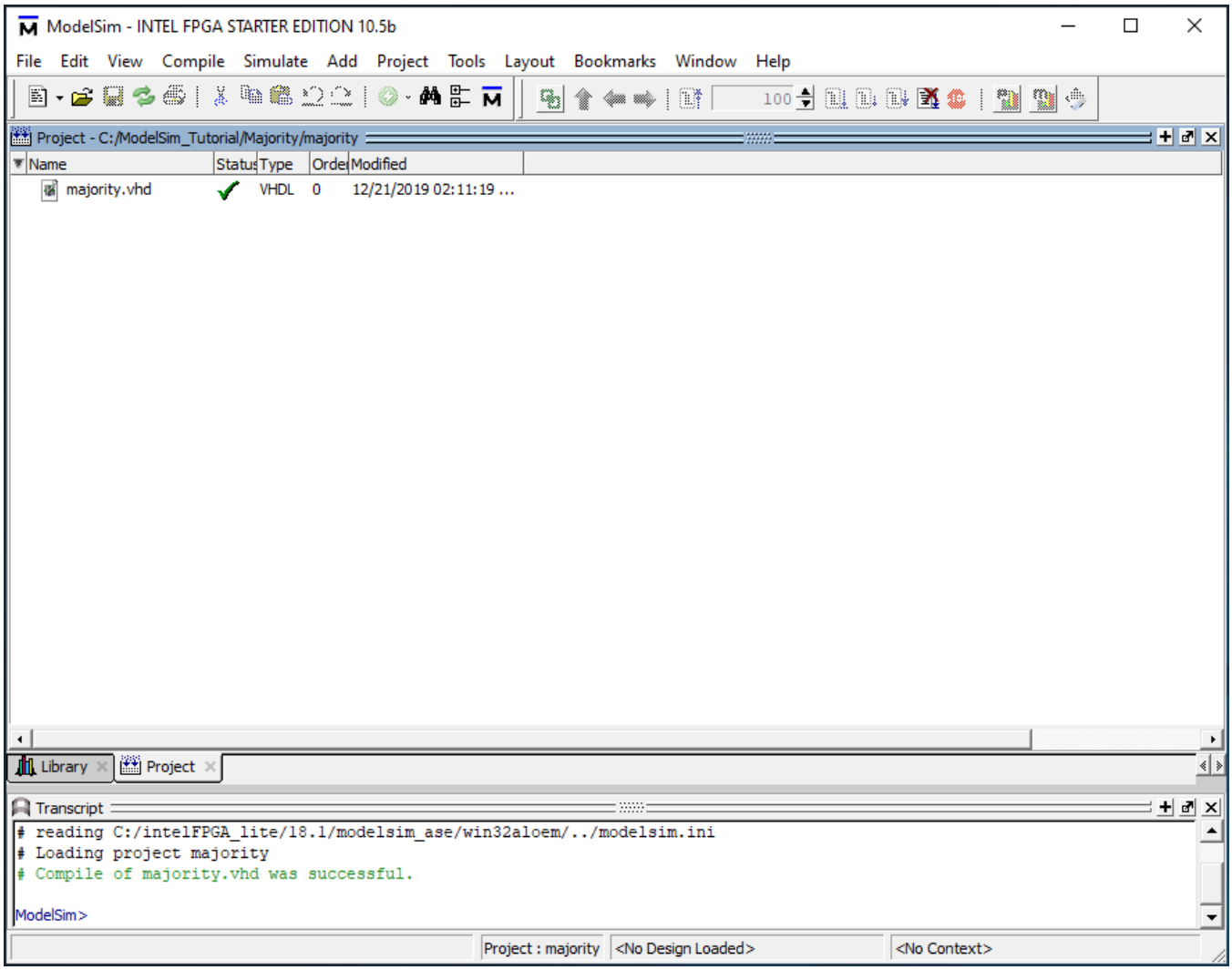

Figure 5. ModelSim window after compilation.

4 Creating Waveforms for Simulation

To perform simulation of the designed circuit, it is necessary to enter the simulation mode by selecting Simulate > Start Simulation. This leads to the window in Figure 6.

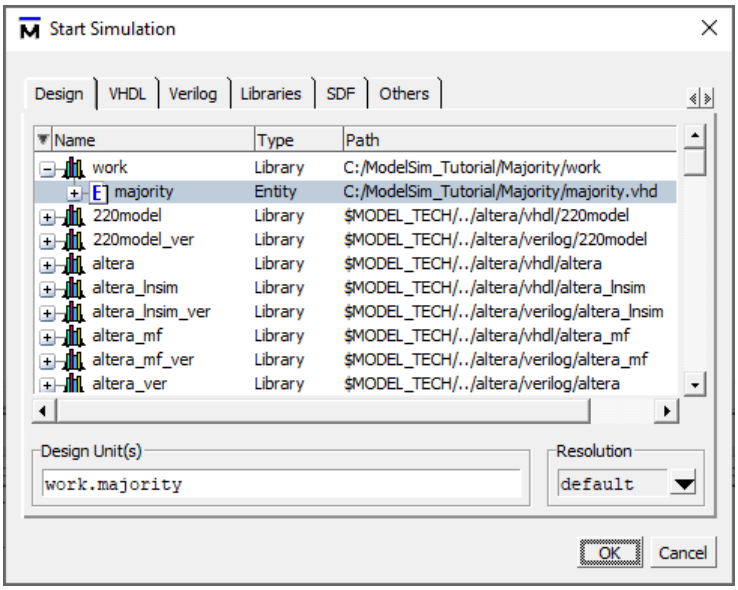

Figure 6. Start Simulation window.

Expand the *work* folder and select the design called *majority*, as shown in the figure. Then click OK. Now, the ModelSim GUI opens a number of windows and toolbars, as illustrated in Figure 7, that are useful for performing a simulation. The `Objects` window, shown in a blue color, lists the input and output signals of the designed circuit. The `Wave` window is used to display waveforms that are associated with these inputs and outputs. Figure 7 shows several toolbars that can be used to select various ModelSim GUI commands. To make your window look like the one in the figure, you may have to open or close some of the available toolbars. Right-clicking in the toolbar area, as indicated in Figure 8, allows you to show or hide toolbars.

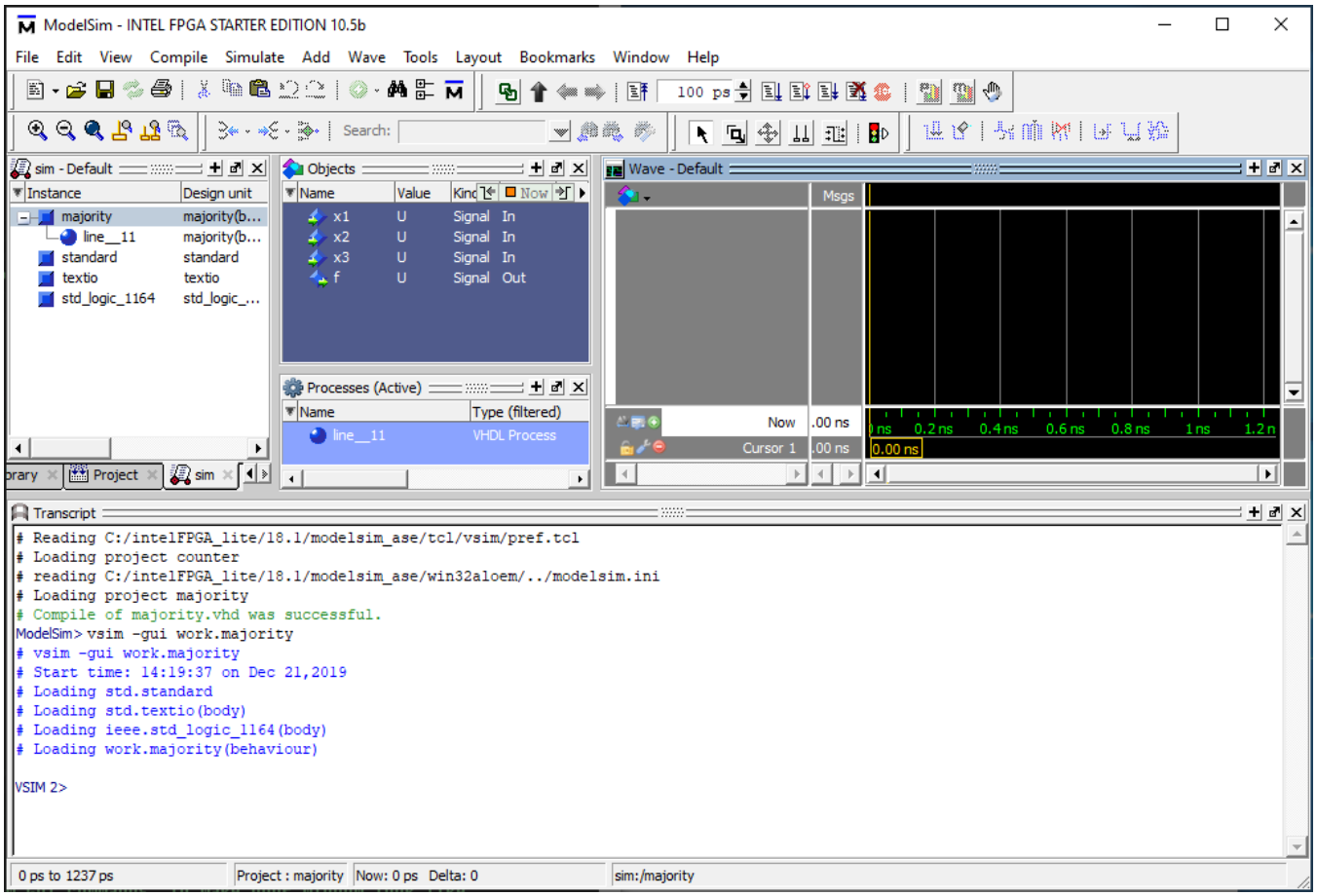

Figure 7. Simulation windows.

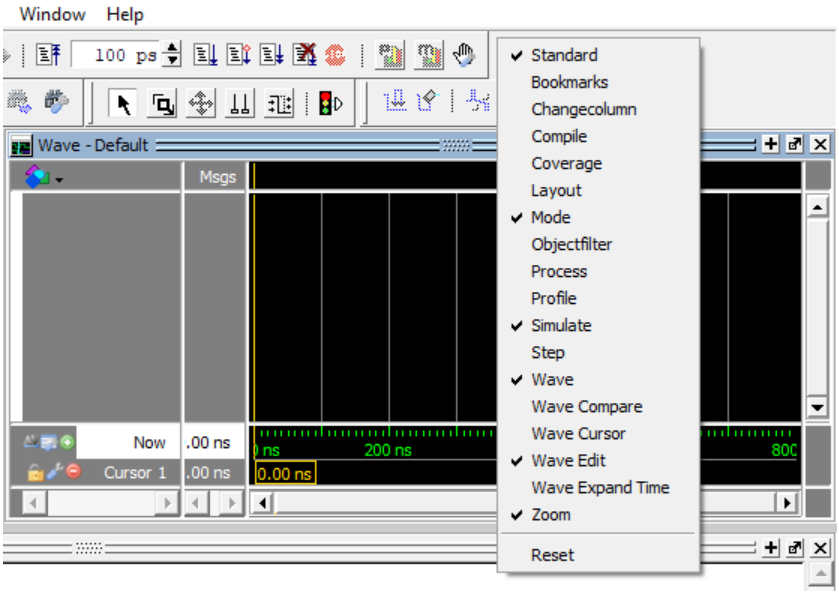

Figure 8. The toolbars shown in Figure 7.

To simulate the circuit we must first specify the values of input signals, which can be done by drawing the input waveforms using the `Wave` window. Right-click in the `Wave` window to select Zoom Range and in the pop-up window that will appear specify the range from 0 to 800 ns. This selection should produce an image like the one in Figure 9. If you need to change the timeline display units, right-click on the timeline and select Grid, Timeline, & Cursor Control..., then select ns in the `time units` drop-down menu.

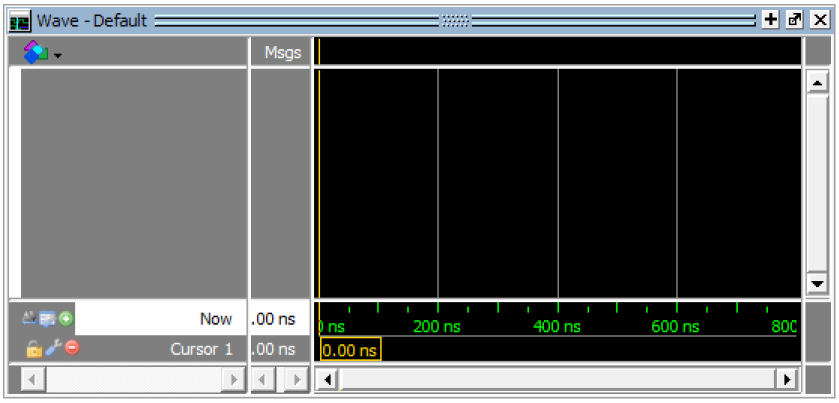

Figure 9. The `Wave` window.

For our simple circuit, we can do a complete simulation by applying all eight possible valuations of the input signals x_1, x_2 and x_3. The output f should then display the logic values defined by the truth table for the majority function. We will first draw the waveform for the x_1 input. In the `Objects` window, right-click on $x1$. Then, choose Modify

> Apply Wave in the drop-down box that appears, as shown in Figure 10. This leads to the window in Figure 11, which makes it possible to specify the value of the selected signal in a time period that has to be defined. Choose Constant as the desired pattern, zero as the start time, and 400 ns as the end time. Click Next. In the window in Figure 12, enter 0 as the desired logic value. Click Finish. Now, the specified signal appears in the `Wave` window, as indicated in Figure 13.

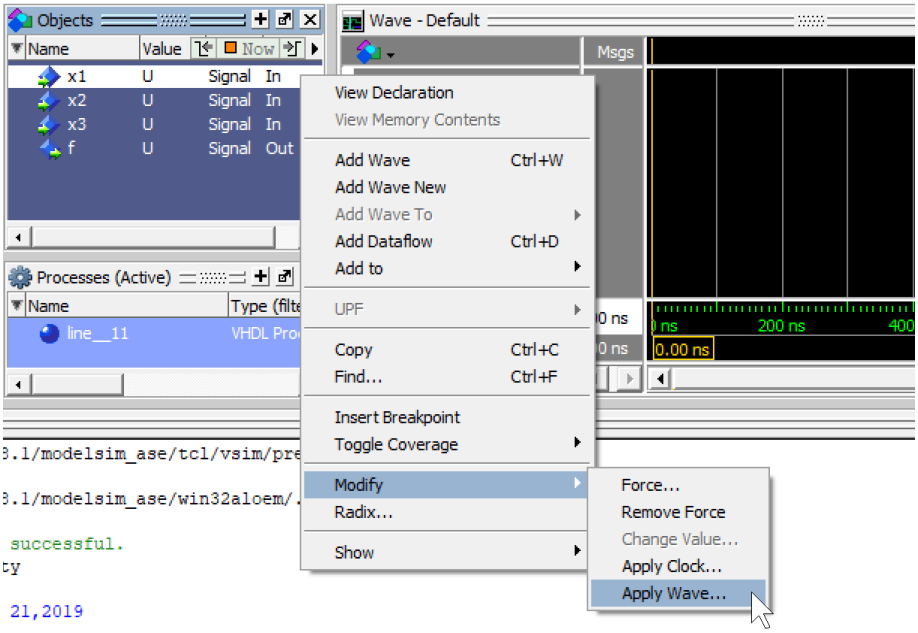

Figure 10. Selecting a signal in the `Objects` window.

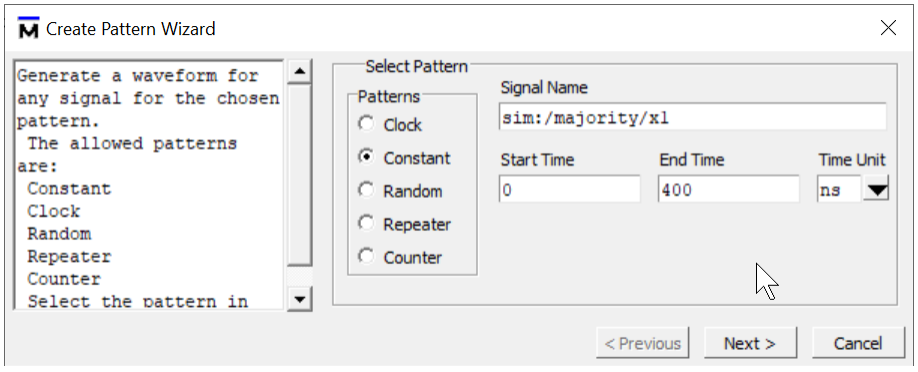

Figure 11. Specifying the type and duration of a signal.

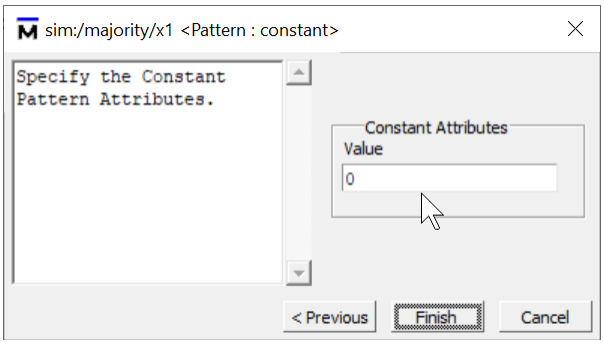

Figure 12. Specifying the value of a signal.

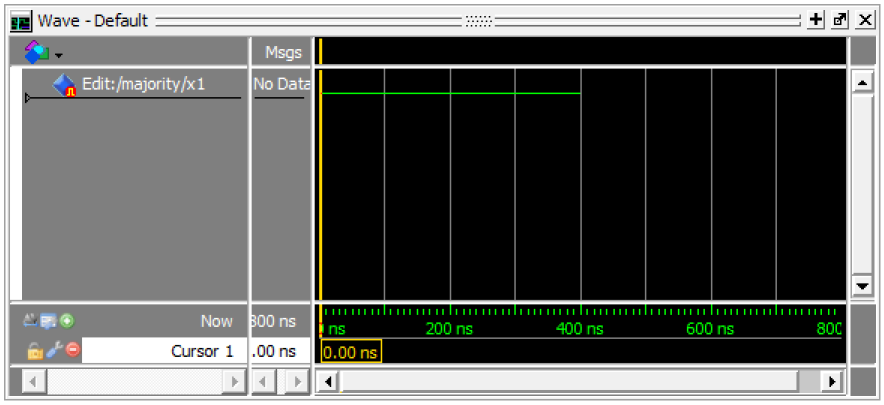

Figure 13. The updated `Wave` window.

To draw the rest of the x_1 signal, right-click on its name in the `Wave` window (make sure to click in the `Wave` window to make this change to the existing waveform, and not in the `Objects` window). In the drop-down window that appears, select Edit > Wave Editor > Create/Modify Waveform. This leads again to the window in Figure 11. Now, specify 400 ns as the start time and 800 ns as the end time. Click Next. In the window in Figure 12, specify 1 as the required logic value. Click Finish. This completes the waveform for x_1, as displayed in Figure 14.

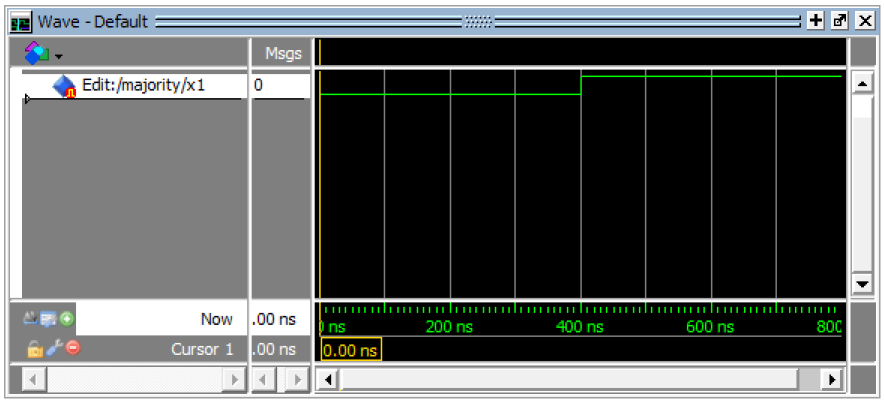

Figure 14. The completed waveform for x_1 input.

ModelSim provides different possibilities for creating and editing waveforms. To illustrate another approach, we will specify the waveform for x_2 by first creating it to have a 0 value throughout the simulation period, and then editing it to produce the required waveform. Repeat the above procedure, by right-clicking on $x2$ in the `Objects` window, to create a waveform for x_2 that has the value 0 in the interval 0 ns to 800 ns. So far, we used the `Wave` window in the `Select Mode` which is indicated by the highlighted icon . Now, click on the Edit Mode icon as indicated in Figure 15 (if the `Edit Mode` icon is *greyed-out*, then first click on the title bar of the `Wave` window to activate it). Note that `Edit Mode` opens some new toolbar icons for use in the editing process.

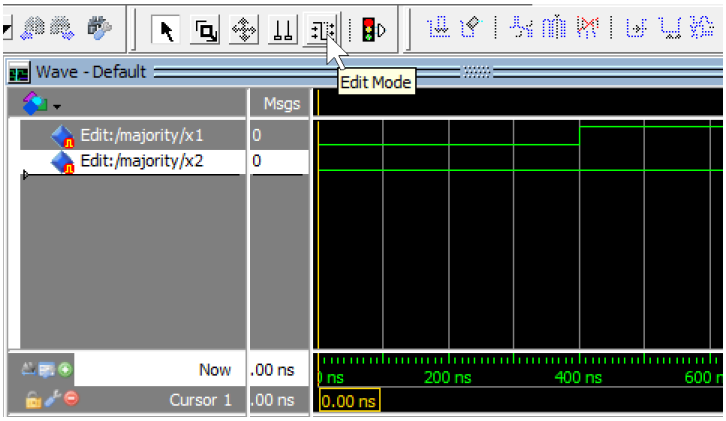

Figure 15. Selecting the Wave Edit mode.

The waveform for x_2 should change from 0 to 1 at 200 ns, then back to 0 at 400 ns, and again to 1 at 600 ns. Select $x2$ for editing by clicking on it. Then, click just to the right of the 200-ns point, hold the mouse button down and sweep to the right until you reach the 400-ns point. The chosen interval will be highlighted in white, as shown in Figure 16. Observe that the yellow cursor line appears and moves as you sweep along the time interval. To change the value of the waveform in the selected interval, click on the Invert icon as illustrated in the figure. A pop-up box in Figure 17 will appear, showing the start and end times of the selected interval. If the displayed times are

not exactly 200 and 400 ns, then correct them accordingly and click OK. The modified waveform is displayed in Figure 18. Use the same approach to change the value of x_2 to 1 in the interval from 600 to 800 ns, which should yield the result in Figure 19.

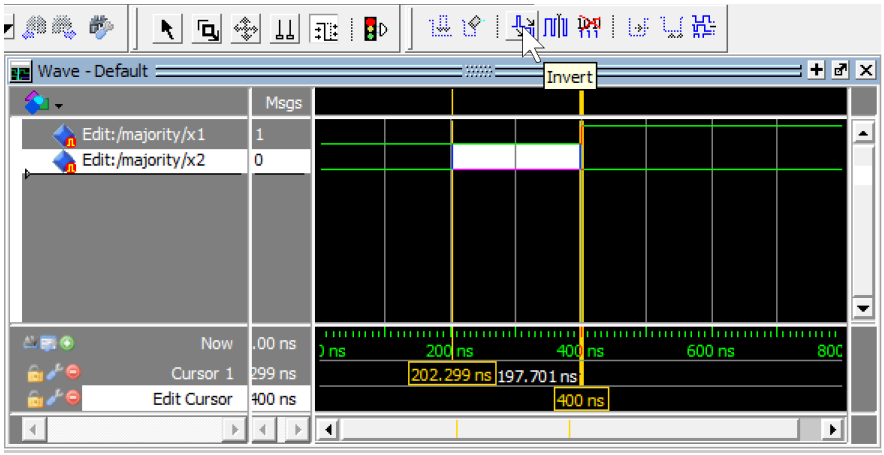

Figure 16. Editing the waveform.

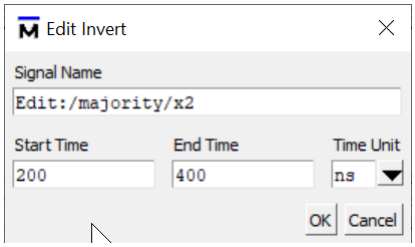

Figure 17. Specifying the exact time interval.

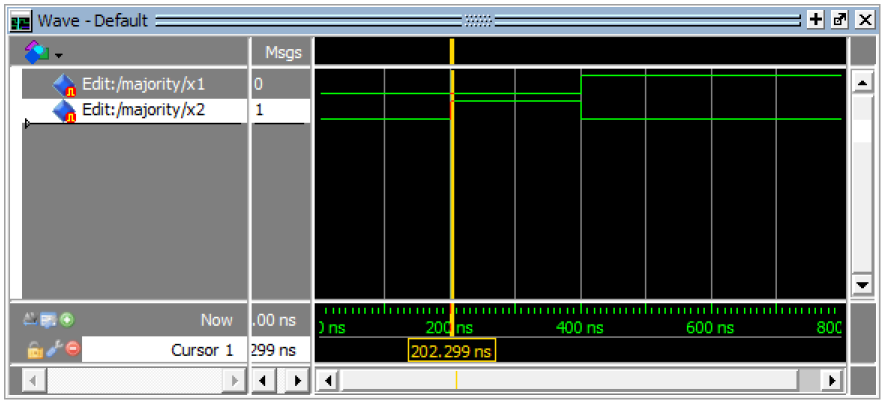

Figure 18. The modified waveform.

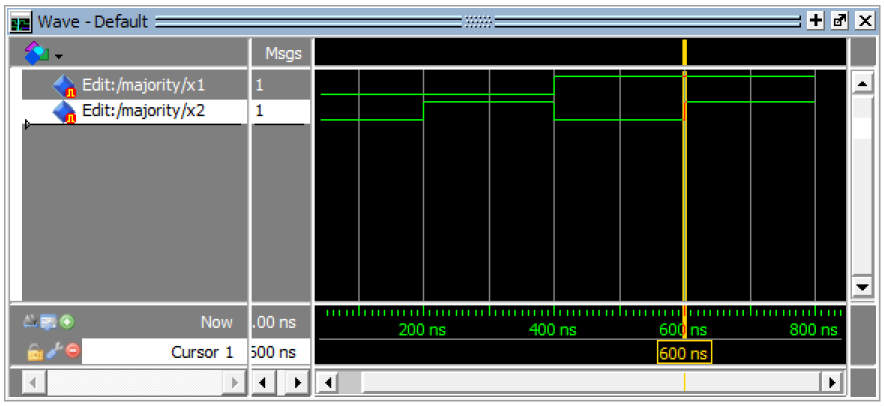

Figure 19. Completed waveforms for x_1 and x_2.

We will use a third approach to draw the waveform for x_3. This signal should alternate between 0 and 1 logic values at each 100-ns interval. Such a regular pattern is indicative of a *clock* signal that is used in many logic circuits. To illustrate how a clock signal can be defined, we will specify x_3 in this manner. Right-click on the x_3 input in the `Objects` window and select Modify > Apply Wave. In the `Create Pattern Wizard` window, select Clock as the required pattern, and specify 0 and 800 ns as the start and end times, respectively, as indicated in Figure 20. Click Next, which leads to the window in Figure 21. Here, specify 0 as the initial value, 200 ns as the clock period, and 50 as the duty cycle. Click Finish. Now, the waveform for x_3 is included in the `Wave` window.

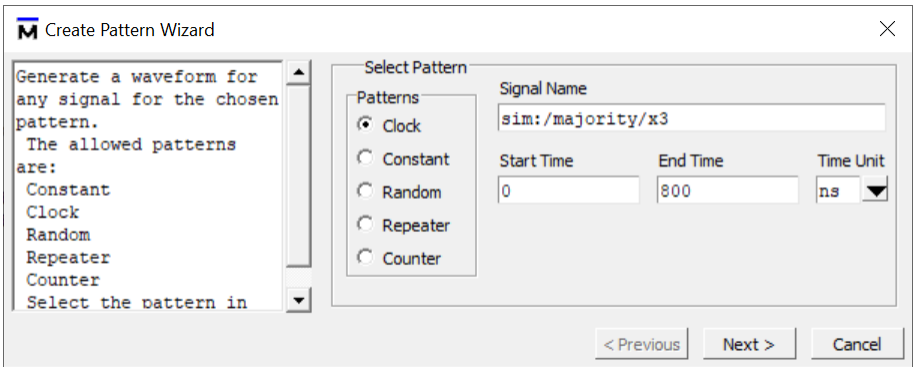

Figure 20. Selecting a signal of clock type.

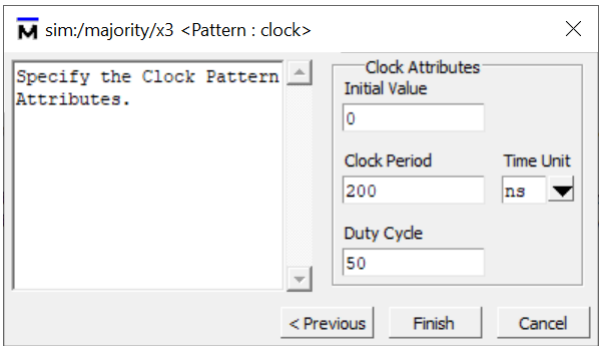

Figure 21. Defining the characteristics of a clock signal.

Lastly, it is necessary to include the output signal f. Right-click on f in the `Objects` window. In the drop-down menu that appears, select Add to > Wave > Selected Signals as shown in Figure 22. Alternatively, you could *drag-and-drop* the signal f from the `Objects` window into the `Wave` window. The result is the image in Figure 23.

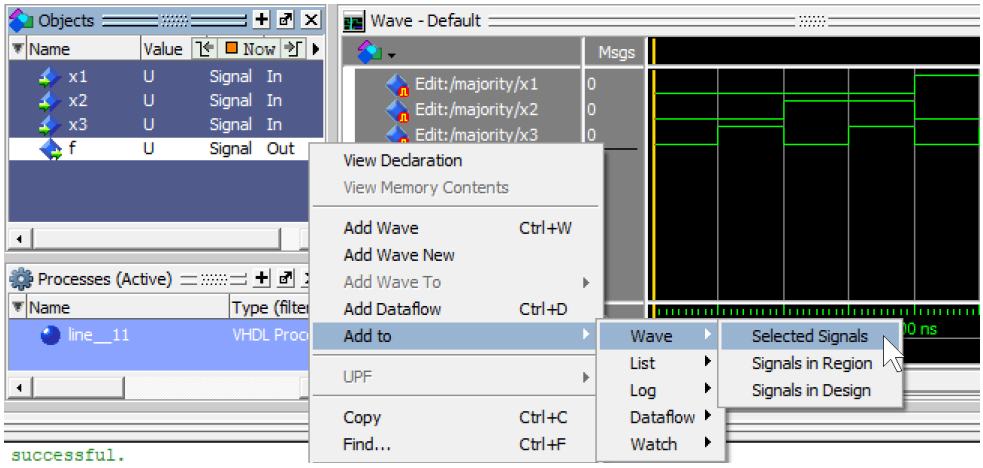

Figure 22. Adding a signal to the `Wave` window.

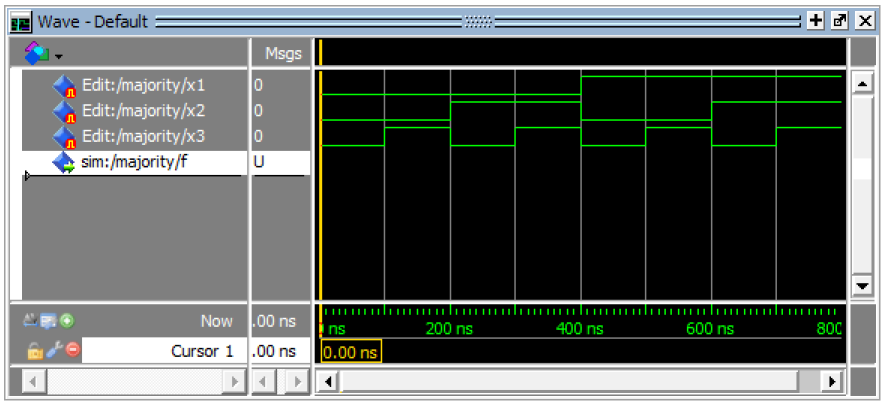

Figure 23. The completed `Wave` window.

Save the created waveforms by going to File > Save Format in the `Wave` window. We will save the file using the default name *wave.do*, as indicated in Figure 24.

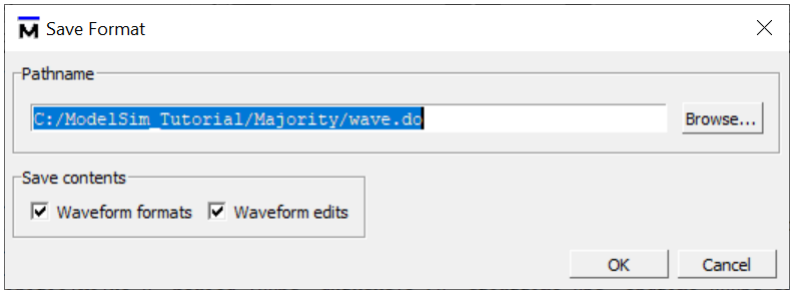

Figure 24. Saving the waveform file.

5 Simulation

To perform the simulation, in the toolbar area of the ModelSim GUI specify that the simulation should run for 800 ns, as illustrated in Figure 25 (If you don't see this toolbar, right-click in the toolbar area and select the Simulate toolbar). Then, click on the Run icon, as indicated in Figure 25. The result of the simulation will be displayed as presented in the figure. Observe that the output f is equal to 1 whenever two or three inputs have the value 1, which verifies the correctness of our design.

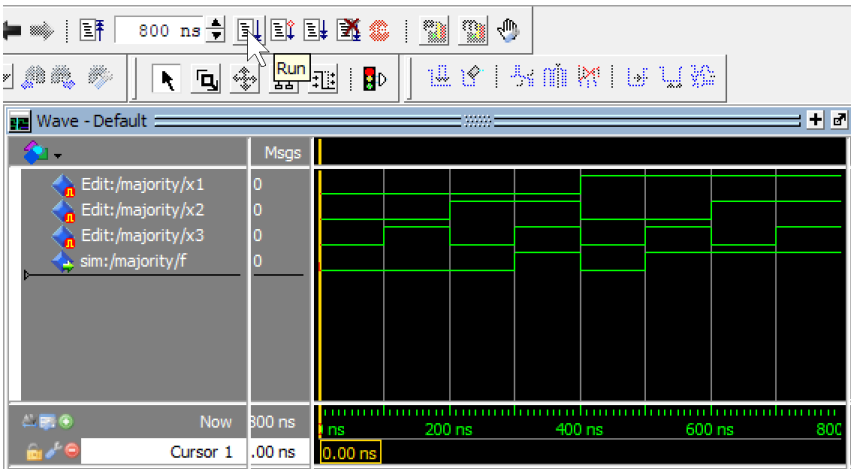

Figure 25. Running the simulation.

6 Making Changes and Resimulating

Changes in the input waveforms can be made using the approaches explained above. Then, it is necessary to restart the simulation using the altered waveforms. For example, change the waveform for x_1 to have the logic value 1 in the interval from 0 to 200 ns, as indicated in Figure 26. Now, click on the Restart icon shown in the figure. A pop-up box in Figure 27 will appear. Leave the default entries and click OK. Upon returning to the `Wave` window, simulate the design again by clicking on the Run icon. The result is given in Figure 28.

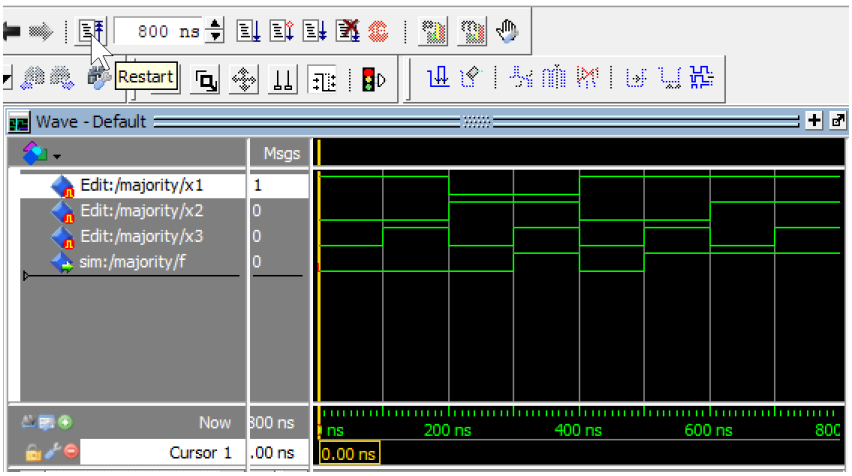

Figure 26. Changed input waveforms.

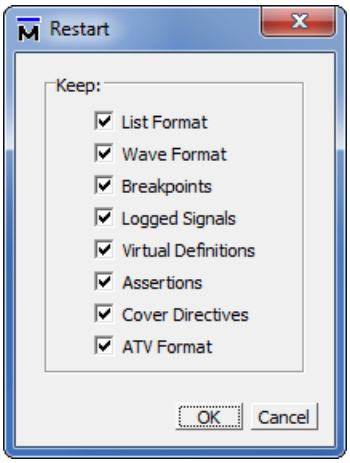

Figure 27. The Restart box.

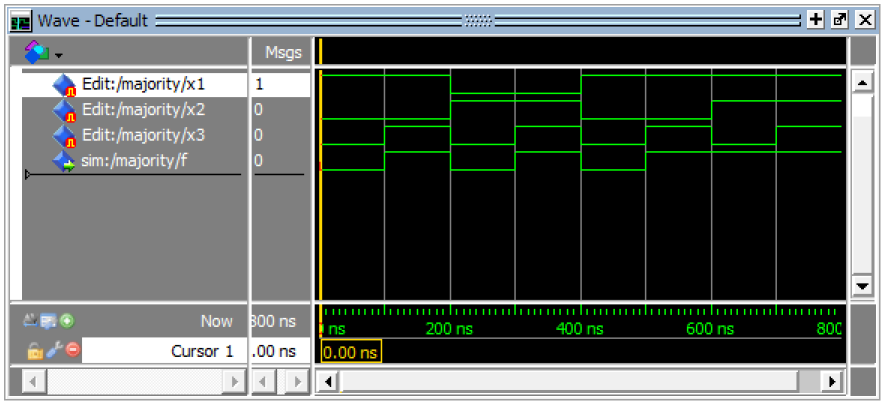

Figure 28. Result of the new simulation.

Simulation is a continuous process. It can be stopped by selecting Simulate > End Simulation in the main ModelSim window.

7 Simulating a Sequential Circuit

We will use another ModelSim project to illustrate some additional features of the `Wave` window. Consider the VHDL code shown below, which specifies a loadable down-counter:

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE ieee.std_logic_signed.ALL;
```

```
ENTITY counter IS
   PORT ( L     : IN     STD_LOGIC;
          Clock : IN     STD_LOGIC;
          Data  : IN     STD_LOGIC_VECTOR(2 DOWNTO 0);
          Q     : BUFFER STD_LOGIC_VECTOR(2 DOWNTO 0));
END ENTITY;

ARCHITECTURE Behaviour OF counter IS
    SIGNAL E : STD_LOGIC;         -- enable
BEGIN
    PROCESS (Clock, L, Data)      -- define the down-counter
    BEGIN
        IF L = '0' THEN
            Q <= Data;
        ELSIF Clock'EVENT AND Clock ='1' AND E = '1' THEN
            Q <= Q - "001";
        END IF;
    END PROCESS;
    -- stop the counter when Q = 0
    E <= '0' WHEN Q = "000" ELSE '1';
END Behaviour;
```

The counter can be synchronously loaded from the *Data* input when when the load input $L = 0$. When the count reaches zero the enable E is set to 0 and this stops the counter.

In the folder named *ModelSim_Tutorial* that you created for this tutorial, make a new subfolder called *Counter* to hold the ModelSim files for this project. In the folder *ModelSim_Tutorial\Counter* enter the VHDL code for the down-counter into a file called *counter.vhd*,

In ModelSim, make a new project by selecting File > New > Project, which opens the window from Figure 1 (if ModelSim prompts you to close the currently-open project, select Yes). In the `Create Project` pop-up box from Figure 2 give the project the name counter and use the Browse button to set the `Project Location` to *ModelSim_Tutorial\Counter*. Click OK to reach the pop-up from Figure 3 and then select Add Existing File. In the window from Figure 4 use the Browse button to add your *counter.v* file to the project.

Now, in the window from Figure 5 select the Compile > Compile All command. If you typed the VHDL code for the counter correctly, then you should see a message in the `Transcript` window reporting compilation success. If there are any errors, then fix the code as needed and compile again.

8 Specifying Simulation Inputs

To perform simulation of the counter, select the command Simulate > Start Simulation to reach the window in Figure 29. As indicated in the figure, expand the work folder, click on the counter design, and then select OK. Your ModelSim display should now look like the one shown in Figure 30. To populate the `Wave` window as displayed in the figure, first click on the signal L in the `Objects` window and then shift-click on the signal E so that all signals are selected. Now, drag-and-drop the selected signals into the `Wave` window. Set the Zoom Range of the Wave window to 1800 ns.

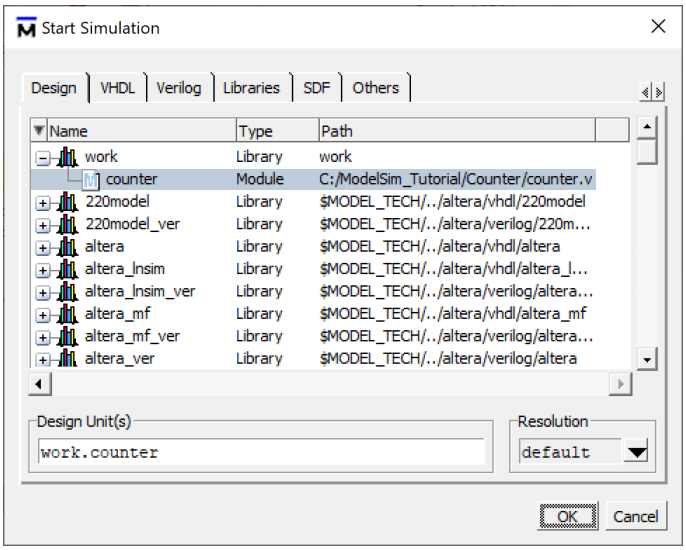

Figure 29. The start simulation window.

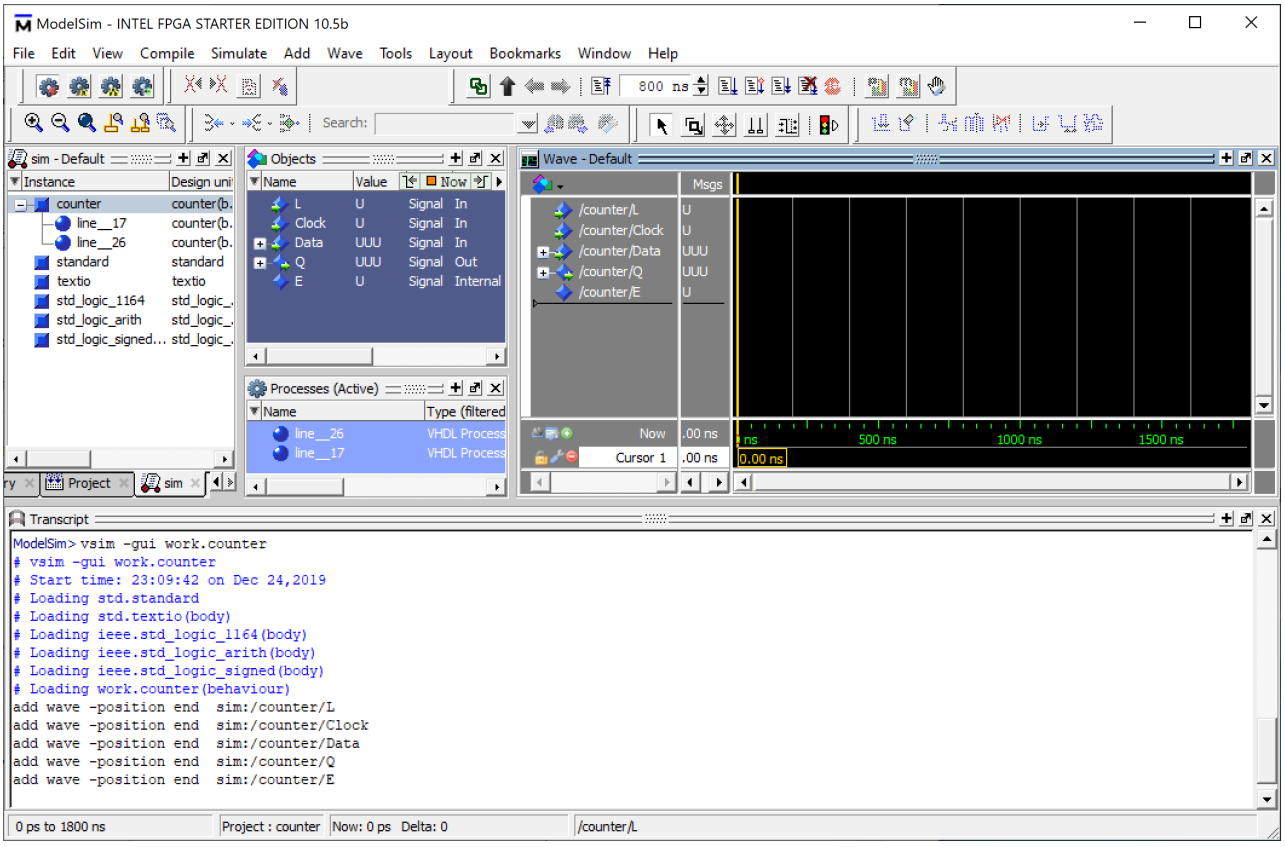

Figure 30. The ModelSim window for simulation of the counter.

Before beginning the simulation we will illustrate some useful features of the Wave window. For each signal we will first select a specific number-radix for displaying the signal's value, and then assign the signal a customized display name. Right-click on the name of the signal L in the Wave window, as illustrated in Figure 31, and select *Radix* > *Binary*. Use this same procedure to set the radix for Clock to Binary, Data and Q to Unsigned, and E to Binary.

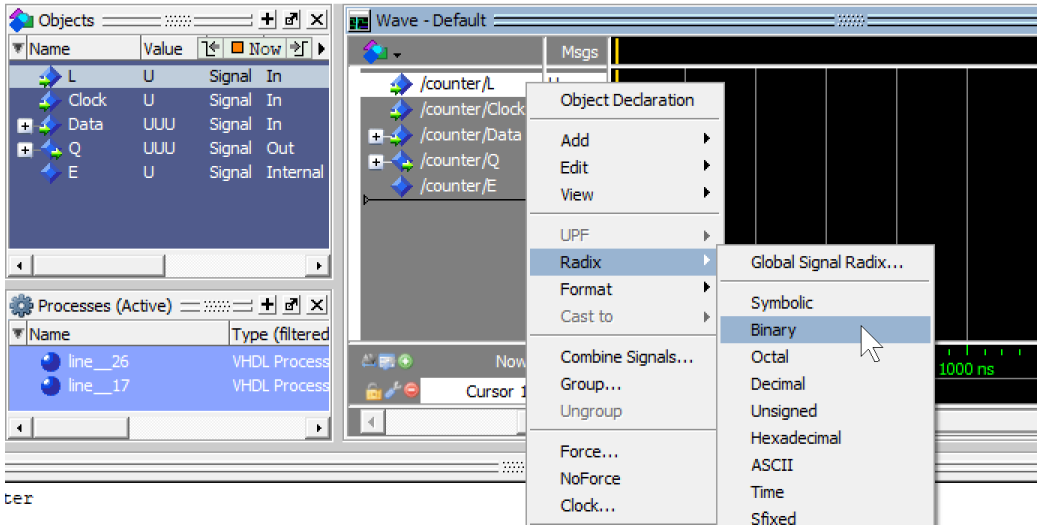

Figure 31. Setting a waveform radix.

Now, as shown in Figure 32, right-click again on L and select Properties to open the dialog in Figure 33. Set the Display Name to Load. Then, use the same process to set the other display names as given in Figure 34.

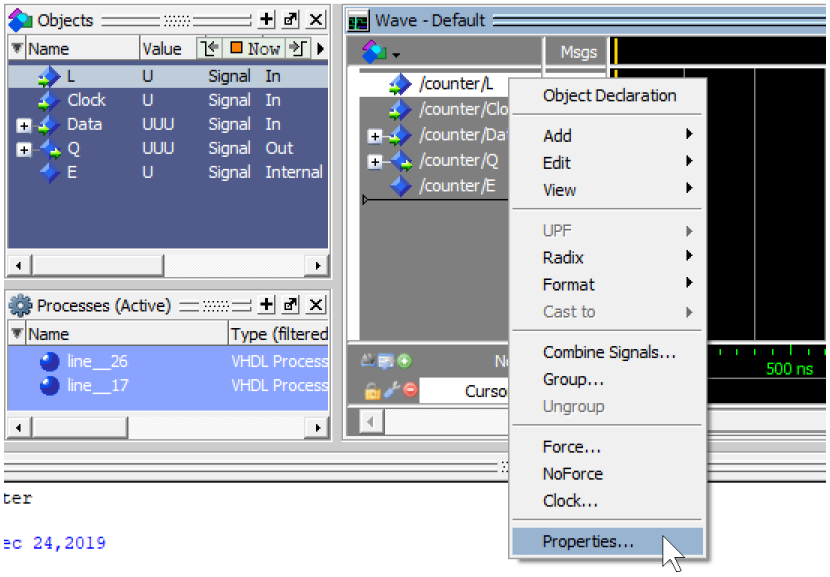

Figure 32. The Properties dialog.

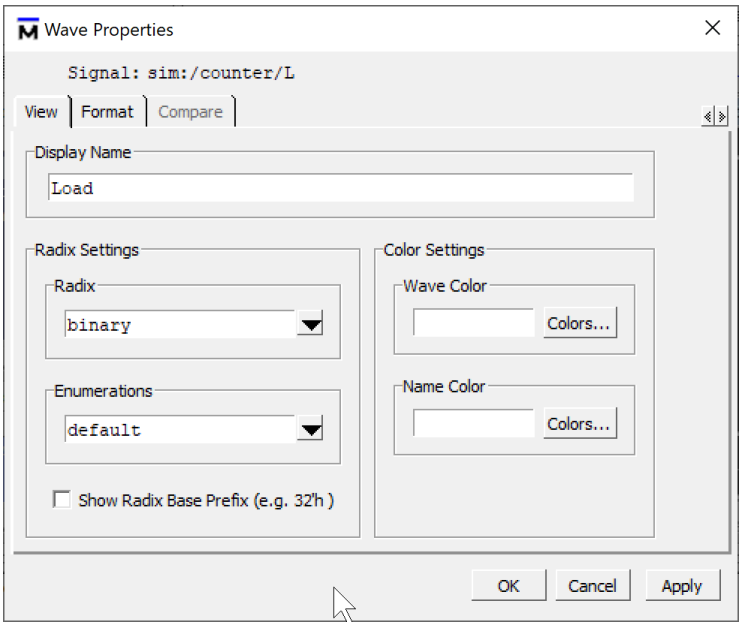

Figure 33. Setting a display name.

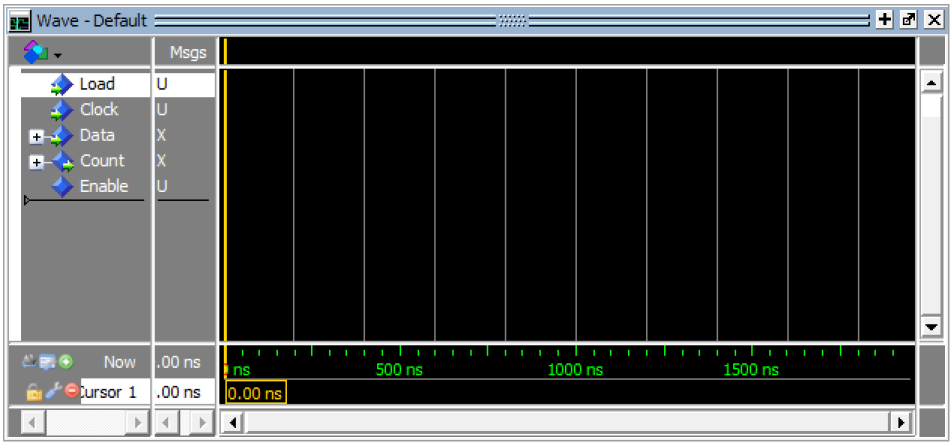

Figure 34. The renamed signals.

For the previous ModelSim project described in this tutorial we employed features of the `Wave` window to *draw* waveforms that were used as inputs to the simulation. For this project we will use a different method by *forcing* the value of signals. Right-click on the *Load* signal, as illustrated in Figure 35, and select Force. This action opens the pop-up window in Figure 36 that displays the orignal signal name *L* and allows its value to be set.

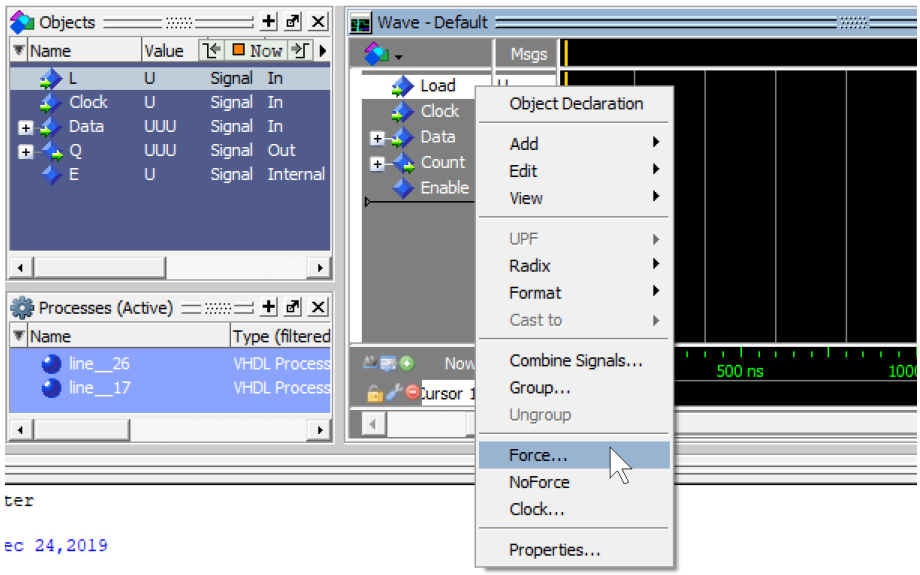

Figure 35. Selecting the Force command.

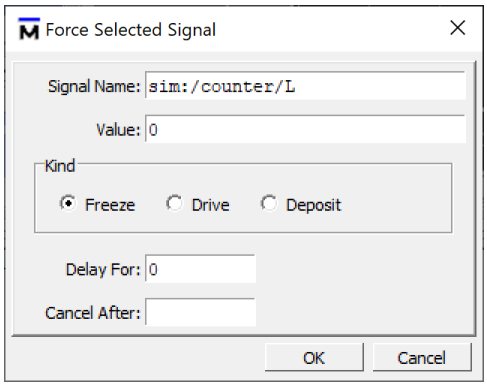

Figure 36. Forcing the *Load* signal to the value 0.

Next, in the `Wave` window right-click on the *Clock* signal and select the `Clock` command to open the pop-up in Figure 37. In the Period box specify a clock-period of 200 ns, as illustrated in the figure, and then click OK. Finally, set the value of the *Data* input signal by right-clicking on its waveform and selecting the *Force* command. As depicted in Figure 38 specify a data value of 10#6. This syntax means that a number base of 10 is being used to specify a signal value of 6.

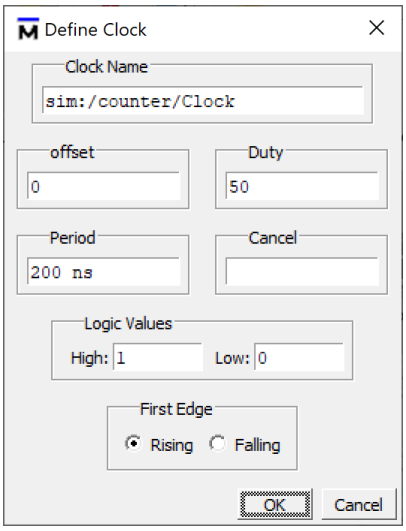

Figure 37. Setting the *Clock* period to 200 ns.

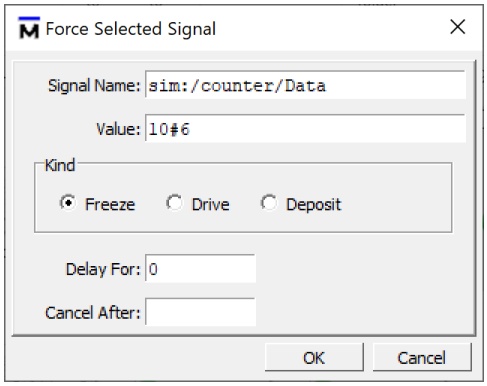

Figure 38. Forcing the value of the *Data* signal..

Since the *Load* signal is asserted to 0 we can now initialize the counter from the *Data* input by running the circuit for one clock cycle. To run the simulation for 200 ns you can use the `Run Length` toolbar box from Figure 25. Alternatively you can enter the command run 200 ns in the `Transcript` window. Perform the simulation to get the result shown in Figure 39.

Now, right-click on the *Load* signal again and select the Force command. In the window from Figure 36 set the *Load* signal to 1, so that it is no longer asserted. Finally, run the simulation for another 1600 ns by using either the `Run Length` toolbar box or by typing run 1600 ns in the `Transcript` window. As shown in Figure 40 the counter decrements on each active clock edge until it reaches 0, after which *Enable = 0* and the counter is stopped.

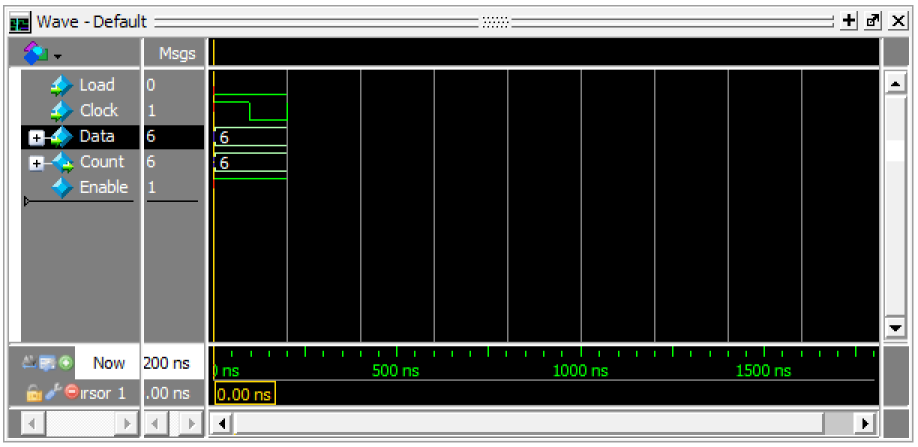

Figure 39. Waveforms after running the simulation for 200 ns.

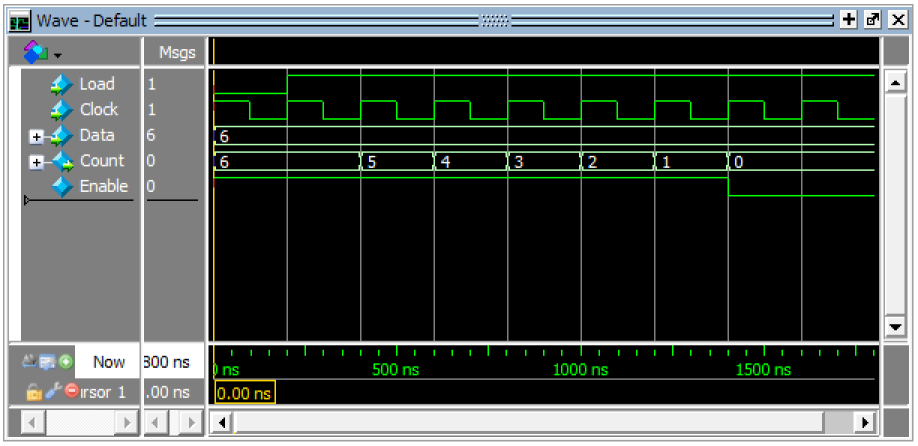

Figure 40. Waveforms after running the full simulation.

You can select the *File* > *Save Format* command to save the formatting information for the waveforms, using a filename such as *wave.do*. If you quit ModelSim and then later reload the *counter* project, then the waveform formats can be reloaded by first starting a new simulation and then entering the command do wave.do in the `Transcript` window.

9 Concluding Remarks

The purpose of this tutorial is to provide a quick introduction to ModelSim, explaining only the rudimentary aspects of functional simulation that can be performed using the ModelSim Graphical User Interface. More details about the ModelSim simulator can be found in other tutorials that are available on the internet.

Copyright © Intel Corporation.