Introduction to Simulation of VHDL Designs

1 Introduction

An effective way of determining the correctness of a logic circuit is to simulate its behavior. This tutorial provides an introduction to such simulation using Altera's University Program Simulation Tools, called *Qsim* and the *Vector Waveform Editor*.

The simulation tools are used as part of the Quartus II CAD system, and they are intended for students who are taking a course in logic circuit design. The tutorial shows how these tools can be used to perform functional simulation of a circuit specified in VHDL. Only a very basic understanding of VHDL is needed for this purpose.

Contents:

The Qsim tool is available for use with Altera's Quartus II software version 10.1 or later. It allows the user to apply inputs to the designed circuit, usually referred to as *test vectors*, and to observe the outputs generated in response. The Qsim tools include a graphical interface for creating the input waveforms.

In this tutorial, the reader will learn about:

- Test vectors needed to test the designed circuit
- Using Qsim to draw test vectors
- Functional simulation, which is used to verify the functional correctness of a synthesized circuit

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

2 Installing the Qsim Tools

To use the Qsim tools, it is first necessary to install the Quartus II CAD system, version 10.1 or later. The Quartus II software is available in the Download Center of Altera's web site. To install the Qsim tools, navigate to Altera's University Program web pages. These pages can be accessed from Altera's home page at *www.altera.com*, by clicking on Training > University Program. On the left side of the page click on Software, and then select University Program Installer. Download this software and install it on your computer.

Figure 1. Altera University Program installer setup.

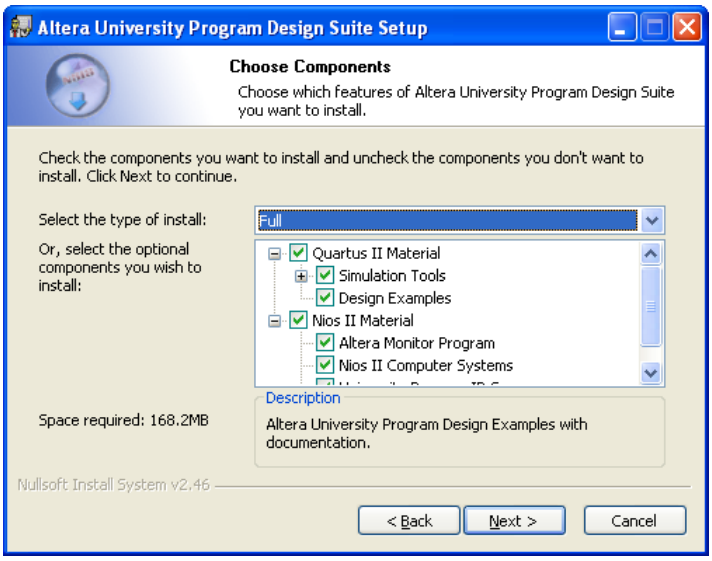

Figure 2. Selecting the Simulations Tools component.

3 Design 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 majority3 IS
    PORT(
        x1, x2, x3 : IN STD_LOGIC;
        f : OUT STD_LOGIC; );
END majority3;
ARCHITECTURE majority3_rtl OF majority3 IS
BEGIN
    f <= (x1 AND x2) OR (x1 AND x3) OR (x2 AND x3);
END majority3_rtl;
```

Enter this code into a file called *majority.vhd*.

The desired circuit has to be first implemented in a Quartus II project. To do so, create a new directory (folder) for the Quartus II project, and for consistence with the description in this tutorial call it *simulator_intro*. Copy the file *majority.vhd* into this directory. Then, create a Quartus II project and call it *majority3*. Compile your design.

Select Start > All Programs > Altera > University Program > Simulation Tools > QSim to open the Qsim tools, which will display the window in Figure 3. In the displayed window select File > Open Project, which leads to the pop-up window in Figure 4. Here, choose the Quartus II project that you created. This is done by selecting the file *majority3.qpf* as shown in the figure. Note that the suffix *.qpf* stands for “quartus project file".

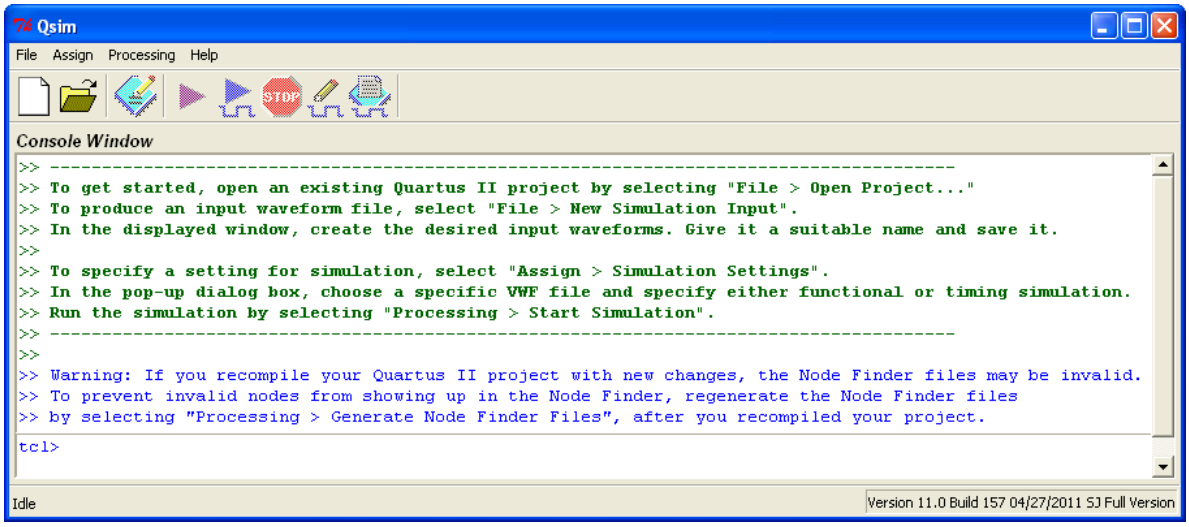

Figure 3. The Qsim window.

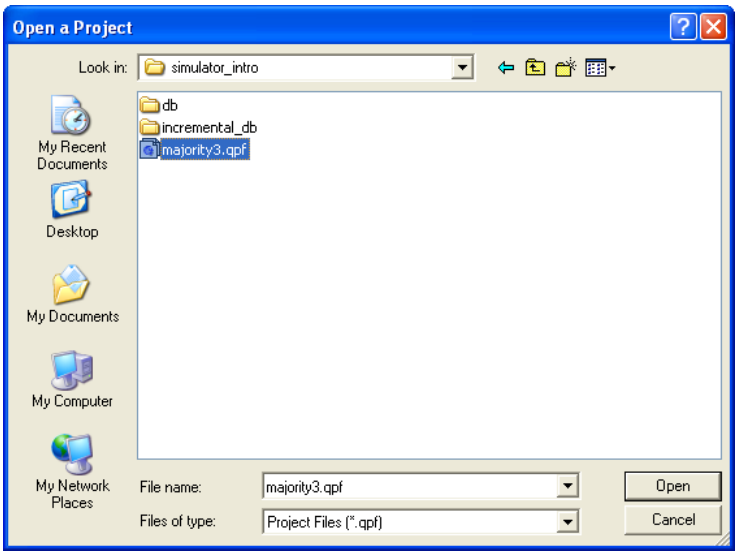

Figure 4. Choosing the existing Quartus II project.

4 Creating Waveforms for Simulation

To create test vectors for your design, select the Qsim command File > New Simulation Input File. This command opens the Waveform Editor tool, shown in Figure 5, which allows you to specify the desired input waveforms.

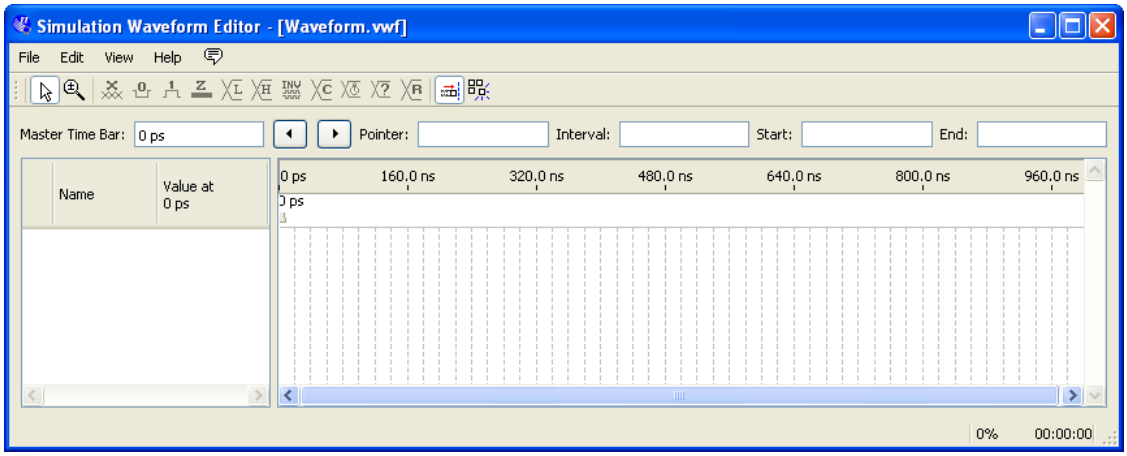

Figure 5. The Waveform Editor window.

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 run the simulation for 800 ns; so, select Edit > Set End Time in the Waveform Editor and in the pop-up window that will appear specify the time of 800 ns, and click OK. This will adjust the time scale in the window of Figure 5.

Before drawing the input waveforms, it is necessary to locate the desired signals in the implemented circuit. In FPGA jargon, the term “node" is used to refer to a signal in a circuit. This could be an input signal (input node), output signal (output node), or an internal signal. For our task, we need to find the input and output nodes. This is done by using a utility program called the Node Finder.

In the Waveform Editor window, select Edit > Insert > Insert Node or Bus. In the pop-up window that appears, which is shown in Figure 6, click on Node Finder.

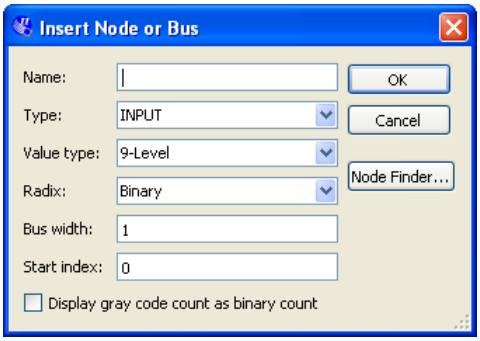

Figure 6. The Insert Node or Bus dialog.

The Node Finder window is presented in Figure 7. A filter is used to identify the nodes of interest. In our circuit, we are only interested in the nodes that appear on the *pins* (i.e. external connections) of the FPGA chip. Hence, the filter setting should be Pins: all. Click on List, which will display the nodes as indicated in the figure. In a large circuit there could be many nodes displayed. We need to select the nodes that we wish to observe in the simulation. This is done by highlighting the desired nodes and clicking on the > button. Select the nodes labeled *x1*, *x2*, *x3*, and *f*, which will lead to the image in Figure 8. Click OK in this window and also upon return to the window in Figure 6. This returns to the Waveform Editor window, with the selected signals included as presented in Figure 9.

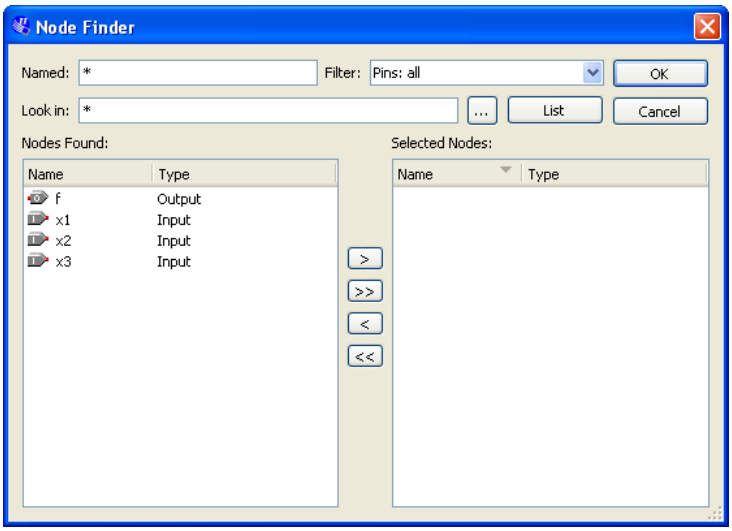

Figure 7. The Node Finder dialog.

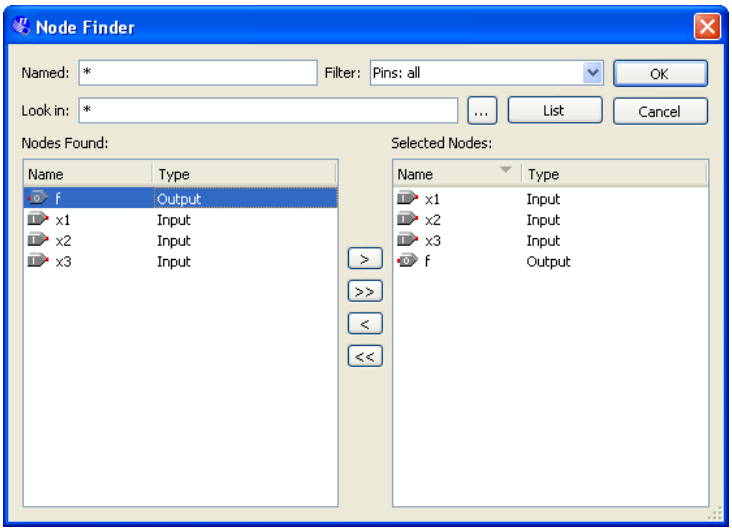

Figure 8. The selected signals.

Observe that in Figure 9 all input signals are at logic level 0. The output, *f* is shown as undefined. Next, we have to draw the input waveforms. Then, we will simulate the circuit, which will produce the output waveform.

To make it easier to draw the input waveforms, the Waveform Editor displays dashed grid lines. The spacing of the grid lines can be adjusted by selecting Edit > Grid Size, and in the pop-up box in Figure 10 specifying the desired size. The spacing of grid lines in Figure 9 is 10 ns. Another convenience in drawing is to have transitions of a waveform snap on grid lines. This feature is activated by clicking on the Snap to Grid icon .

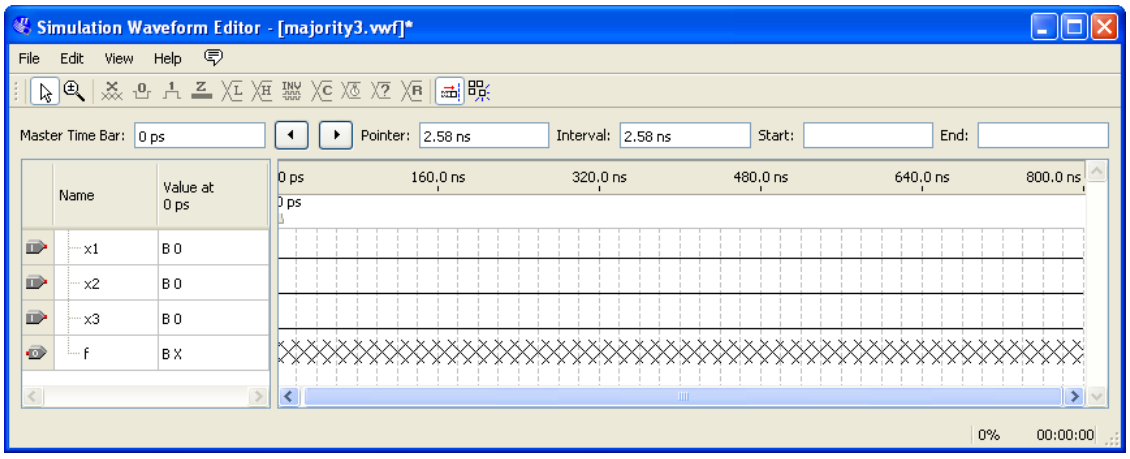

Figure 9. Signals in the Waveform Editor window.

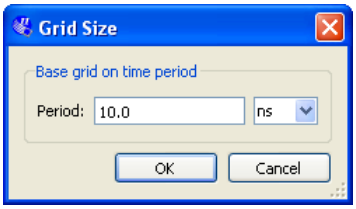

Figure 10. Specifying the grid spacing.

Input waveforms can be drawn in different ways. The most straightforward way is to indicate a specific time range and specify the value of a signal. To illustrate this approach, click the mouse on the *x1* waveform near the 400-ns point and then drag the mouse to the 800-ns point. The selected time interval will be highlighted in blue, as depicted in Figure 11. Change the value of the waveform to 1 by clicking on the Forcing High (1) icon , as illustrated in Figure 12.

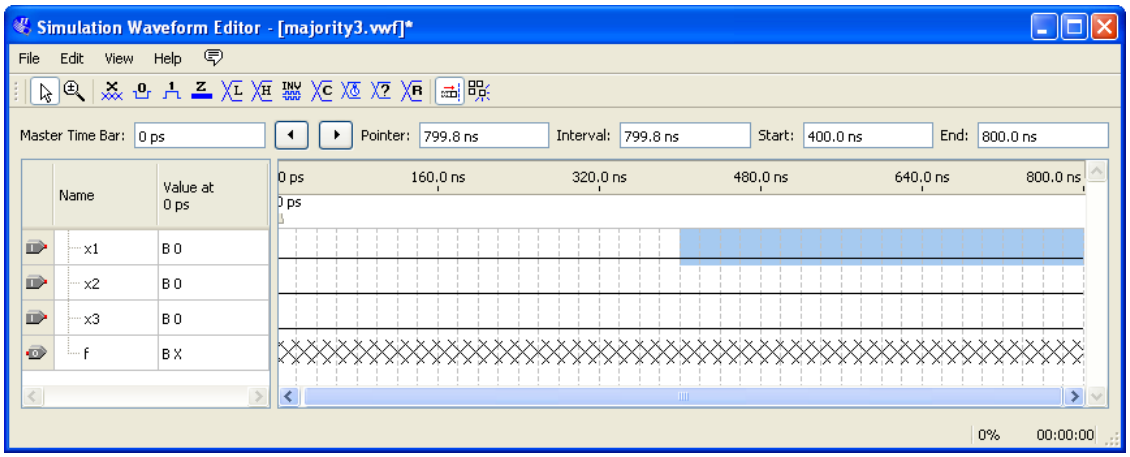

Figure 11. Selection of a time interval.

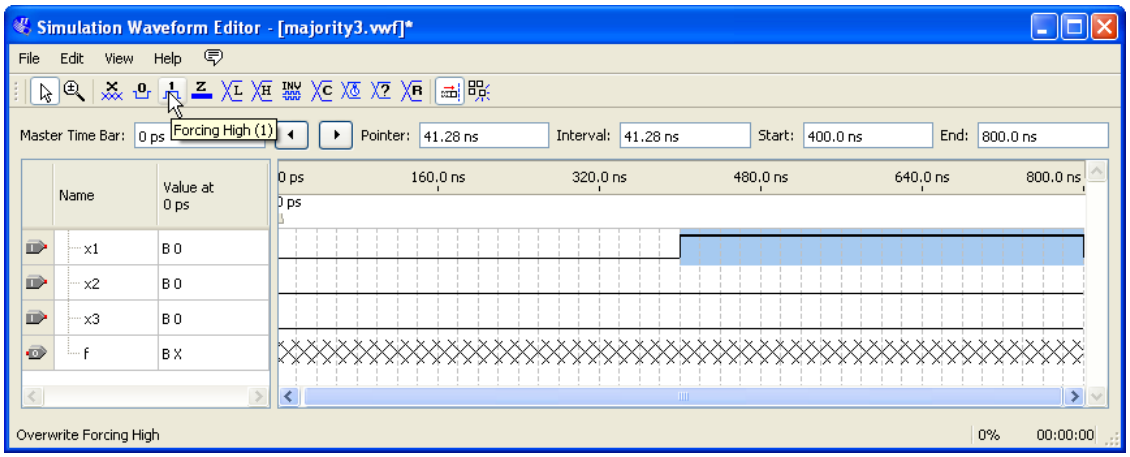

Figure 12. Drawing the waveform for *x1*

In creating the waveform for *x1*, we used the icon to implement the logic value 1. Another possibility is to invert the value of the signal in a selected time interval by using the Invert icon . We will use this approach to create the waveform for *x2*, which should change from 0 to 1 at 200 ns, then back to 0 at 400 ns, and again to 1 at 600 ns. Select the interval from 200 to 400 ns and click on the icon, as illustrated in Figure 13. Then do the same for the interval from 600 to 800 ns.

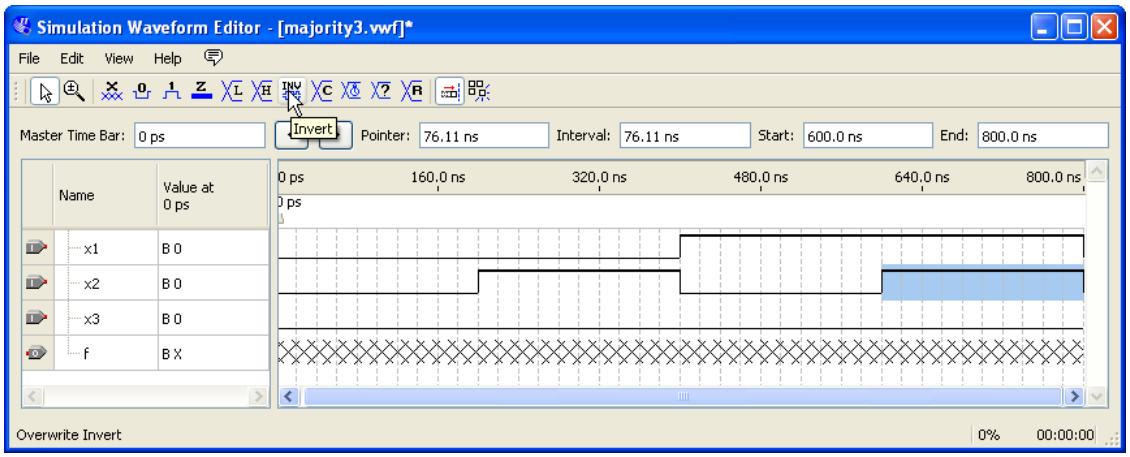

Figure 13. Drawing the waveform for *x2*.

We will use a third approach to draw the waveform for *x3*. This signal should alternate between logic values 0 and 1 at each 100-ns interval. Such a regular pattern is indicative of a *clock* signal that is used in many logic circuits. Even though there is no clock signal in our example circuit, it is convenient to specify *x3* in this manner. Click on the *x3* input, which selects the entire 800-ns interval. Then, click on the Overwrite Clock icon , as indicated in Figure 14. This leads to the pop-up window in Figure 15. Specify the clock period of 200 ns and the duty cycle of 50%, and click OK. The result is depicted in Figure 16.

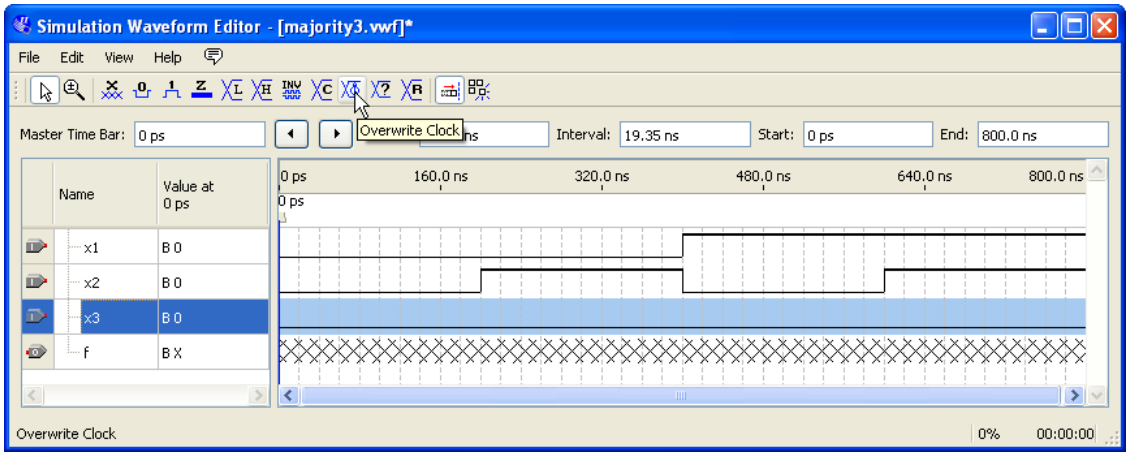

Figure 14. Drawing the waveform for *x3*.

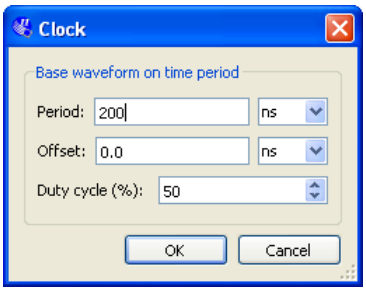

Figure 15. Defining the clock characteristics

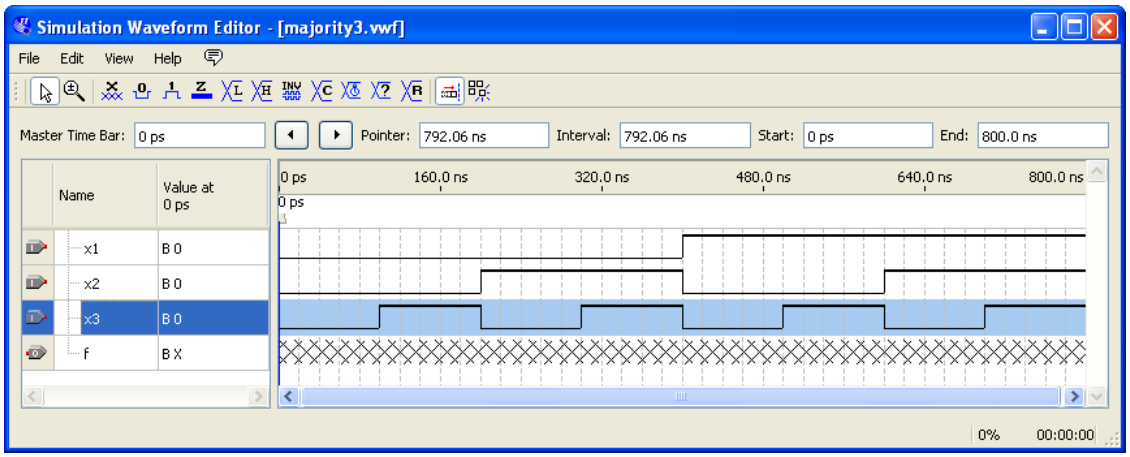

Figure 16. The completed input waveforms.

Save the waveform file using a suitable name; we chose the name *majority3.vwf*. Note that the suffix *vwf* stands for *vector waveform file*.

5 Simulation

Return to the Qsim window (in Figure 3). Begin by clicking on Assign > Simulation Settings... and specify the path to the *vwf* file you just created. Then, select Function as the Simulation Type.

To enable the functional simulation to be performed, it is necessary to generate a *functional netlist* of the circuit. This netlist specifies the logic elements and the connections needed to implement the circuit. Select Processing > Generate Functional Simulation Netlist, or click on the icon  .

Now, we can simulate the circuit. Select Processing > Start Simulation, or click on the icon  . A pop-up window will indicate that “simulator was successful". Click OK. Another pop-up window will state that “the file is read-only and cannot be edited". This states that the output of the simulation is a file that you cannot alter. Any

changes in simulation have to be done by modifying the *majority3.vwf* file and resimulating the circuit. Click OK. Qsim will now display the waveforms produced in the simulation process, which are depicted in Figure 17. 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.

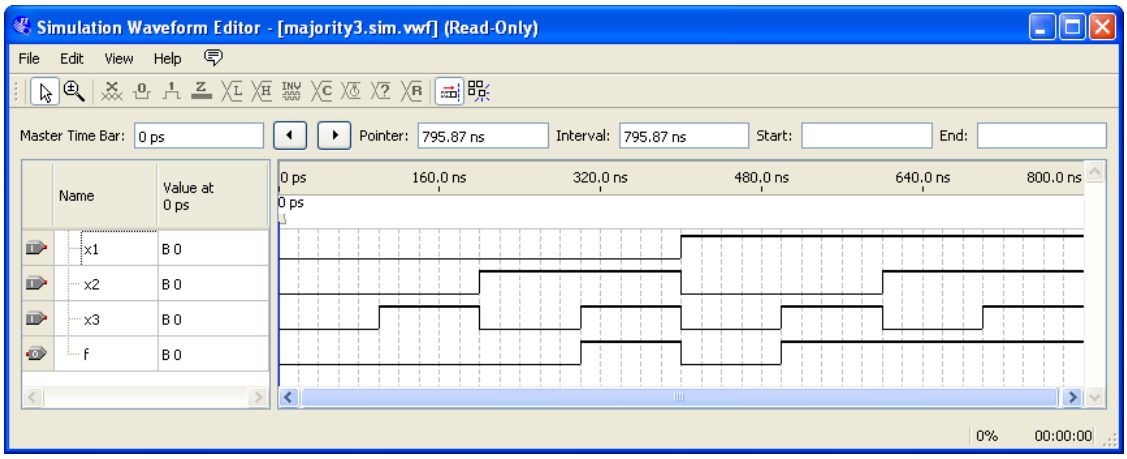

Figure 17. Result of the simulation.

6 Making Changes and Resimulating

Changes in the input waveforms can be made using the approaches explained above. The circuit can then be resimulated using the altered waveforms. For example, change the waveform for x_1 to have the logic value 1 in the interval from 100 to 240 ns, as indicated in Figure 18. Now, simulate the circuit again. The result is given in Figure 19. If errors in the circuit are discovered, then these errors can be fixed by changing the VHDL code and recompiling the design using the Quartus II software. Qsim can then be used to open again the *majority3.qpf* file and resimulate the corrected design.

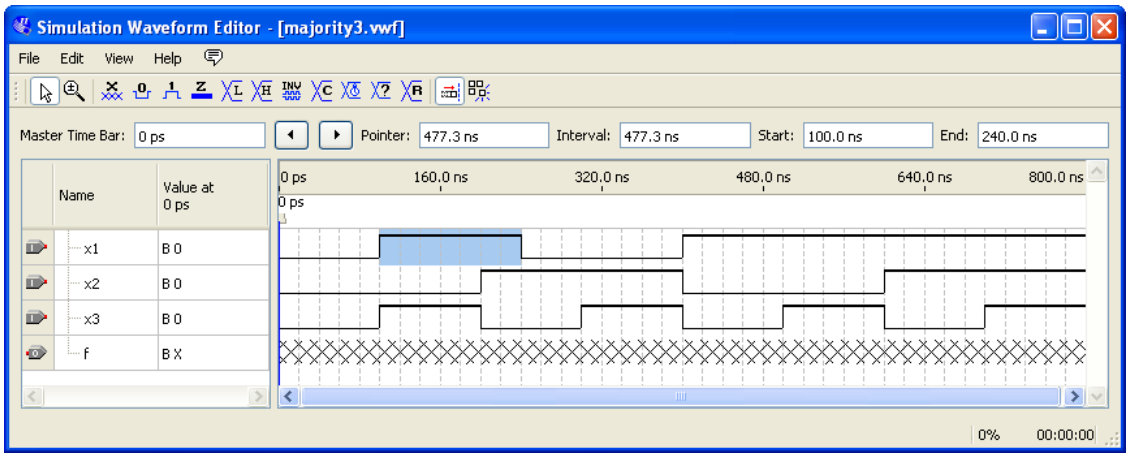

Figure 18. Modified input waveforms.

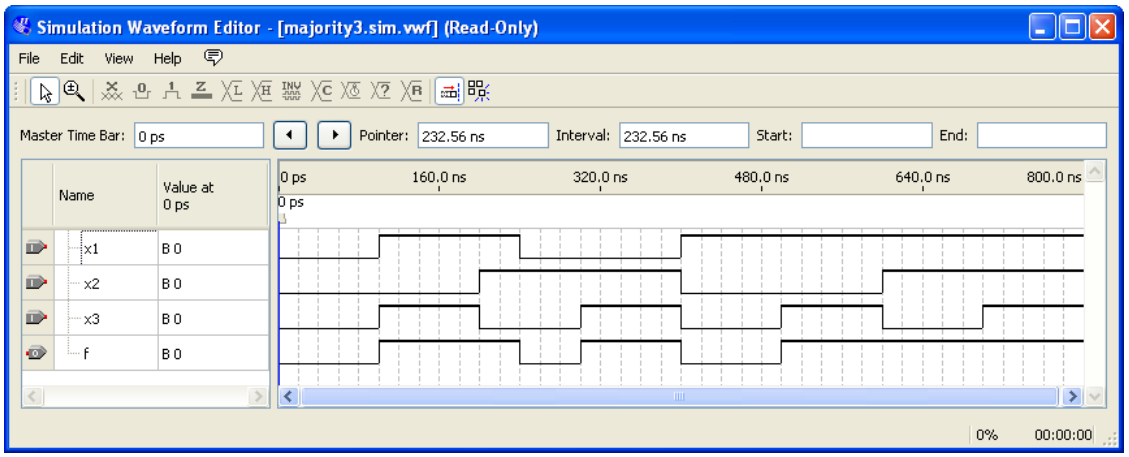

Figure 19. Result of the new simulation.

7 Concluding Remarks

The purpose of this tutorial is to provide a quick introduction to the Altera University Program Simulator, explaining only the rudimentary aspects of functional simulation. A follow-on tutorial is available that shows how to perform timing simulation.

A Simulation Waveform Editor

In section 4 we introduce the Waveform Editor tool, which is used to view and edit waveforms that are used in simulation. Additional features of the Waveform Editor are described in this appendix.

A.1 Waveform Editor Toolbar

The Waveform Editor window is illustrated in Figure 1. The tool includes several commands which can be accessed by using the mouse, including File, Edit, View, and Help. Below these commands, as shown in the figure, there is a toolbar that contains a number of icons which are useful when manipulating waveforms. This toolbar should be visible by default, but if it is not visible, then right-click near the top of the window (below the title bar) and select Waveform Editor in the dialogue that opens.

The toolbar icons are described below.

Selection Tool

This tool is used to select waveform intervals and apply changes. To make a selection, click on any part of a waveform and drag the blue box across the desired interval. It's possible to select multiple waveforms at the same time, as shown in Figure 1, or select entire waveform(s) by clicking on its name(s).

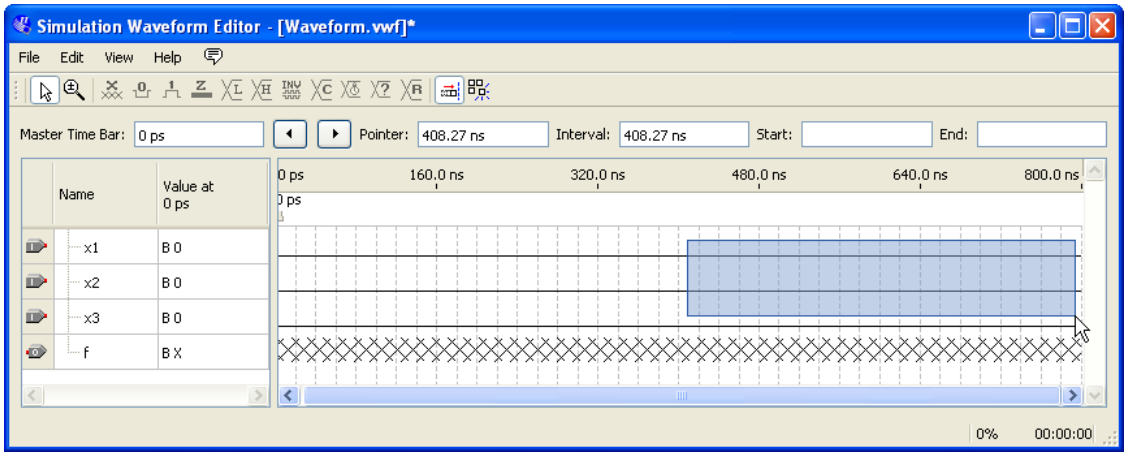

Figure 1. Using the Selection Tool to select a portion of multiple waveforms.

Double clicking the selection tool anywhere on a waveform will select the largest interval with the same value from where the cursor points. Double clicking on a selected interval with different values in it brings up the option to set arbitrary values for that interval.

Zoom Tool

This tool is used to zoom in or zoom out in the waveform display, as indicated in Figure 2. Left-clicking zooms into the display and right-clicking zooms out.

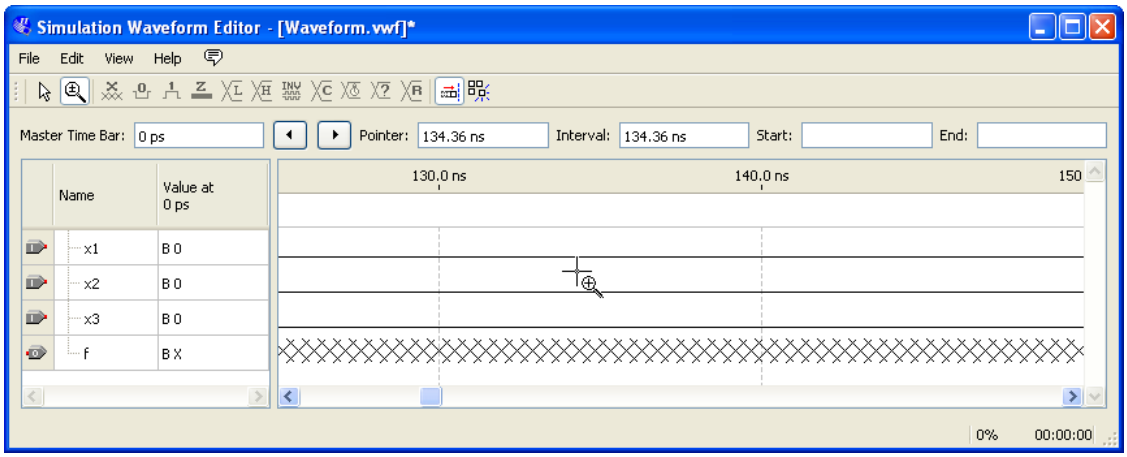

Figure 2. Using the Zoom Tool.

Forcing Unknown(X)

This tool allows the selected part of a waveform to be set to the value Unknown (x). An example is given in Figure 3, using the majority function circuit that was described in section **??**. The value of the signal *x3* has been set to unknown for the first half of the simulation. Running the simulation with these input values results in the output waveform *f* that is shown in the figure. Note that *f* has the value U, which indicates an unknown value for the output.

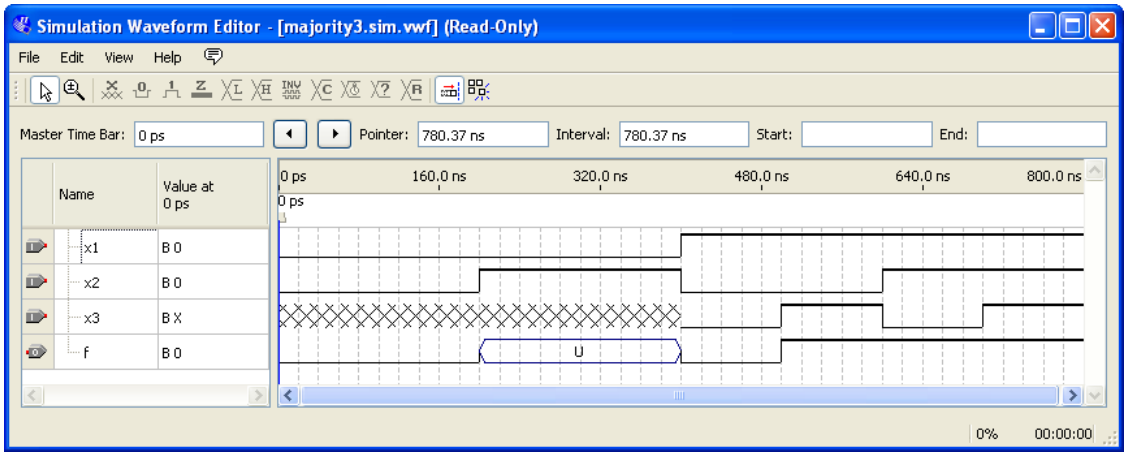

Figure 3. Setting the value of an input to Unknown (X).

Forcing Low (0) and Forcing High(1)

These tools are used to force the selected part of a waveform to the value low (0) or high (1), as shown in Figures 4 and 5, respectively.

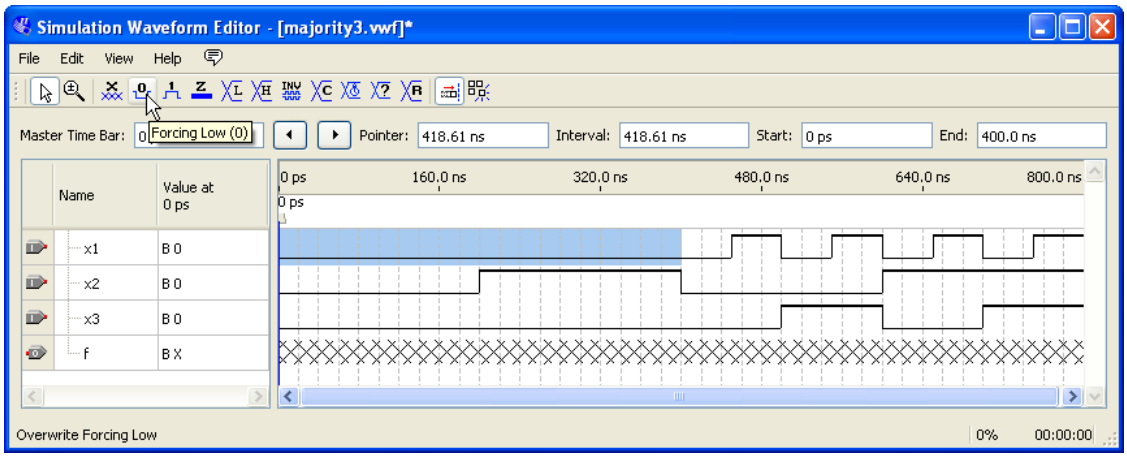

Figure 4. Forcing x_1 to be low from 0 to 400 ns.

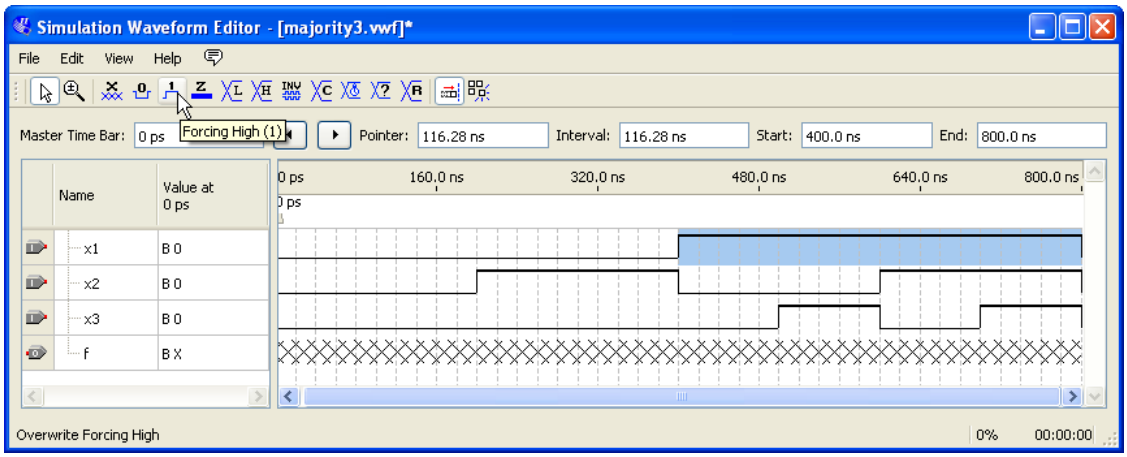

Figure 5. Forcing x_1 to be high from 400 to 800 ns.

High Impedance (Z)

This tool forces the selected waveform to the value High Impedance (Z), as shown in Figure 6. The high impedance value represents a signal that has not been set to any specific value—that is, an input pin that is not connected. Forcing output waveforms to have high impedance does not affect the output simulation waveforms.

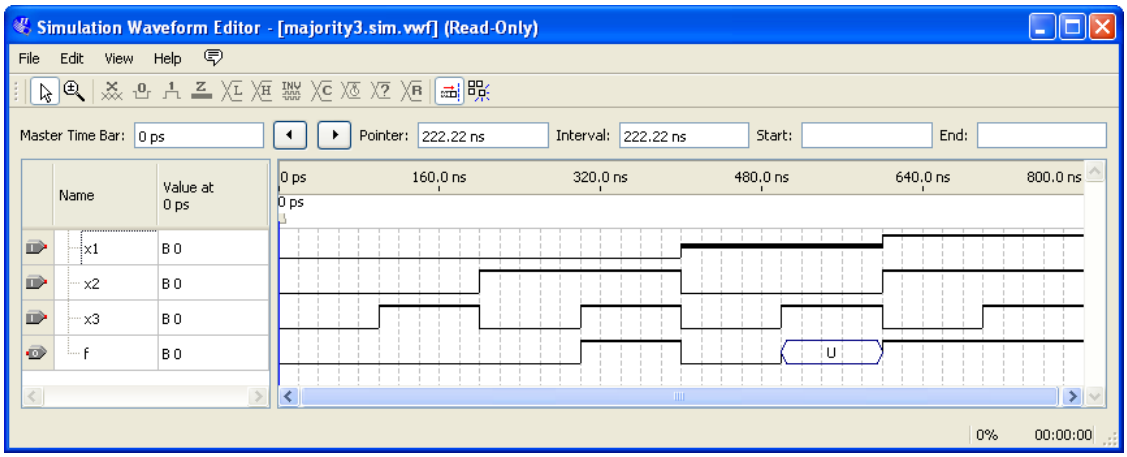

Figure 6. Setting a signal to high impedance.

Weak Low (L) and Weak High (H)

These tools are used to set a signal to the values Weak Low (L) or Weak High, which represents a circuit in which a bidirectional signal is pulled down or up by using a resistor. Examples are shown in Figures 7 and 8.

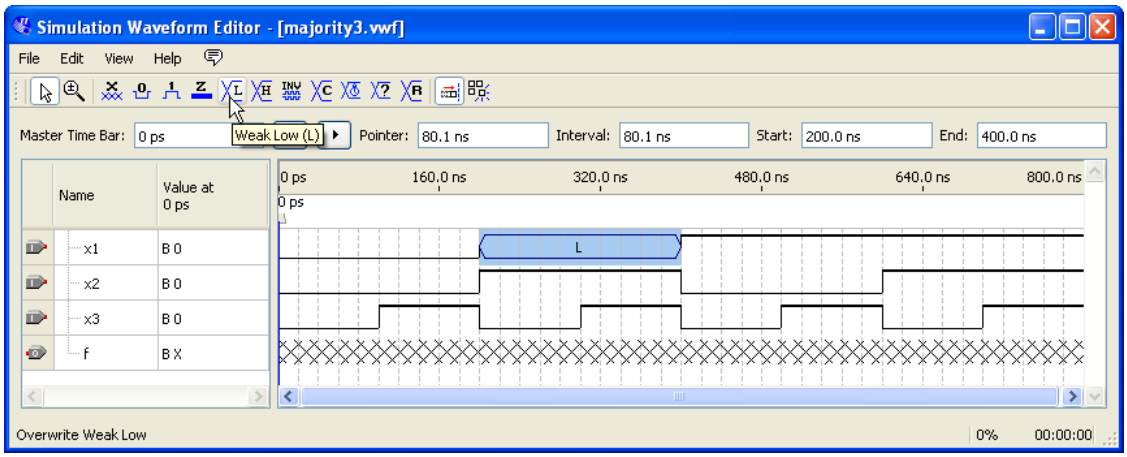

Figure 7. Changing the *x1* signal to be weak low from 200 to 400 ns.

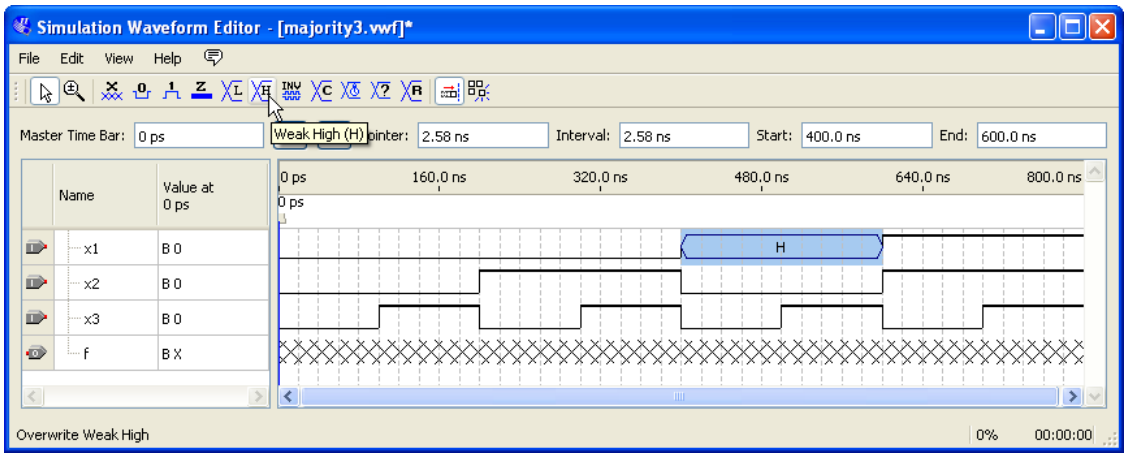

Figure 8. Changing the *x1* signal to be weak high from 400 to 600 ns.

Invert

This tool inverts the value of a selected waveform, as shown in Figure 9. Low signals become high, weak low signals become weak high, and vice versa for both cases. The Invert tool has no effect on a signal that is set to high impedance or unknown.

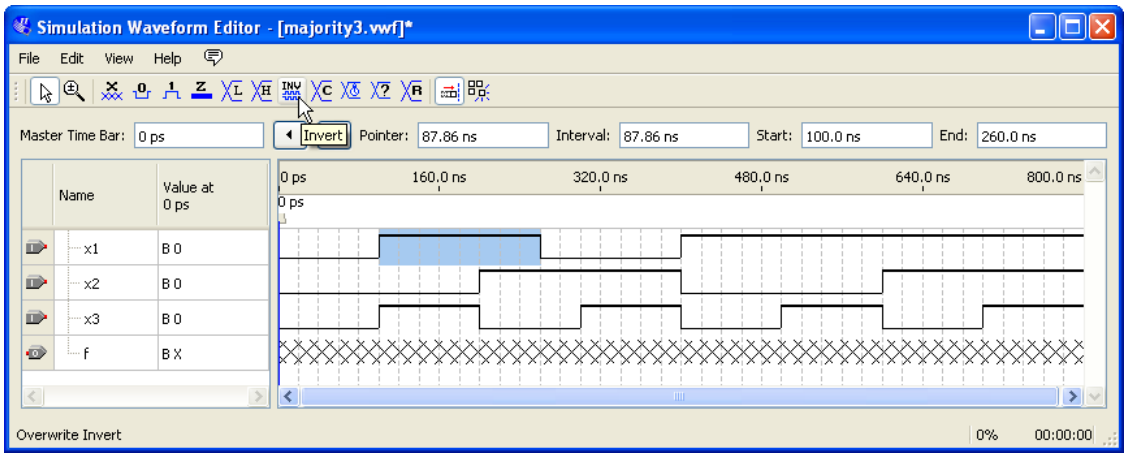

Figure 9. Inverting the *x1* signal from 100 to 260 ns.

Count Value

This tool allows a waveform to be partitioned into sections, in which the value is incremented by a specified amount. The Count Value tool can only be applied to a single waveform or a grouped waveform (see section B.1). The options that are available when using the Count Tool are illustrated in Figure 10.

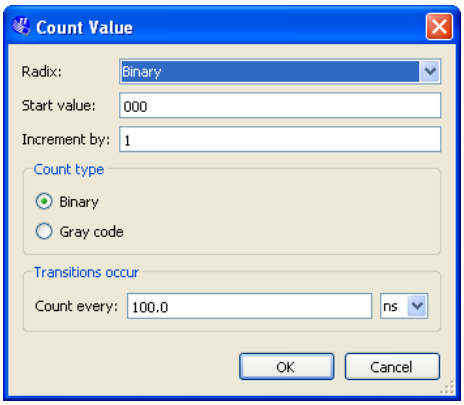

Figure 10. Options available for for the Count Value tool.

As an example, Figure 11 shows the 3-bit input signal called *count* set to increment by one every 100 ns.

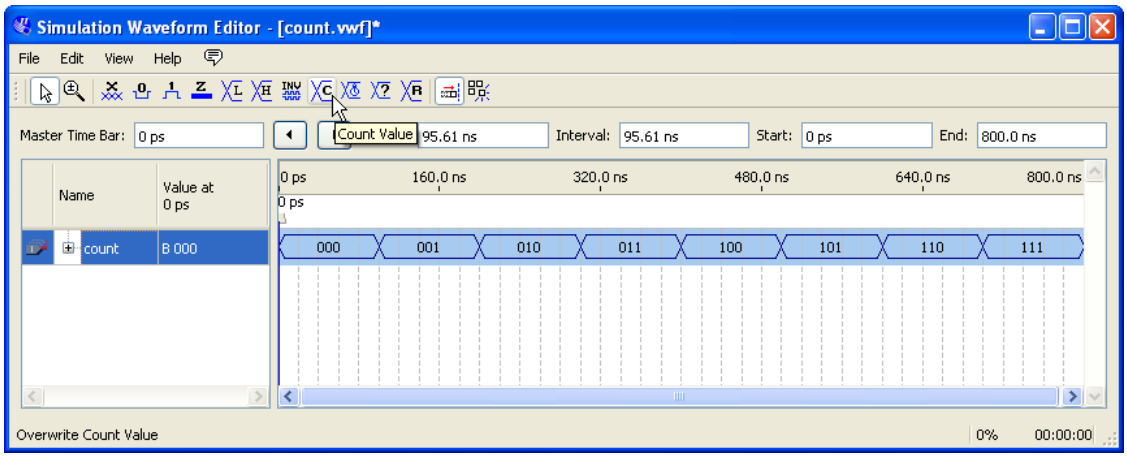

Figure 11. An example of using the Count Value tool.

Overwrite Clock

This tool is used to generate a periodic waveform, which is often used as a clock signal. The options available when using the Overwrite Clock tool are shown in Figure 12.

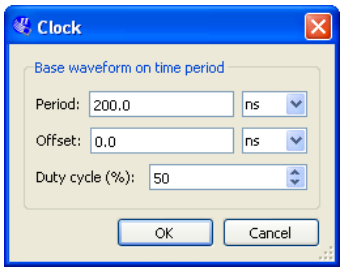

Figure 12. Options available for the Overwrite Clock tool.

In the example of Figure 13, the x_3 signal has been generated with a period of 200 ns, an offset of 0 ns, and a duty cycle of 50%.

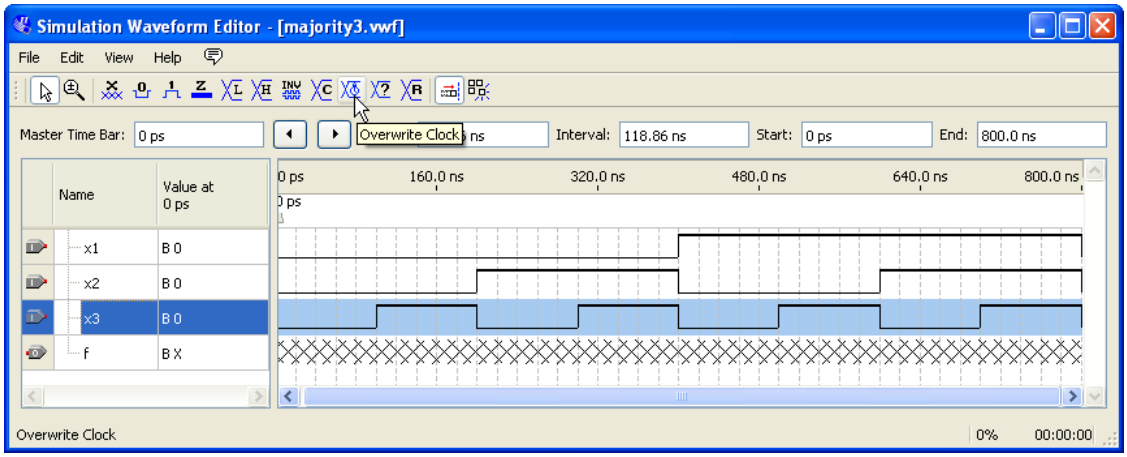

Figure 13. An exmaple of using the Overwrite Clock tool.

Arbitrary Value

This tool allows a signal to be set to an arbitrary value, which is particularly useful for specifying the value of a multibit waveform. The options available when using the Arbitrary Value tool are shown in Figure 14.

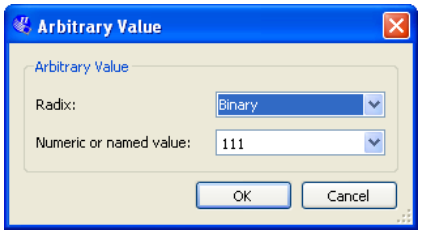

Figure 14. Options available for for the Arbitrary Value tool.

As an example, in Figure 15 the *count* signal is set to three different arbitrary binary values as specified by the user.

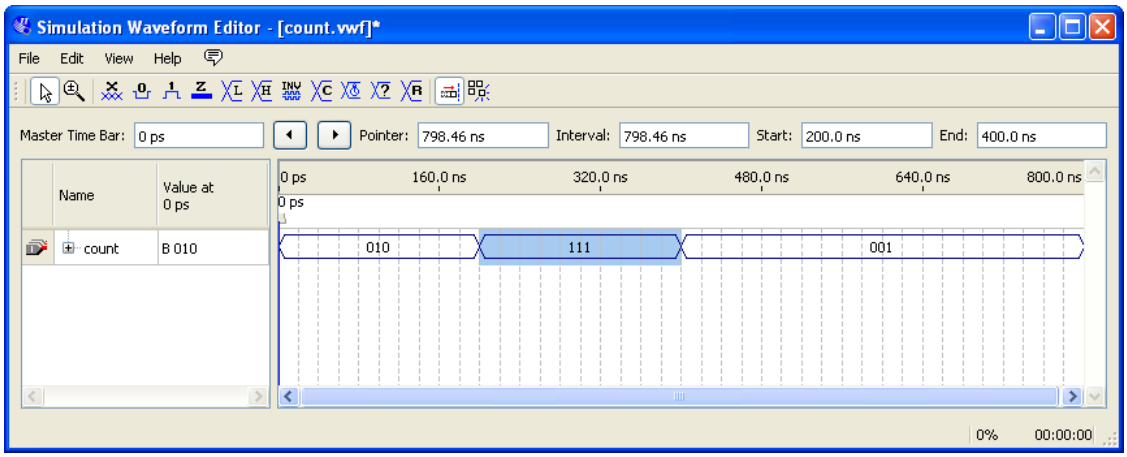

Figure 15. The Arbitrary Value tool is used to set values for the *count* signal.

Random Values

This tool assigns random values to the selected waveform(s), with several options as shown in Figure 16.

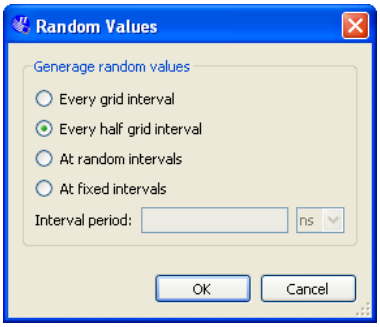

Figure 16. Various options available for the Random Value tool.

For example, in Figure 17, the signal x_1 has been given random values.

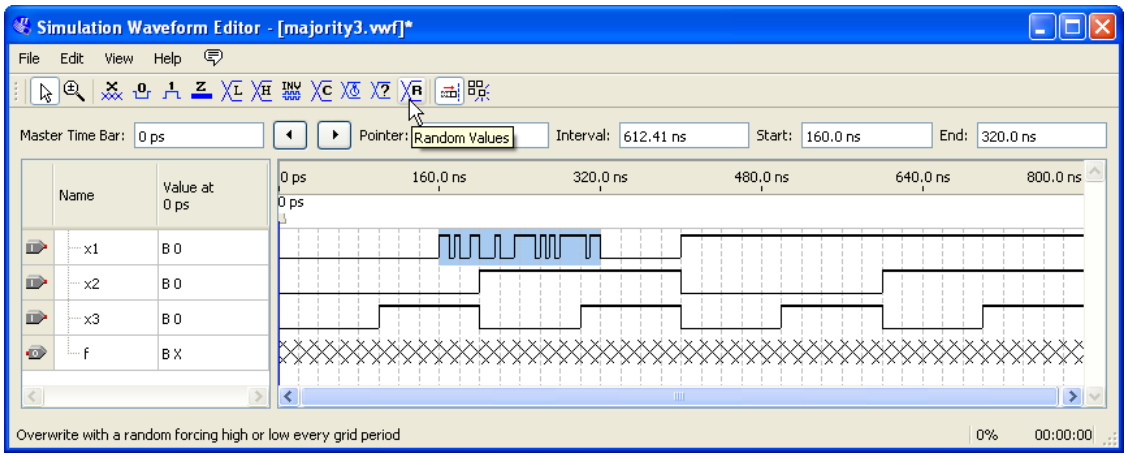

Figure 17. An example of the Random Value tool being used.

Snap to Grid

This option allows selections made with the Selection Tool to snap to the light grey grid lines running vertically down the waveform display. This option can be toggled on and off by pressing the Snap to Grid button. It is set to on by default. Figure 18 shows an example of the Selection Tool being used with the Snap to Grid option turned off.

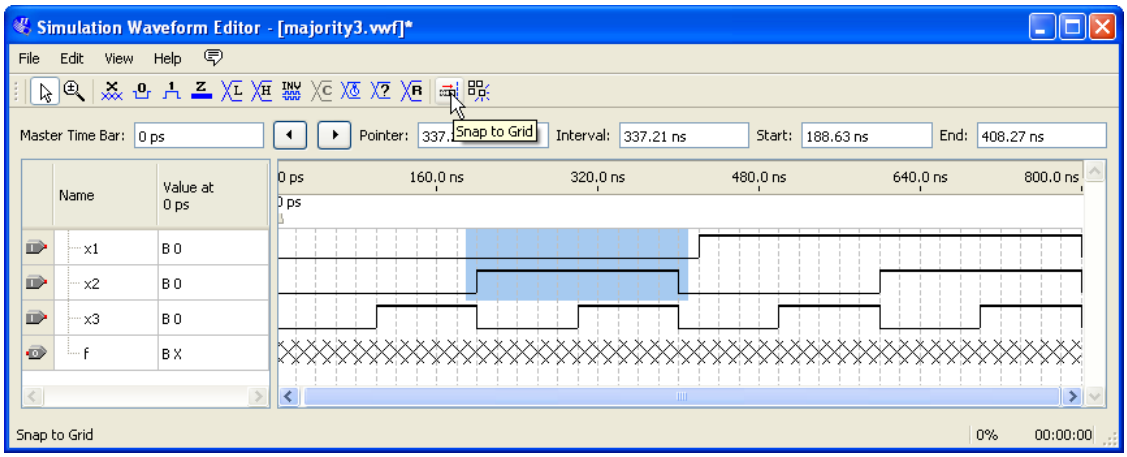

Figure 18. An example of the Snap to Grid option turned off.

Snap to Transition

This option allows the Selection Tool to automatically extend a selection to the first transition encountered on both sides of the selection of one or more waveforms. For example, with the Snap to Transition option turned on, the Selection Tool rectangle shown in Figure 19 would be expanded to create the selections illustrated in Figure 20. This option can be toggled on and off by pressing the Snap to Transition button, and is set to off by default.

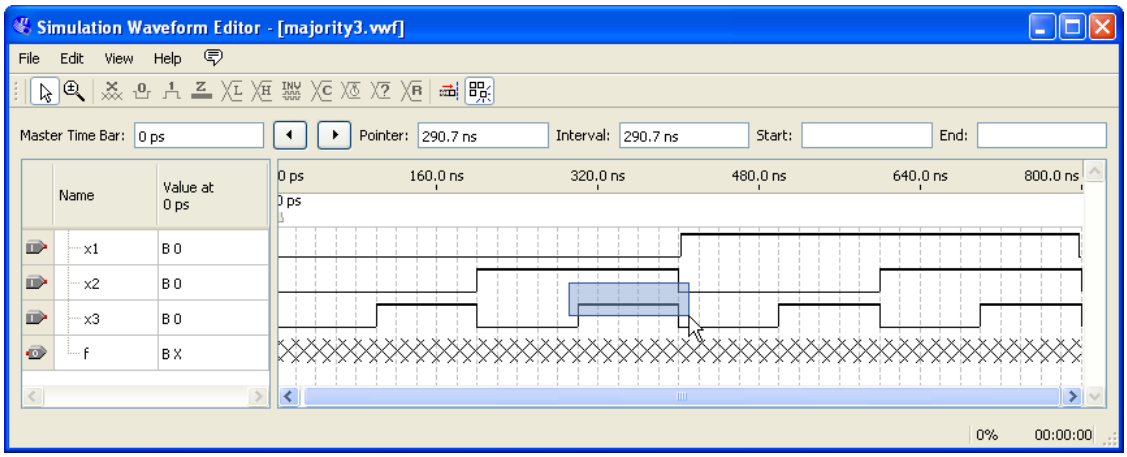

Figure 19. Making a selection with the Snap to Transition option enabled.

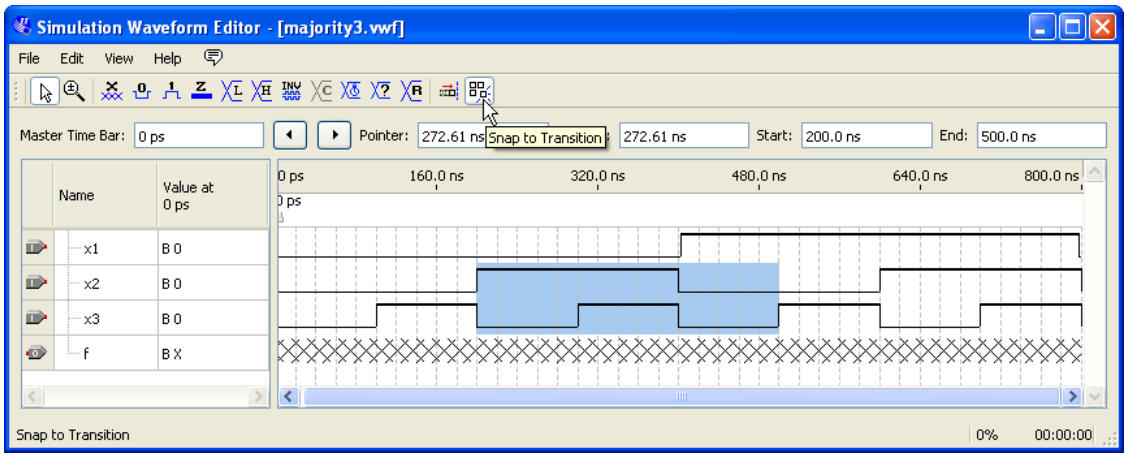

Figure 20. The expanded selection resulting from Figure 19.

B Using Multibit Signals

This section describes features of the Simulation Waveform Editor that are useful for dealing with multibit signals.

B.1 Grouping and Ungrouping Signals

Individual signals can be grouped together to create a multibit waveform. This is done by first selecting the desired waveforms by clicking on their names in the leftside of the Waveform Editor as indicated in Figure 21. Then, as shown in the figure, the grouping of signals is done by right-clicking on the selection and choosing Grouping > Group....

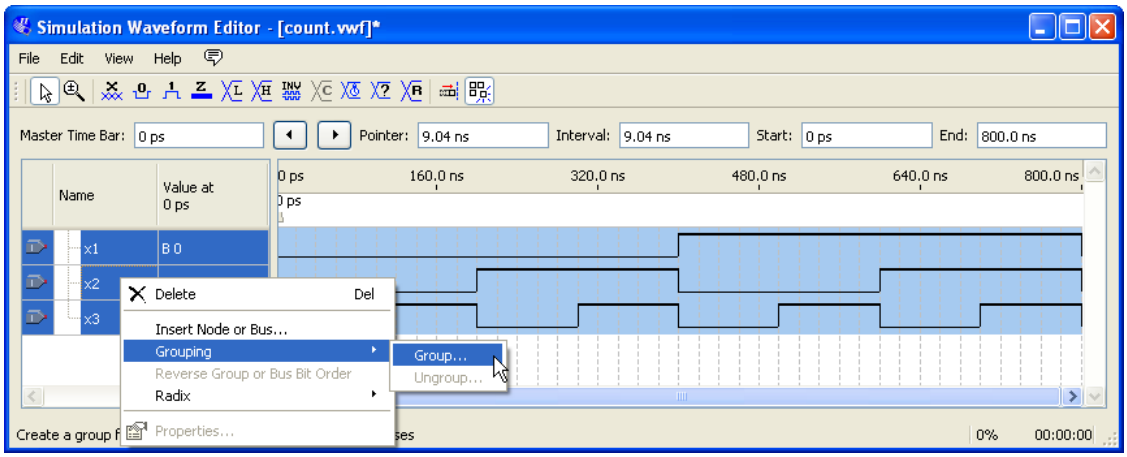

Figure 21. An example of grouping signals.

In the options dialogue that opens, illustrated in Figure 22, a name must be assigned to the group, as well as a radix. In the example shown, the name *count* has been chosen with a binary radix.

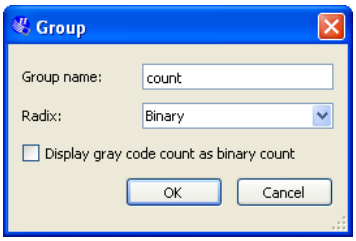

Figure 22. Select a name and radix for the group of signals.

The resulting group of signal is shown in Figure 23. The multibit waveform can be expanded in the waveform editor display by clicking on the + symbol beside the group name. The - symbol is used to collapse the list.

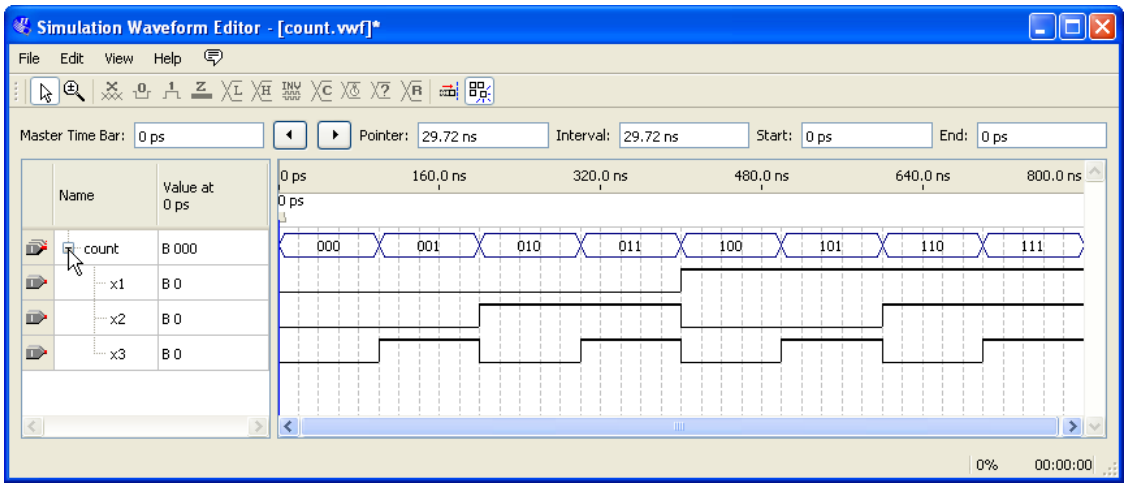

Figure 23. An example of expanding a multibit signal.

A multibit signal can be ungrouped by right-clicking on a previously grouped signals and selecting Grouping > Ungroup....

It is also possible to create hierarchical goupings of signals as illustrated in Figure 24. In this example, the two bit signal called *level2* is combined with the signal called x_3 to create the three bit signal called *level1*. It is only possible to group and ungroup top-level signals.

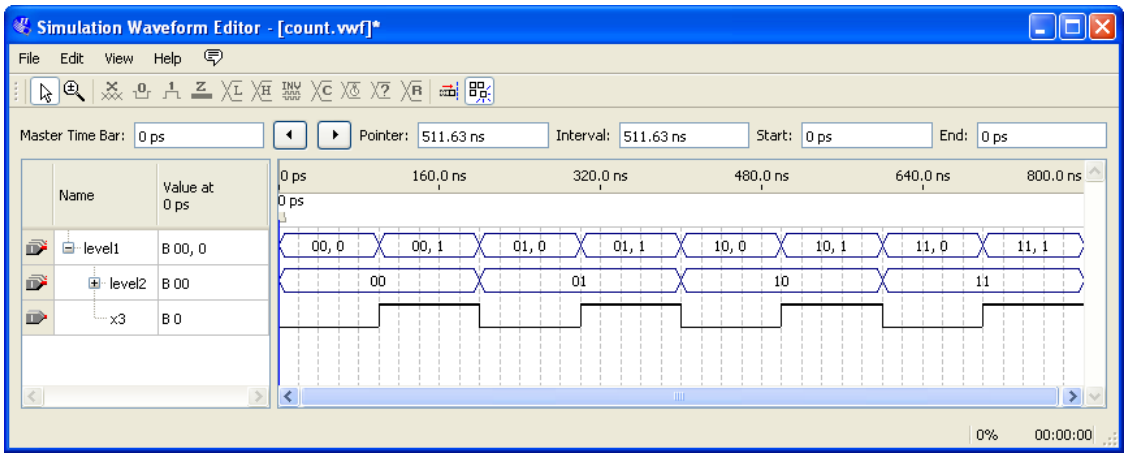

Figure 24. An example of hierarchical groups.

It is also possible to group input and output signals, as shown in Figure 25.

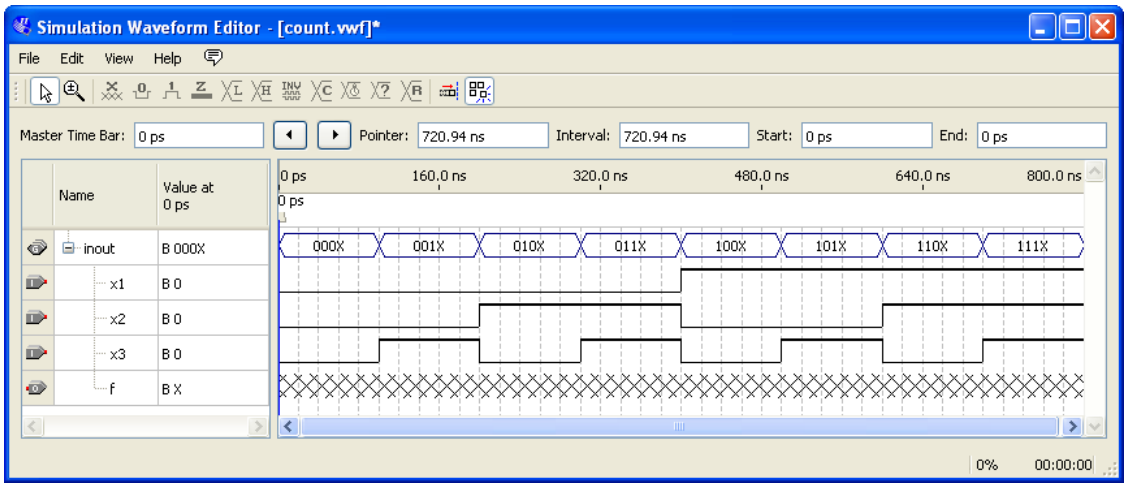

Figure 25. An example of grouping input and output signals.

B.2 Reverse Group or Bus Bit Order

In Figure 23, the three bit signal count is displayed as the 3-tuple $x_1x_2x_3$. It is possible to reverse the order in which the bits are displayed as illustrated in Figure 26. This is done by right-clicking on the name of the multibit signal and selecting Reverse Group or Bus Bit Order, as seen in the figure.

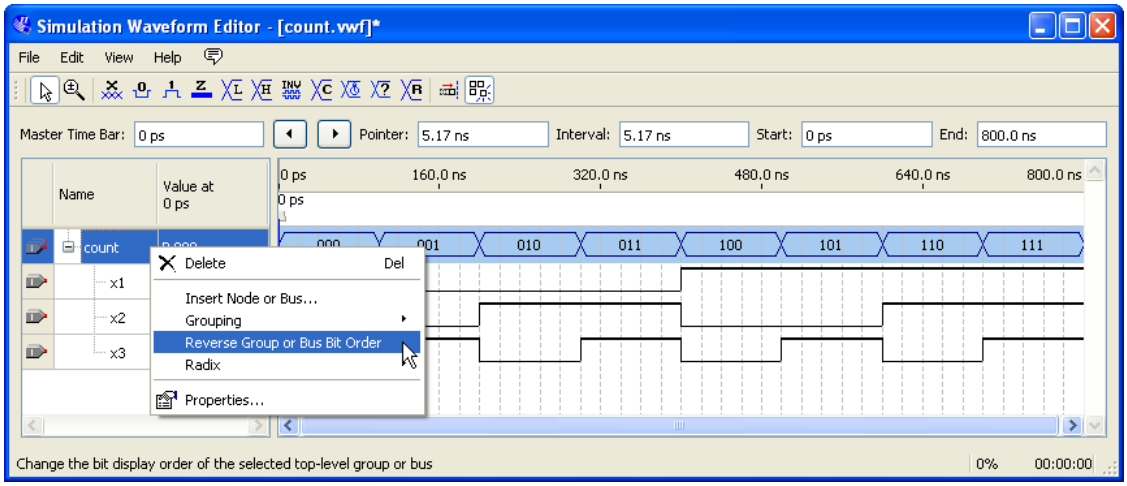

Figure 26. Reversing the bit order on a group of signals.

The effects of the bit reversal can be seen in Figure 27. The count waveform is now displayed as the 3-tuple $x_3x_2x_1$.

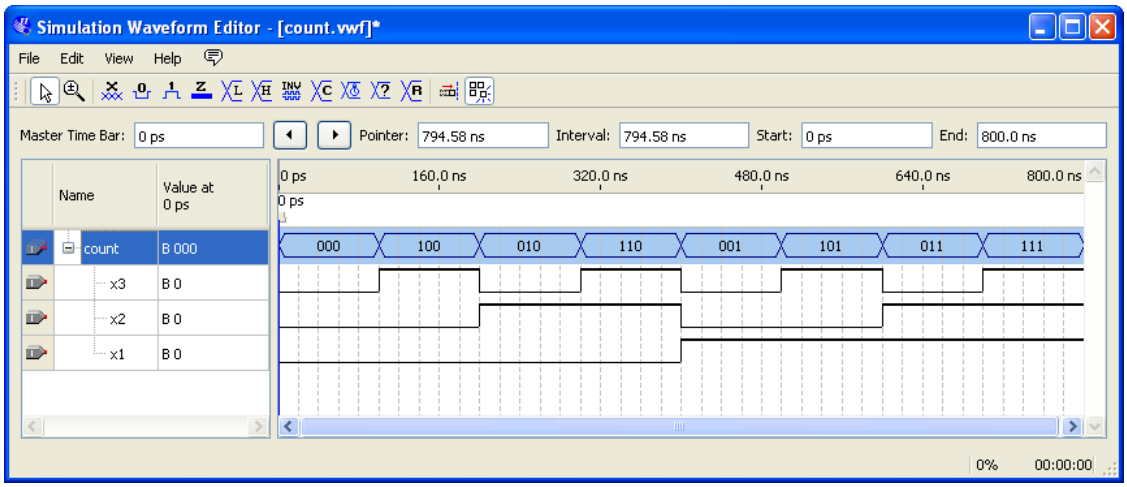

Figure 27. The result of reversing the bit order in Figure 26.

C Displaying the Values of Finite State Machine (FSM) State Variables

If a circuit includes a Finite State Machine (FSM), then it is possible to display the values of the corresponding state variables in the Simulation Waveform Editor. As an example, consider the fragment of VHDL code shown in Figure 28. This code includes the definition of an FSM whose state variables are represented by the two-bit signal named *Tstep_Q*.

```
LIBRARY ieee; USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;

ENTITY proc IS
   PORT (  DIN                 : IN       STD_LOGIC_VECTOR(15 DOWNTO 0);
           Resetn, Clock, Run  : IN       STD_LOGIC;
           Done                : BUFFER   STD_LOGIC;
           BusWires            : BUFFER   STD_LOGIC_VECTOR(15 DOWNTO 0));
END proc;

ARCHITECTURE Behavior OF proc IS
   … declare components
   … declare signals
   TYPE State_type IS (T0, T1, T2, T3);
   SIGNAL Tstep_Q, Tstep_D: State_type;
   …

   BEGIN
      CASE Tstep_Q IS
         WHEN T0 => IF(Run = '0') THEN Tstep_D <= T0;
                    ELSE Tstep_D <= T1;
                    END IF; - - data is loaded into IR in this time step
         … other states
      END CASE;
   END PROCESS;
   …
```

Figure 28. Fragment of example VHDL code.

This VHDL can be included in a Quartus II project, which can then be compiled and simulated. The values of the *Tstep_Q* signal can then be displayed in the Waveform Editor along with other signals in the circuit. However, a special process has to be used to insert the *Tstep_Q* signal into the waveform display. Recall from Figures 6 and 7 that the Node Finder tool is normally used as a convenient mechanism for importing signal names into the Waveform Editor. But for FSM state variables, it is not possible to use the Node Finder tool. Instead, it is necessary to use the dialogue from Figure 6 directly as illustrated in Figure 29. Here, the name *Tstep_Q* is manually typed into the name field, and the type of signal is set by using the drop-down dialogue to MACHINE. Clicking OK in Figure 29 imports the FSM signal into the Waveform Editor.

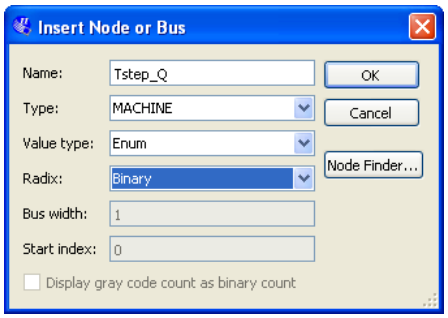

Figure 29. Specifying the *Tstep_Q* signal in the Insert Node or Bus window.

An example of simulation output that includes the *Tstep_Q* signal is given in Figure 30. This figure also shows the values of other signals in the circuit, which are needed in the simulation. Note that the values of Tstep_Q are displayed in a symbolic manner, according to the state names which were given in the VHDL code in Figure 28.

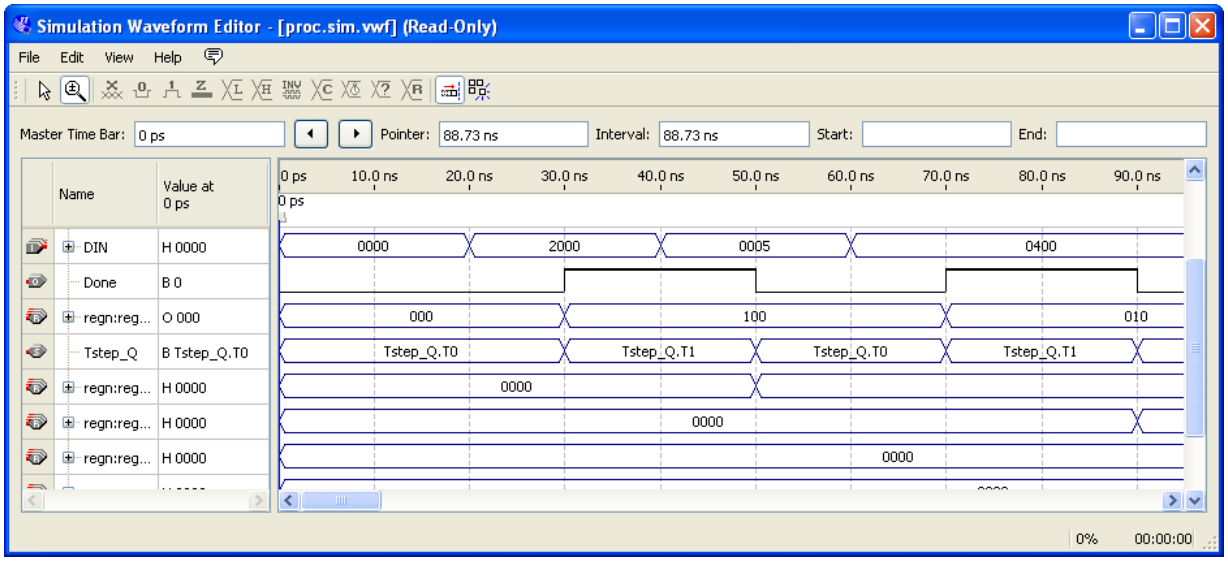

Figure 30. Simulation results of the *Tstep_Q* signal.