

# Simulate a Verilog Design Using a Waveform Input File

This tutorial is not developed by Intel. Instead, it follows the format and the contents of an Intel tutorial for Quartus Prime 18.1. Some necessary changes are made for the new version of Quartus Prime 23.1.

## 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 the Intel® Quartus® Prime CAD system.

The simulation method used in this tutorial is based on drawing waveforms, similar to timing diagrams, that are inputs for a simulator tool. The outputs of the simulator are also in the form of waveforms. This tutorial is intended for students who are taking a course in logic circuit design. We show how to use the Simulation Waveform Editor tool provided in the Quartus Prime software to perform a simulation of a circuit specified in Verilog HDL. Only a very basic understanding of Verilog is needed for this purpose.

### Contents:

- Design Project
- Creating Waveforms for Simulation
- Simulation
- Making Changes and Resimulating
- Concluding Remarks

For Quartus Prime 23.1, a Simulation Waveform Editor tool is available for use. It allows the user to apply inputs to the designed circuit, usually referred to as *test vectors*, in the form of waveforms and to observe the outputs generated in response.

In this tutorial, the reader will learn about:

- Test vectors needed to test the designed circuit
- Using the Simulation Waveform Editor tool to draw test vector waveforms
- Functional simulation, which is used to verify the functional correctness of a synthesized circuit

*(NOTE: Timing simulation is used to verify the timing of signals in a synthesized circuit. However, timing simulation is not supported by Cyclone V, which is used on DE1-SoC boards, which will be used in our labs. Timing Simulation for DE1-SoC boards will be identical to the functional simulation.)*

This tutorial is aimed at the reader who wishes to simulate circuits defined by using the Verilog hardware description language.

## 2 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 Verilog, this circuit can be specified as follows:

```
module majority3 (x1, x2, x3, f);
input    x1, x2, x3;
output  f;
assign  f = (x1 & x2) | (x1 & x3) | (x2 & x3);
endmodule
```

Implement this circuit as follows:

- Create a project *majority3*.
- Include a file *majority3.v*, which corresponds to the verilog snippet above.
- Select the correct device that is associated with your DE1-SoC board.
- Compile the design. Make sure your project is successfully compiled

## 3 Creating Waveforms for Simulation

To create test vectors for your design, select File > New... > Verification/Debugging Files > University Program VWF in the Quartus Prime window where the design project is open. This opens the **Simulation Waveform Editor** tool, shown in Figure 1, which allows you to specify the desired input waveforms.

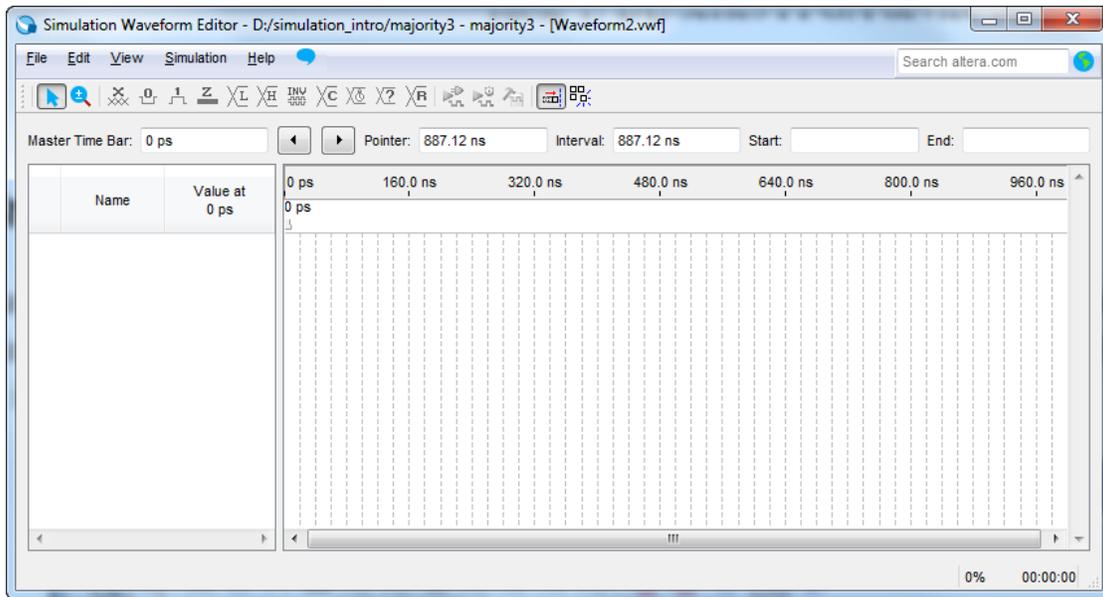


Figure 1. The Waveform Editor window.

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 1.

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 2, click on `Node Finder`.

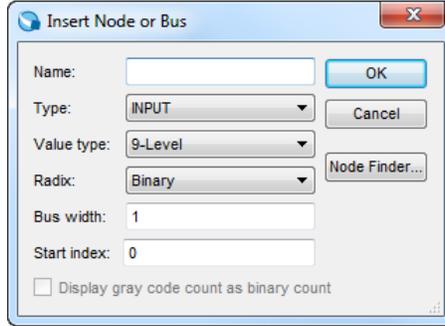


Figure 2. The Insert Node or Bus dialog.

The Node Finder window is presented in Figure 3. 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 4. Click OK in this window and also upon return to the window in Figure 2. This returns to the Waveform Editor window, with the selected signals included as presented in Figure 5.

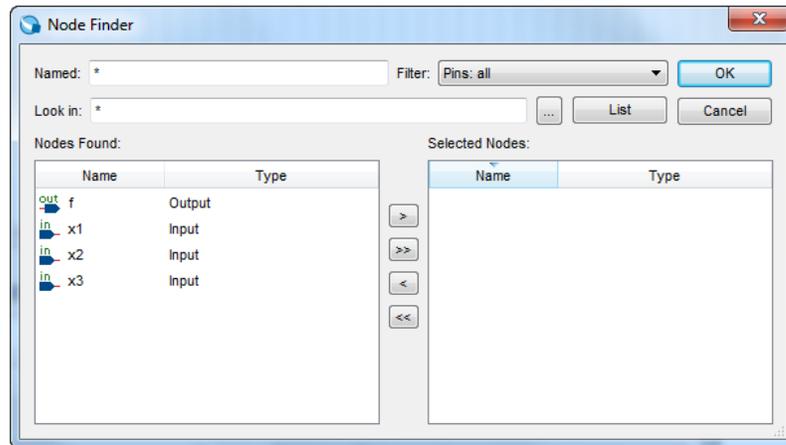


Figure 3. The Node Finder dialog.

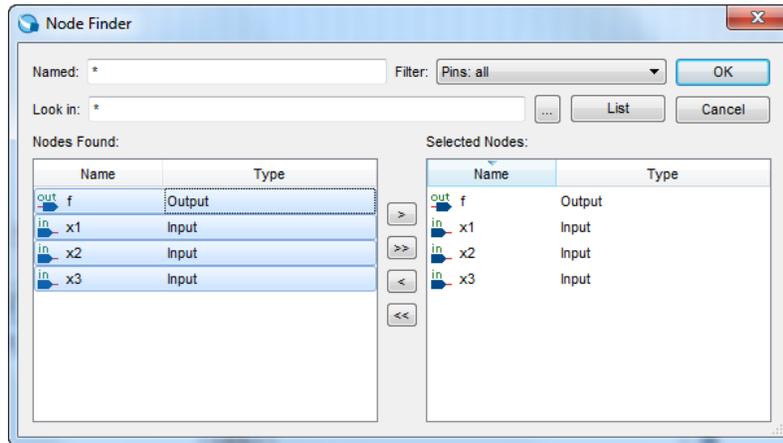


Figure 4. The selected signals.

Observe that in Figure 5 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 6 specifying the desired size. The spacing of grid lines in Figure 5 is 20 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 , or by selecting the command **Edit > Snap to Grid**.

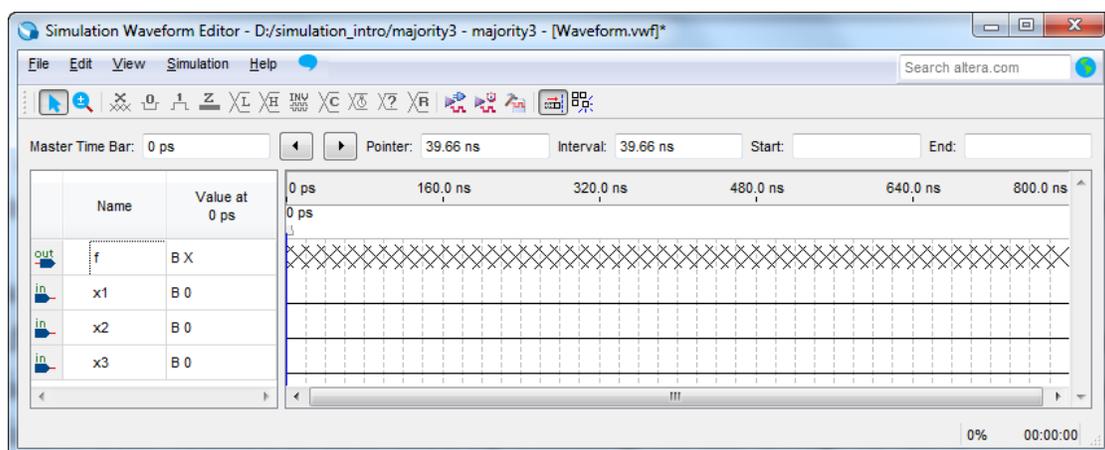


Figure 5. Signals in the Waveform Editor window.

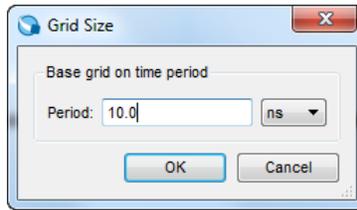


Figure 6. Specifying the grid spacing.

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 7. Change the value of the waveform to 1 by clicking on the Forcing High (1) icon , as illustrated in Figure 8.

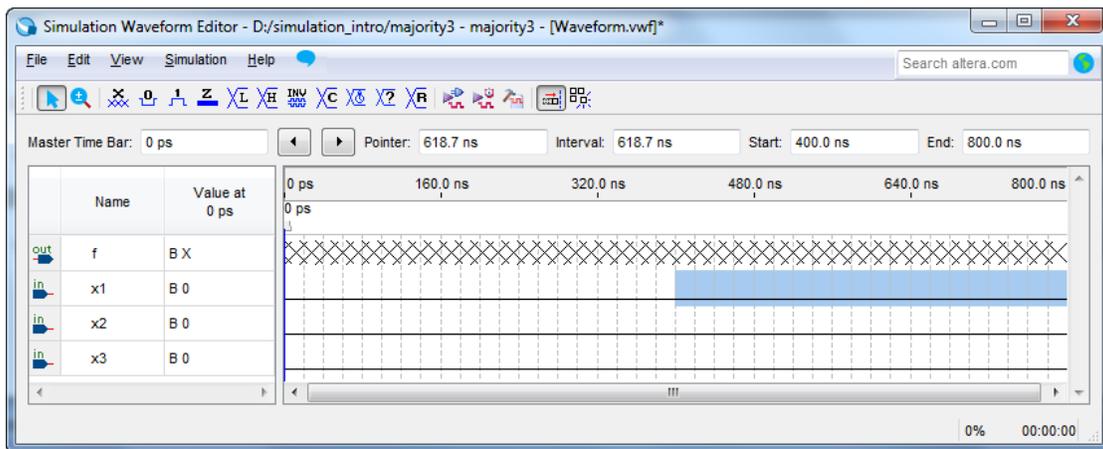


Figure 7. Selection of a time interval.

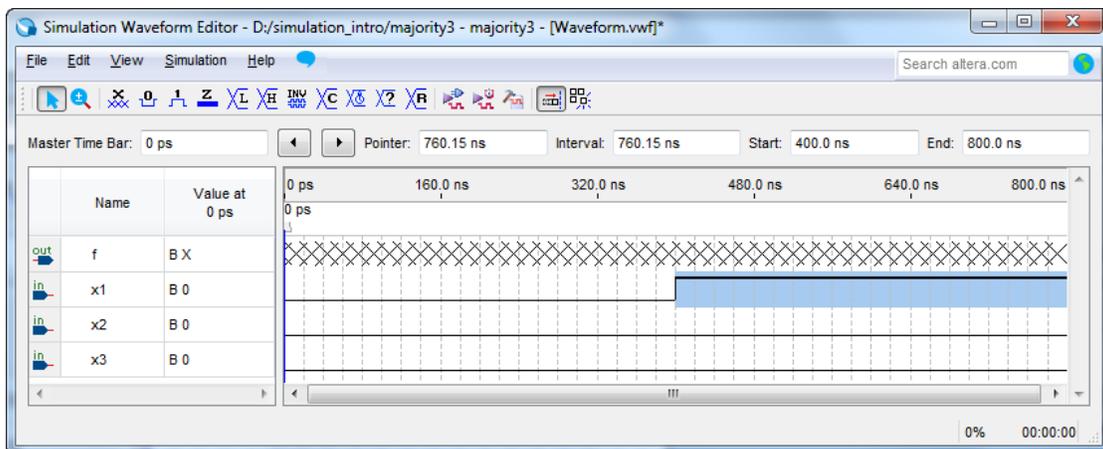


Figure 8. Drawing the waveform for *x1*

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. Then do the same for the interval from 600 to 800 ns, as illustrated in Figure 9.

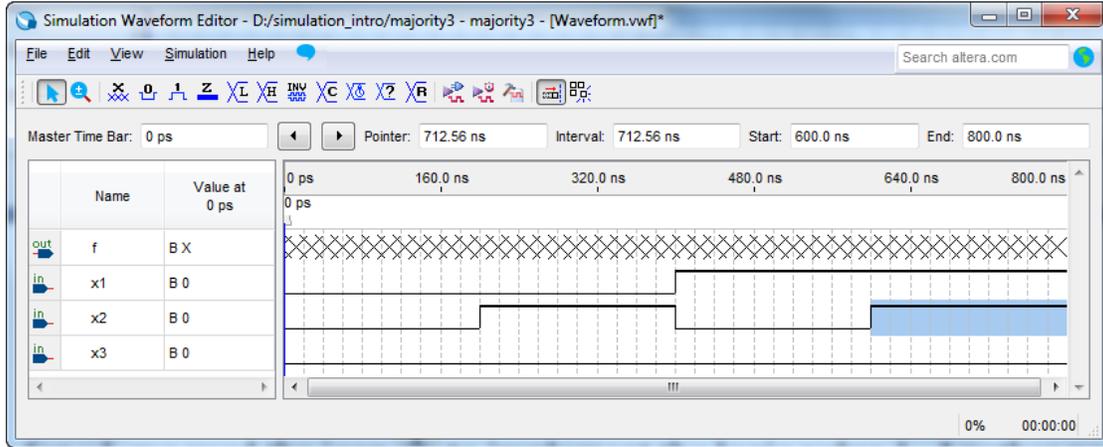


Figure 9. Drawing the waveform for  $x2$ .

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 10. This leads to the pop-up window in Figure 11. Specify the clock period of 200 ns and the duty cycle of 50%, and click OK. The result is depicted in Figure 12.

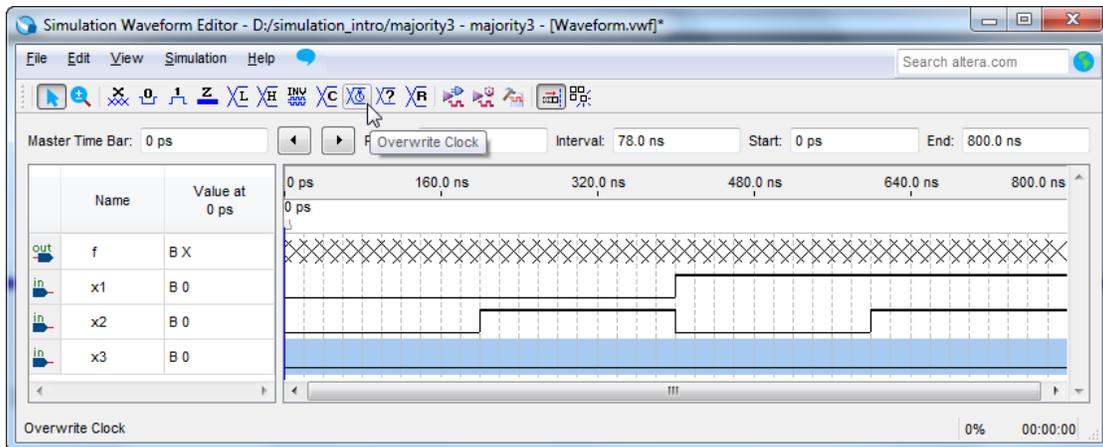


Figure 10. Drawing the waveform for  $x3$ .

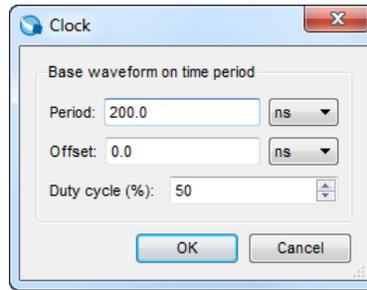


Figure 11. Defining the clock characteristics

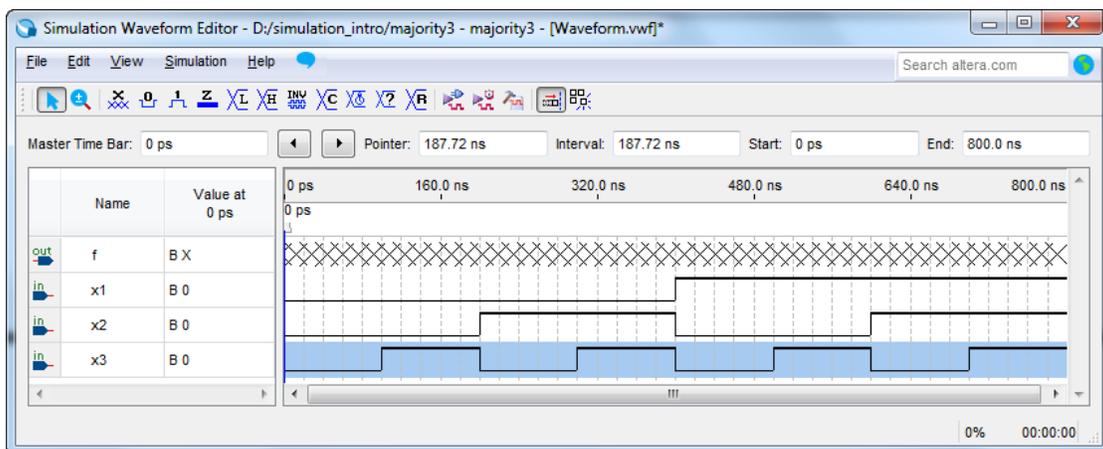


Figure 12. The completed input waveforms.

Save the waveform file as *Waveform.vwf*, or other names you like. Note that the suffix *vwf* stands for *vector waveform file*. VWF files that are added to the Quartus Prime project can be accessed at any time in the Project Navigator Widget's Files tab. **Please note:** if the vector waveform file is not named as "*Waveform.vwf*", some simulation settings need to be changed to make the simulation work. See Section 5 for Functional Simulation.

## 4 Quartus Prime Set Up for Simulation

For Quartus Prime 23.1, the default EDA tool for simulation is Questa-Intel simulator. Normally, two versions of Questa Intel simulators will be available. They are: Questa-Intel FPGA Edition and Questa-Intel Starter Edition. Both will work for our lab projects, but the Questa-Intel FPGA Edition can perform at higher speed and can handle more design instances.

Check the followings to make sure you are using the correct simulator:

In Quartus Prime, go to **Assignments> Settings**, then in the left panel of the window, select “**Simulation**” under “**EDA Tool Settings**”. In the right side of the window, select “**Questa Intel FPGA**” for “**Tool Name**”. See Figure 13.

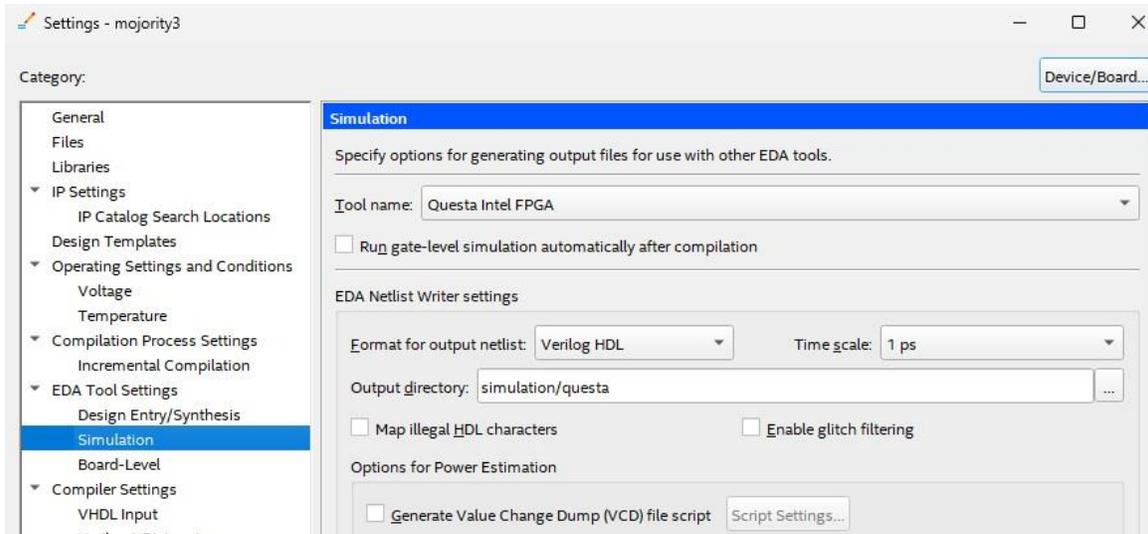


Figure 13. Tool Name Setting for EDA Tool

Then in Quartus Prime, go to **Tools > Options...> EDA Tool Options**. In the right side of the window, set “**Questa Intel FPGA**” to the folder where the Questa-Intel FPGA is installed. In Figure 14, this installation folder is “*C:/IntelFPGA/23.1std/questa-fe/win64*”, which is for Questa-Intel FPGA Edition. If the Questa-Intel Starter Edition is preferred, the folder would be “*C:/IntelFPGA/23.1std/questa-fse/win64*”.

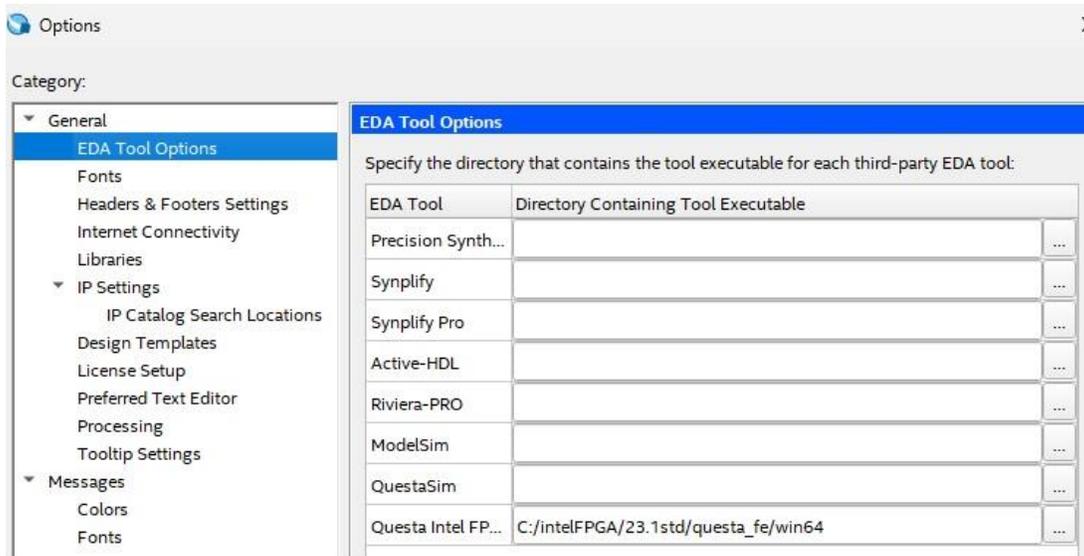


Figure 14. EDA Tool Option Settings for Questa-Intel FPGA Edition

If other EDA tools are available and you would like to use them, such as if you would like to use ModelSim Intel FPGA, you can specify it in Quartus Prime by going through the above two steps.

## 5 Functional Simulation

The Waveform Editor is originally designed to work with ModelSim simulator for old version of Quartus software. Some simulation settings need to be changed to make it work with Questa-Intel simulators for Quartus Prime 23.1std Edition.

In **Simulation Waveform Editor**, click menu **Simulation** and select **Simulation Settings**, in the opened Simulation Options window, click the tab for **Functional Simulation Settings**, and check or make the following changes (See Figure 15):

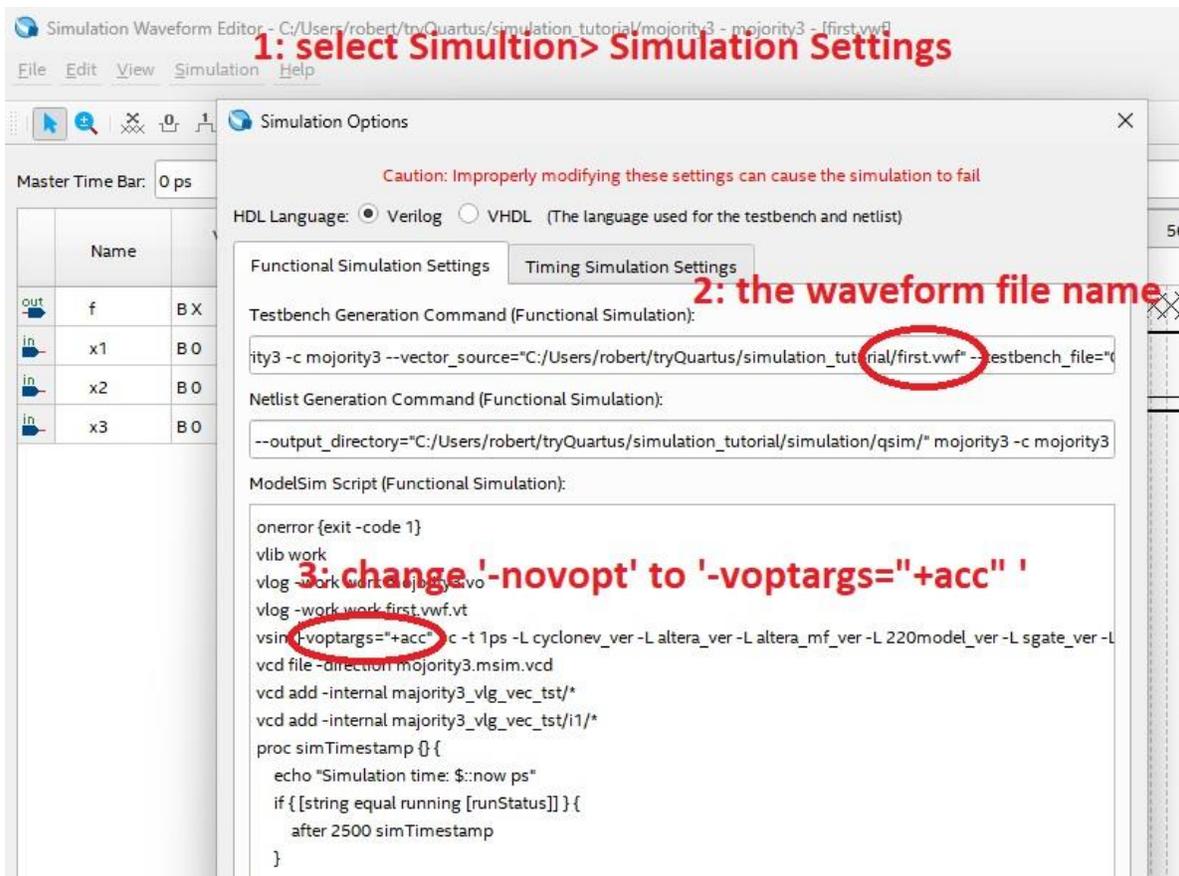


Figure 15 Change Simulation Settings for Using Waveform Input File with Questa-Intel Simulator

1. In the **Testbench Generation Command** box, check the waveform file name for argument of “- - vector\_source”. By default, it is “Waveform.vwf”. If you have saved your waveform file in a different name, make sure here is your file’s name.
2. In the **ModelSim Script** area, find the line for **vsim** command, and change its option of ‘-novopt’ to

'-voptargs='+acc'' (without the single quotation marks) .

Then click the **Save** button to save the settings and to close the window for Simulation Setting.

Then in the Simulation Waveform Editor, select **Simulation > Run Functional Simulation**, or click on the icon on the tool bar for Functional Simulation. A pop-up window will show the progress of the simulation, then automatically close when it is complete. A second Simulation Waveform Editor window then opens the output waveform, as depicted in Figure 16. The output waveform is read-only, so any changes in simulation have to be done by modifying the *Waveform.vwf* file and resimulating the circuit. 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.

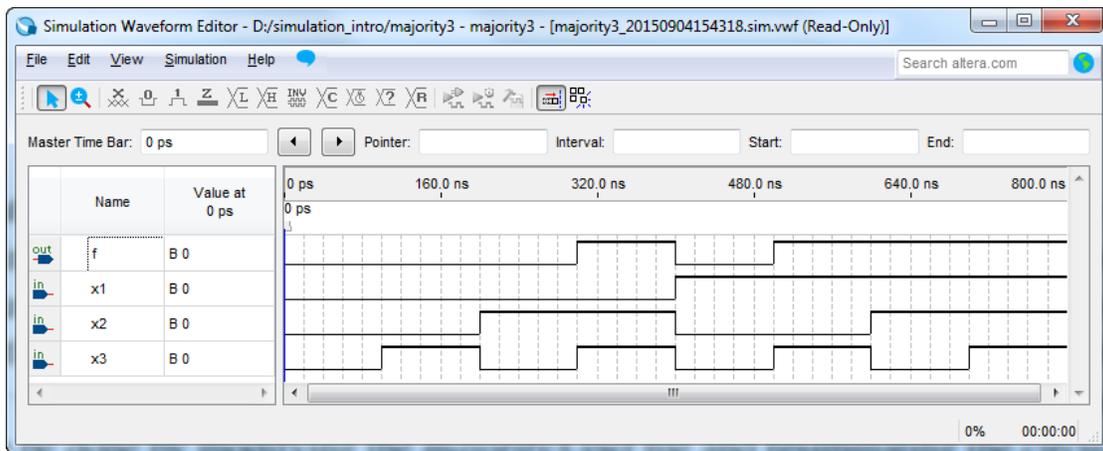


Figure 16. Result of the functional simulation.

## 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 17. Now, simulate the circuit again. The result is given in Figure 18. If errors in the circuit are discovered, then these errors can be fixed by changing the Verilog code and recompiling the design using the Quartus Prime software.

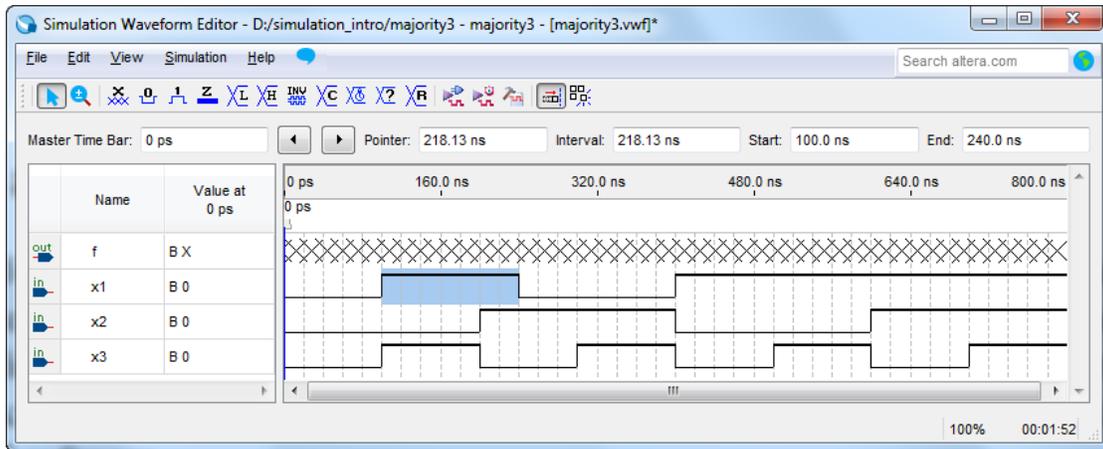


Figure 17. Modified input waveforms.

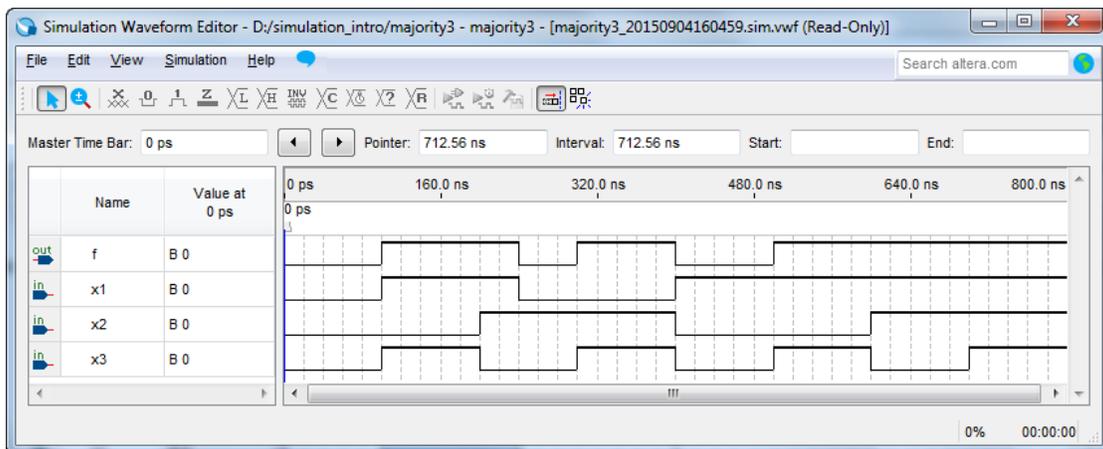


Figure 18. Result of the new simulation.

## 7 Concluding Remarks

The purpose of this tutorial is to provide a quick introduction to the Simulation Waveform Editor, explaining only the rudimentary aspects of functional and timing simulations. Details about additional features of the Simulation Waveform Editor can be found in the appendix of this document.

# A Simulation Waveform Editor

In section 3 we introduced 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, Simulation, 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 menu that appears.

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).

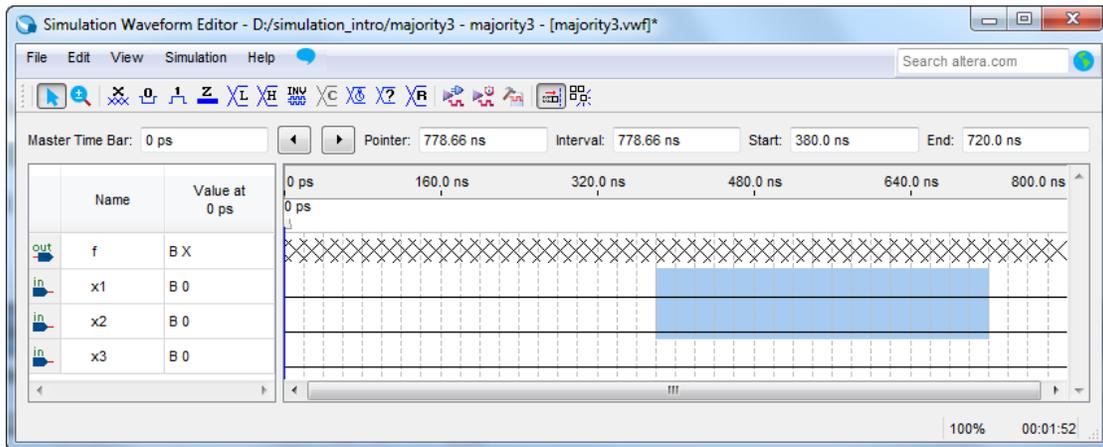


Figure 1. Using the Selection Tool to select a portion of multiple waveforms.

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 brings up the window 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.

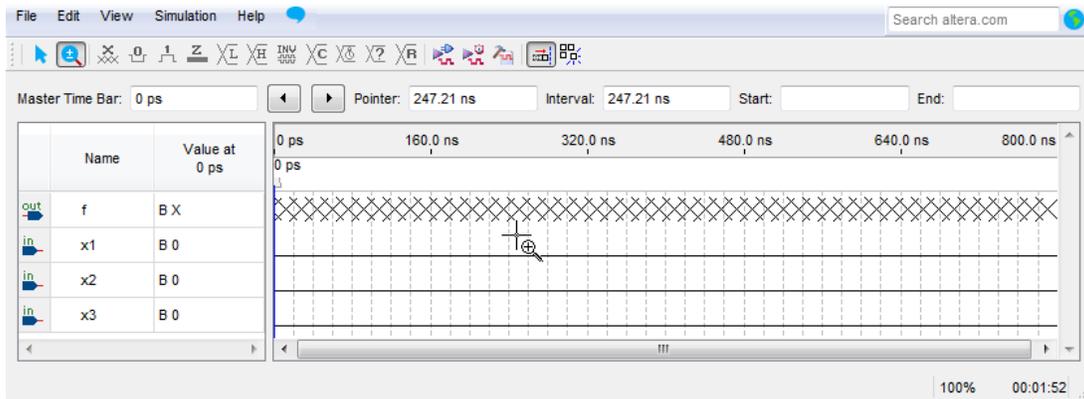


Figure 2. Using the Zoom Tool.

### 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 majority3 function circuit that was described in section 2. The value of the signal  $x_3$  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 the value of  $f$  is unknown between 200 to 400 ns.

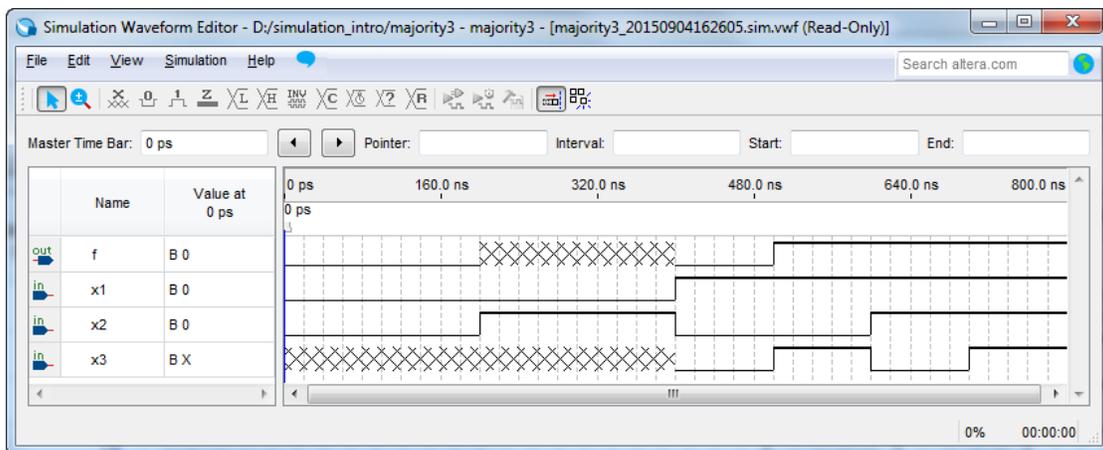


Figure 3. Setting the value of an input to Unknown (X).

### 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.

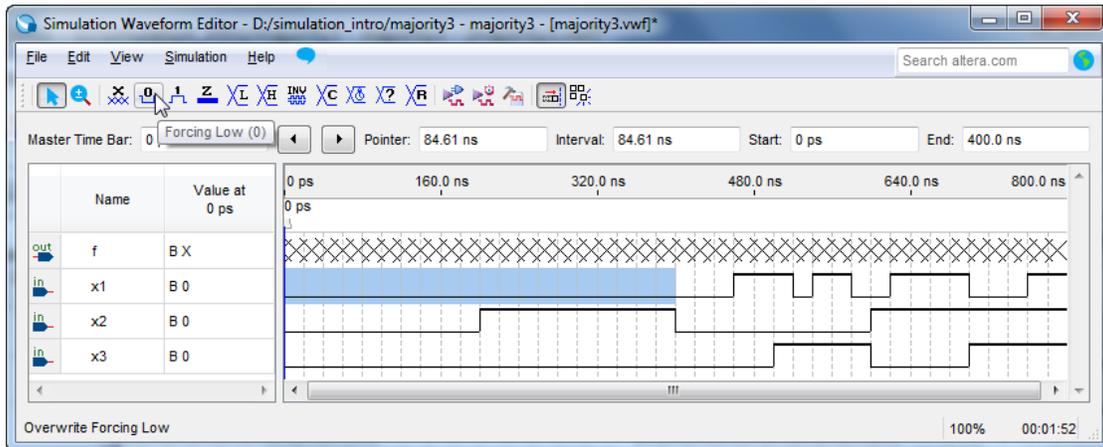


Figure 4. Forcing  $x_1$  to be low from 0 to 400 ns.

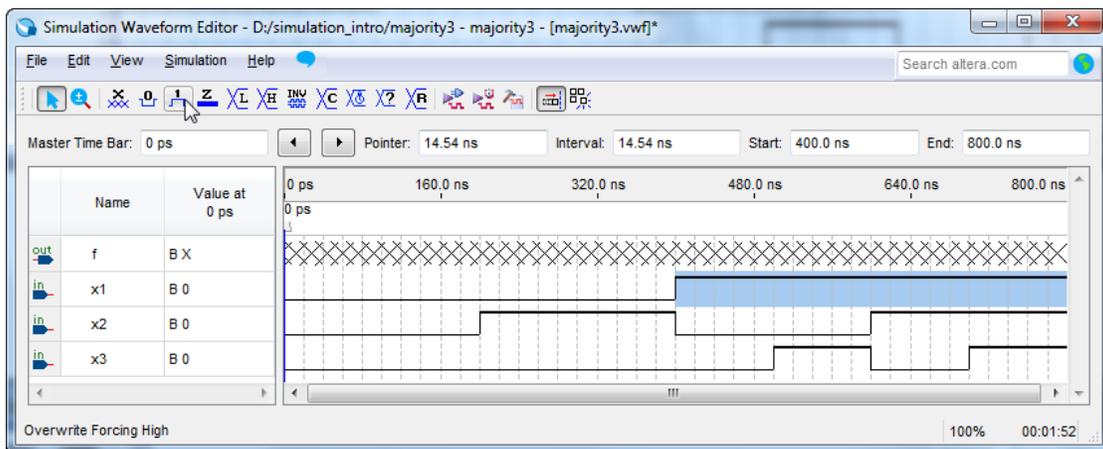


Figure 5. Forcing  $x_1$  to be high from 400 to 800 ns.

### 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.

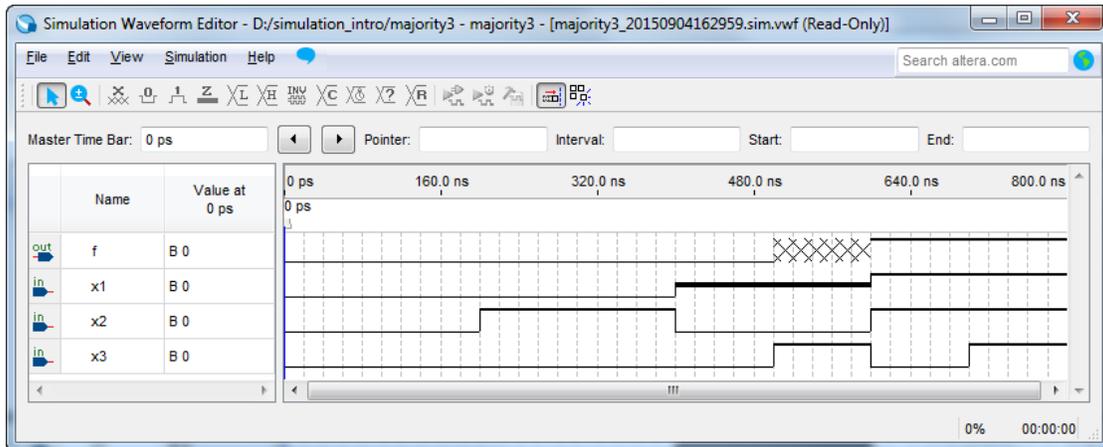


Figure 6. Setting a signal to high impedance.

Weak Low (L)  and Weak High (H) 

These tools are used to set a signal to the values Weak Low (L) or Weak High (H), 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.

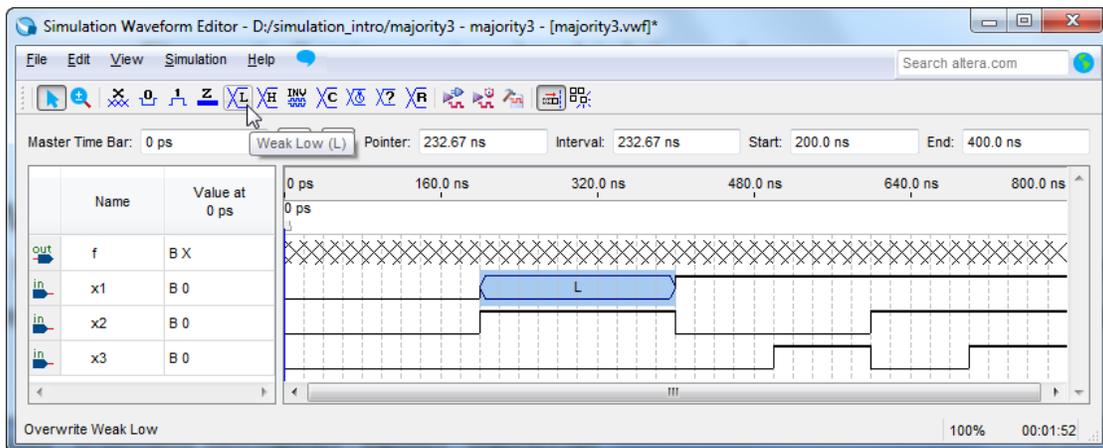


Figure 7. Changing the *x1* signal to be weak low from 200 to 400 ns.

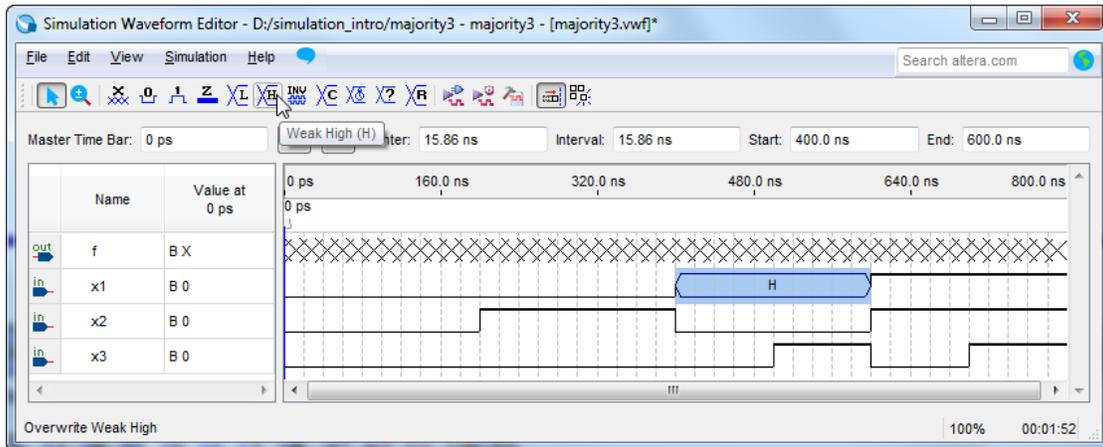


Figure 8. Changing the x1 signal to be weak high from 400 to 600 ns.

### 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.

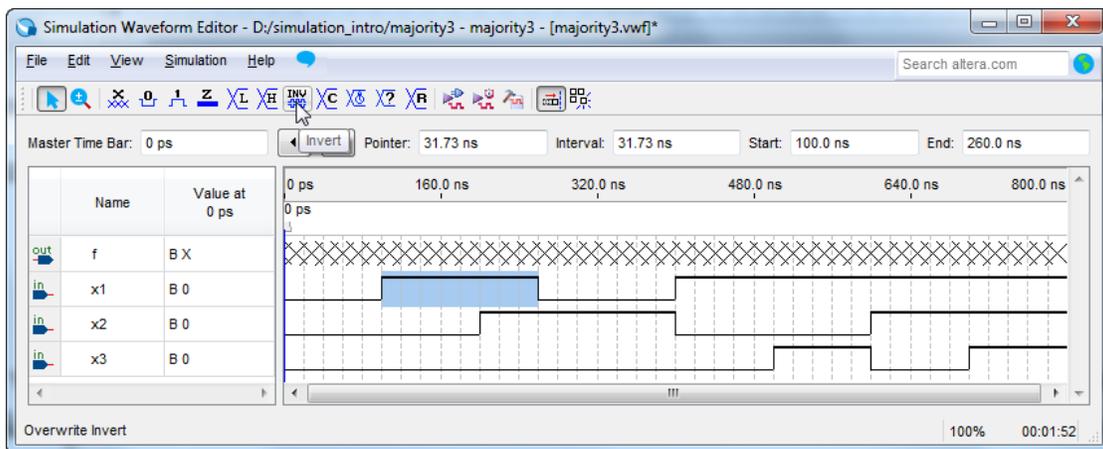


Figure 9. Inverting the x1 signal from 100 to 260 ns.

### 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.

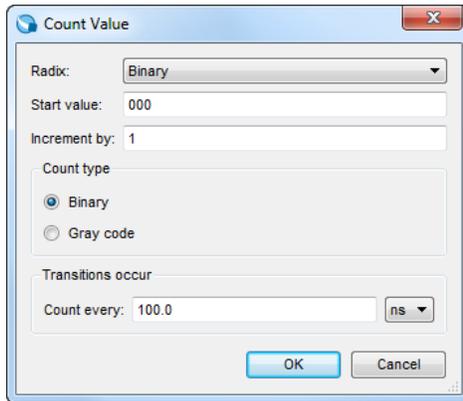


Figure 10. Options available for for the Count Value tool.

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

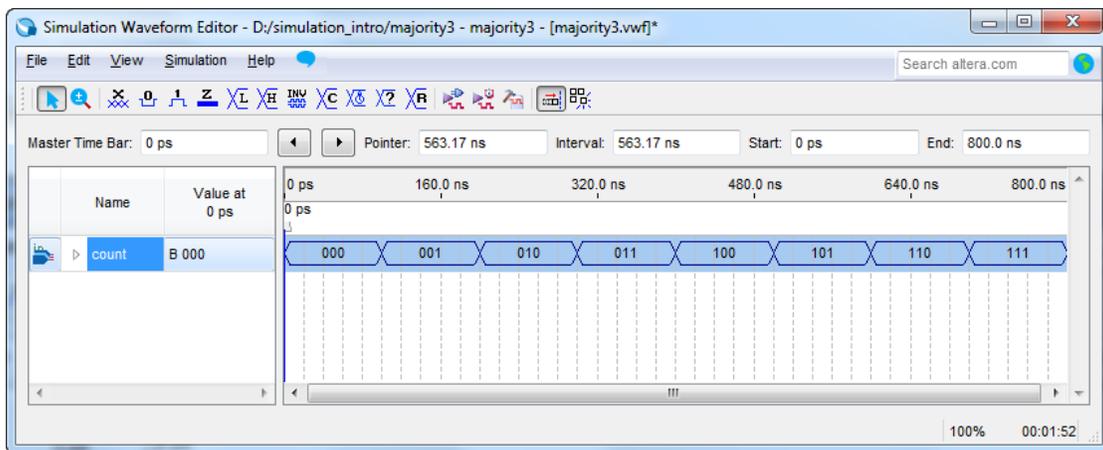


Figure 11. An example of using the Count Value tool.

### 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.

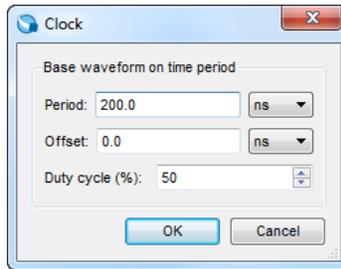


Figure 12. Options available for the **Overwrite Clock** tool.

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%.

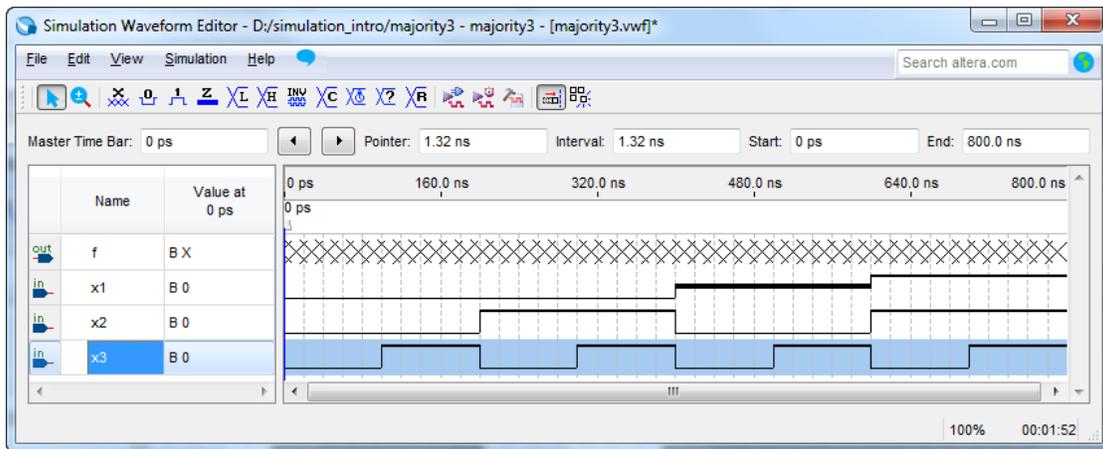


Figure 13. An example of using the **Overwrite Clock** tool.

### 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.

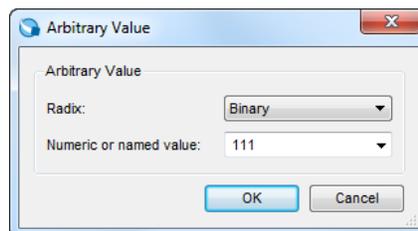


Figure 14. Options available for for the **Arbitrary Value** tool.

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

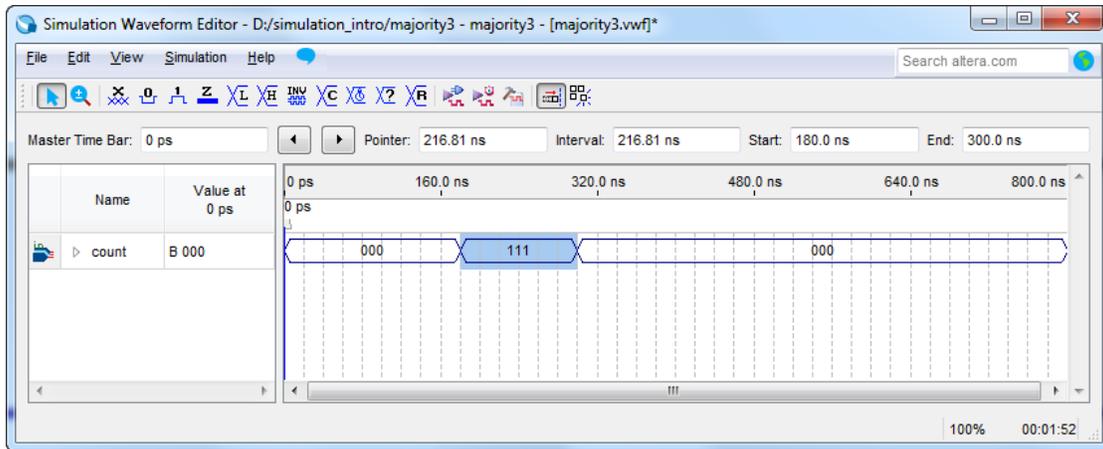


Figure 15. The Arbitrary Value tool is used to set values for the *count* signal.

### Random Values

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

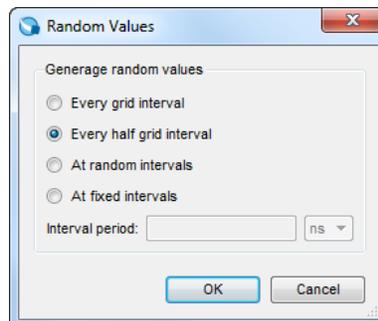


Figure 16. Various options available for the Random Value tool.

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

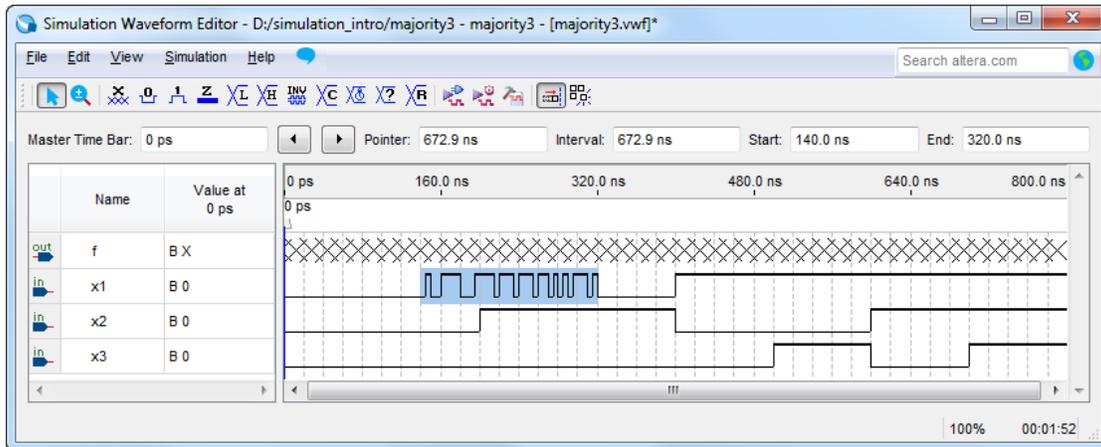


Figure 17. An example of the Random Value tool being used.

### 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.

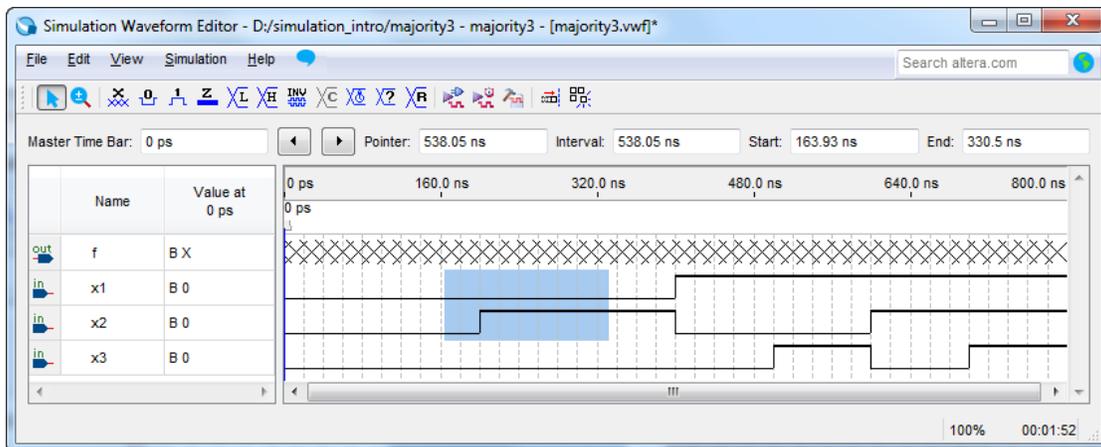


Figure 18. An example of the Snap to Grid option turned off.

### Snap to Transition

This option allows the Selection Tool to automatically extend both the left and right sides of the selection rectangle outwards to the first transition encountered by any waveform in the editor. 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.

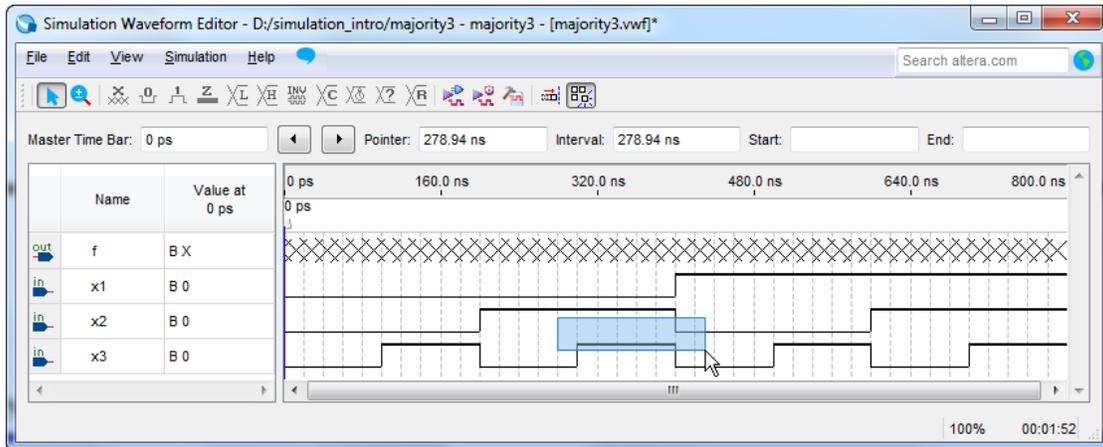


Figure 19. Making a selection with the Snap to Transition option enabled.

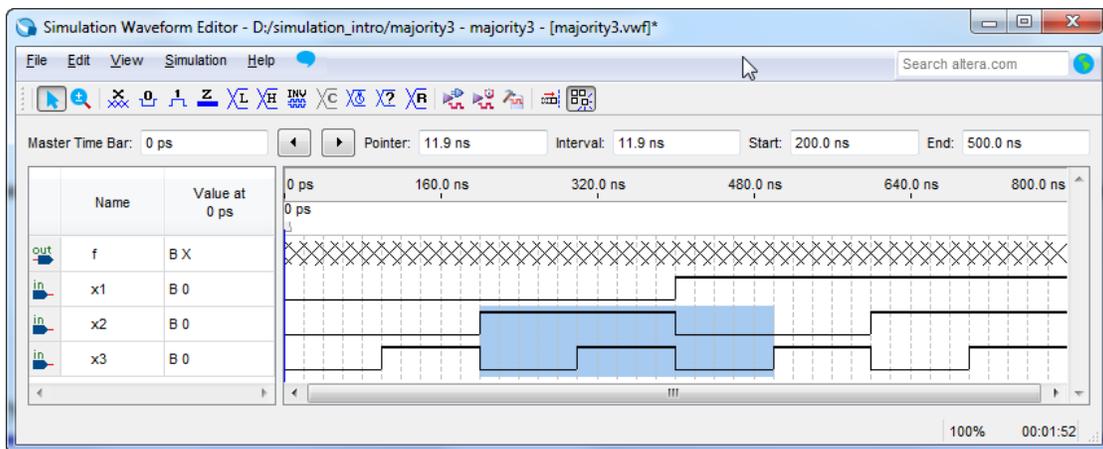


Figure 20. The expanded selection resulting from Figure 19.

## 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 left side of the Waveform Editor with the key **Ctrl** pressed 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...**

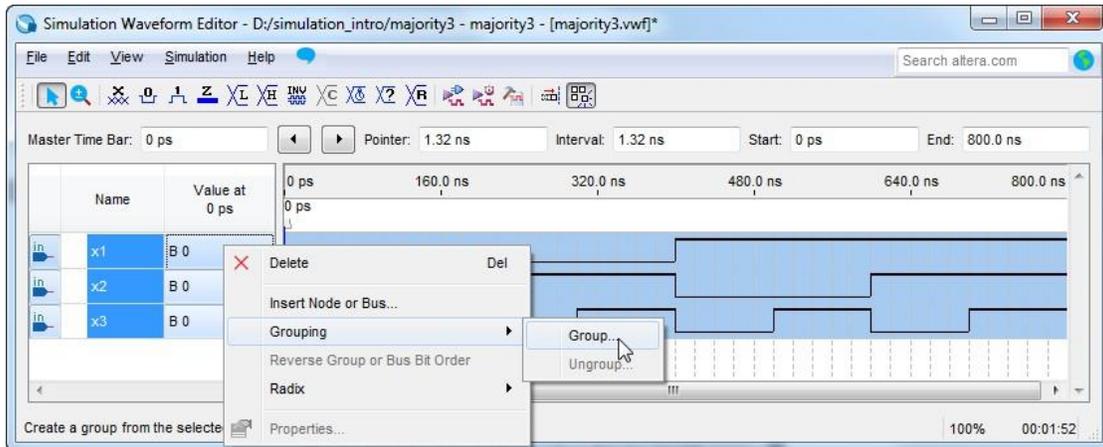


Figure 21. An example of grouping signals.

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.

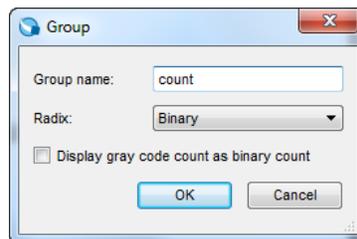


Figure 22. Select a name and radix for the group of signals.

The resulting group of signals is shown in Figure 23. The multibit waveform can be expanded in the waveform editor to display its individual signals.

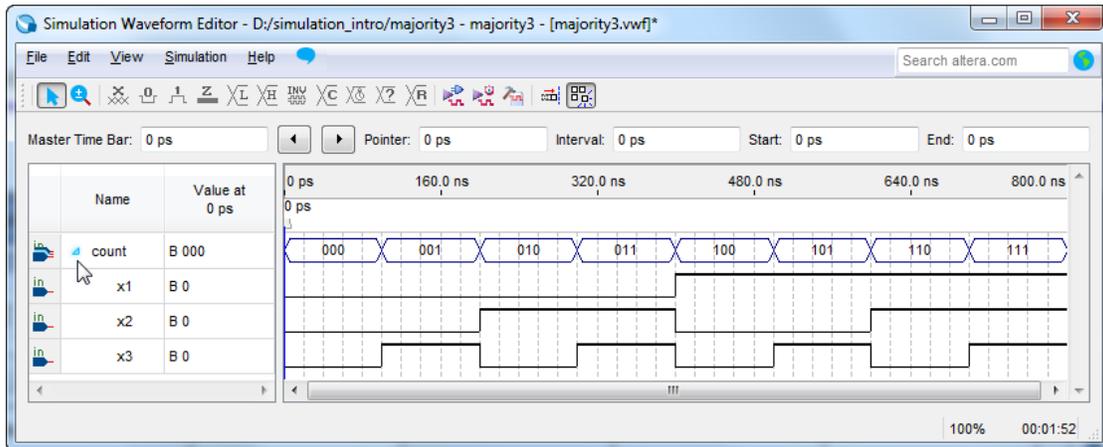


Figure 23. An example of expanding a multibit signal.

A multibit signal can be ungrouped by right-clicking on the group of signals and selecting Grouping > Ungroup. . . .

It is also possible to create hierarchical groupings of signals as illustrated in Figure 24. In this example, the two bit signal called *level2* is combined with the signal called *x3* to create the three bit signal called *level1*. It is only possible to group and ungroup top-level signals.

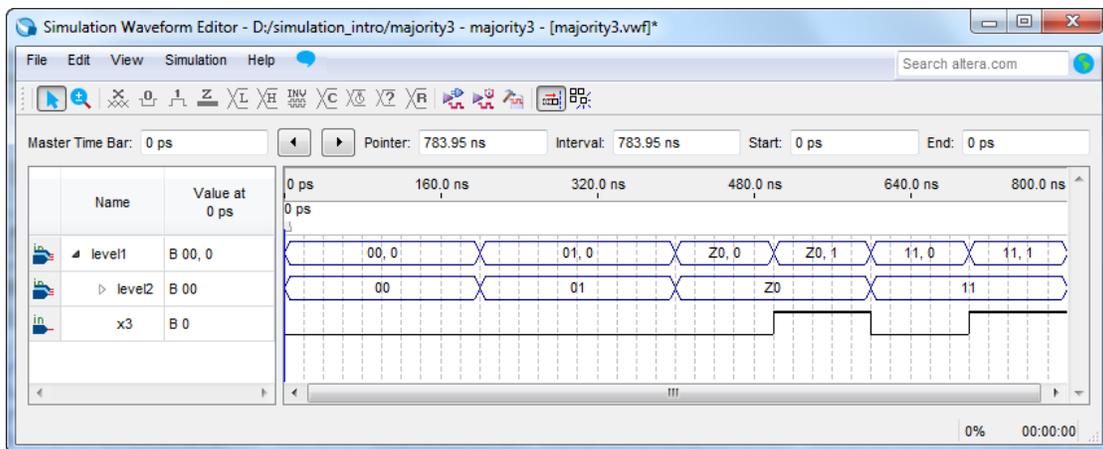


Figure 24. An example of hierarchical groups.

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

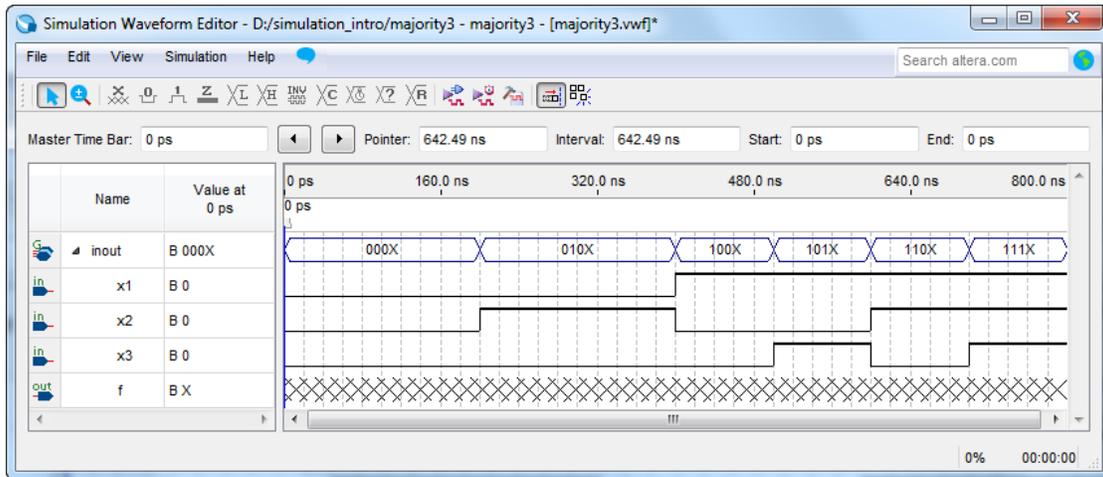


Figure 25. An example of grouping input and output signals.

## 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.

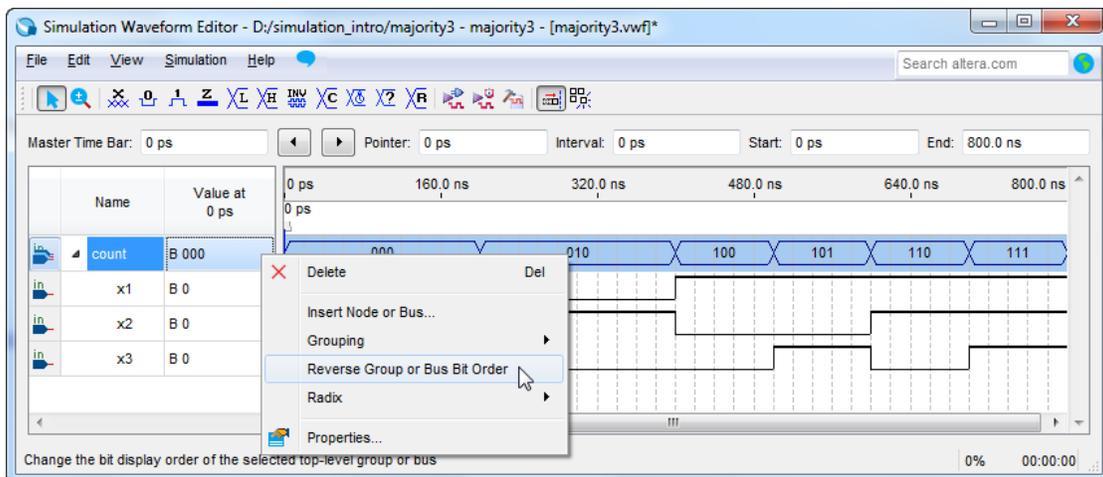


Figure 26. Reversing the bit order on a group of signals.

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

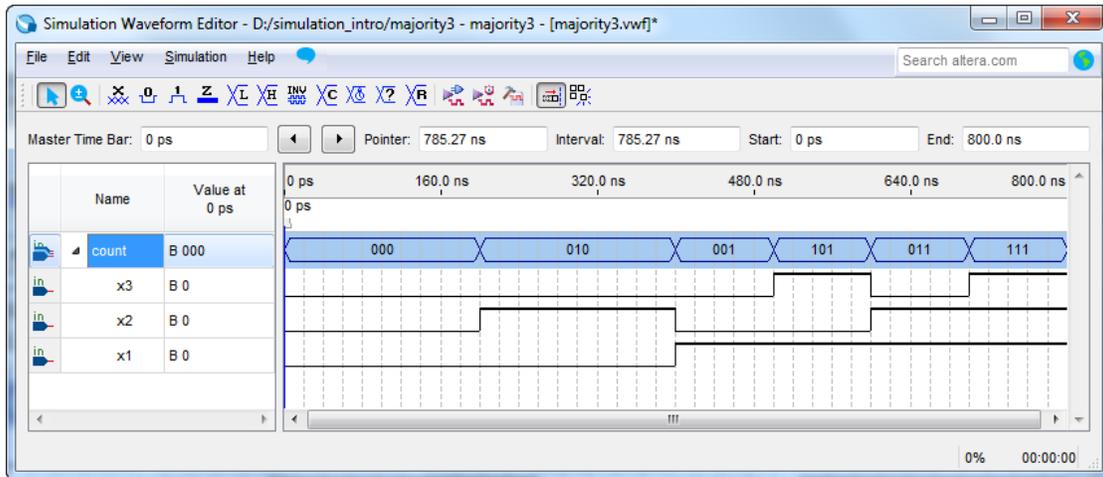


Figure 27. The result of reversing the bit order in Figure 26.