Lane Department of Computer Science and Electrical Engineering

Lab: 5

CpE 272 Digital Logic Laboratory

Introduction to Logic Simulation

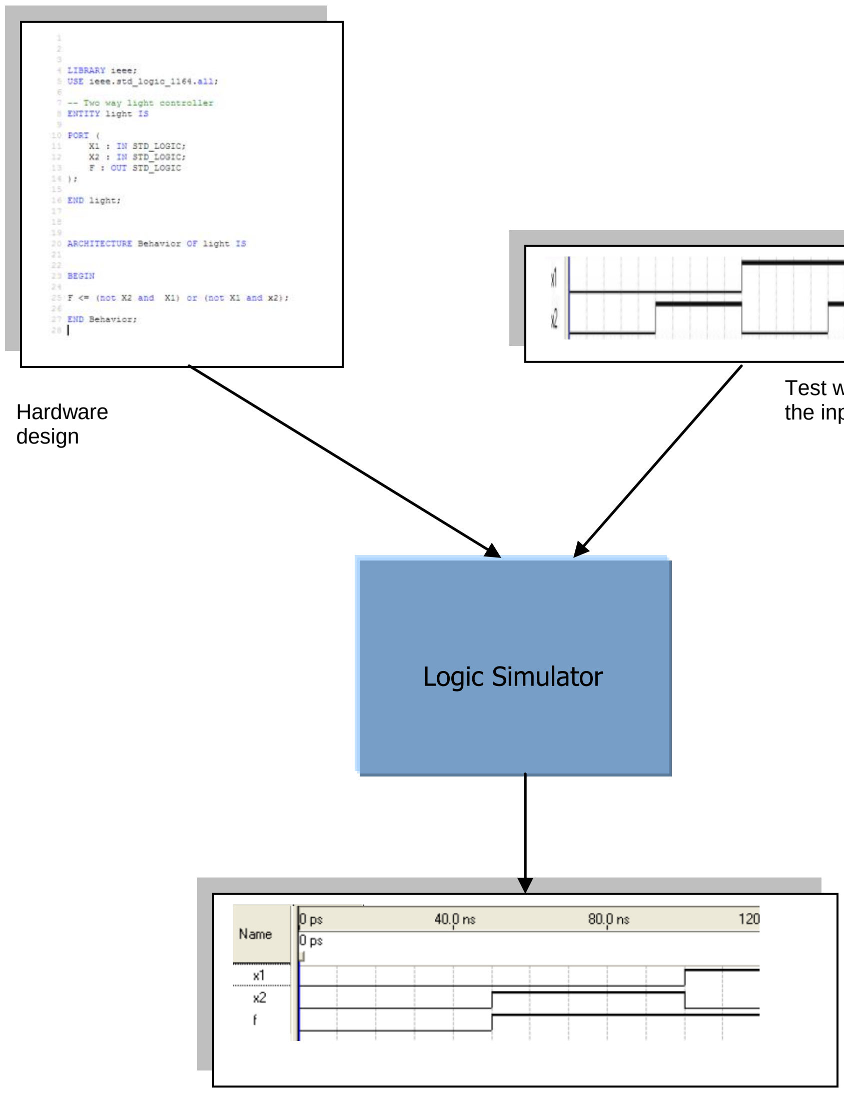
Fall 2010

West Virginia University - College of Engineering and Mineral Resources

Introduction

The technique of logic simulation is widely used to verify the correctness of a hardware design. Most of the time logic simulation is the first step towards taking hardware design from concept to realization. Without simulations, we have to download the hardware design in to the FPGA and verify it every time we made a change to the design.

Figure 1 illustrates the concept of logic simulation. The simulator takes two inputs, the VHDL file and an input waveform file and produces the resulting waveform.



```



LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Two way light controller
ENTITY light IS

PORT (
    X1 : IN STD_LOGIC;
    X2 : IN STD_LOGIC;
    F : OUT STD_LOGIC
);

END light;



ARCHITECTURE Behavior OF light IS


BEGIN

F <= (not X2 and  X1) or (not X1 and x2);

END Behavior;
```

Figure 1. The concept of logic simulation

Simulations simplify this by providing a means of verify the circuit at early levels of the design flow. Figure 2 shows the simplified design flow with simulations.

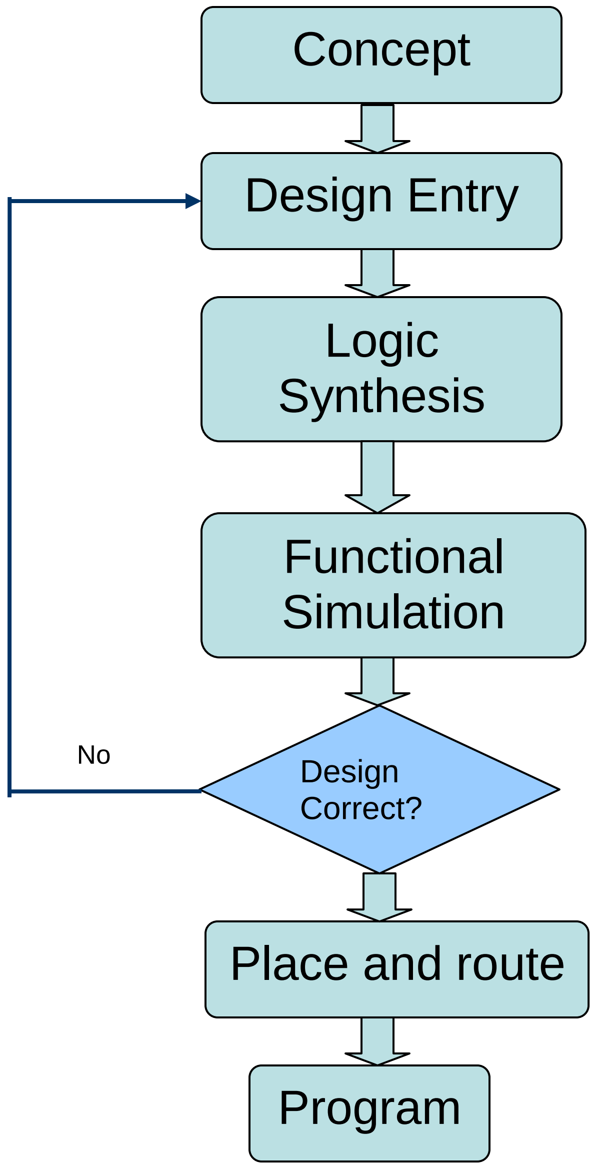
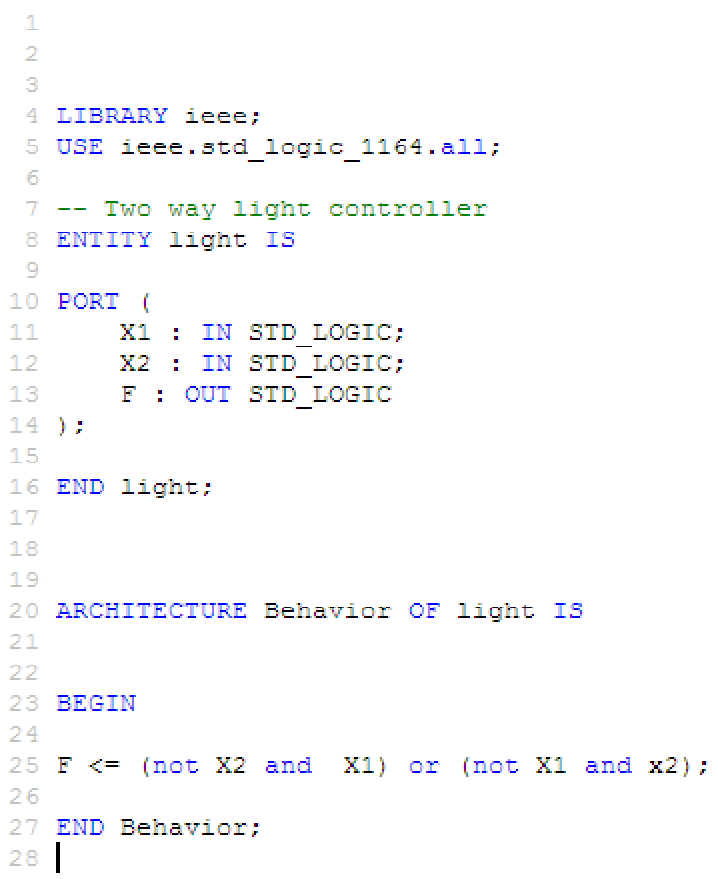


Figure 2. The design flow with logic simulation

Part I – Creating a simple project

1. Create a new project.
2. Add a VHDL file and type the code shown in Figure 3.
3. Compile the code



```vhdl



LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Two way light controller
ENTITY light IS

PORT (
    X1 : IN STD_LOGIC;
    X2 : IN STD_LOGIC;
    F : OUT STD_LOGIC
);

END light;



ARCHITECTURE Behavior OF light IS


BEGIN

F <= (not X2 and  X1) or (not X1 and x2);

END Behavior;

```

Figure 3. VHDL code for two way light controller

Part II - Simulating the Designed Circuit

Quartus II software includes a simulation tool that can be used to simulate the behavior of a designed circuit. Before the circuit can be simulated, it is necessary to create the desired waveforms, called *test vectors*, to represent the input signals. It is also necessary to specify which outputs, as well as possible internal points in the circuit, the designer wishes to observe. The simulator applies the test vectors to a model of the implemented circuit and determines the expected response.

We will use the Quartus II Waveform Editor to draw the test vectors, as follows:

1. Open the Waveform Editor window by selecting **File > New**, which gives the window shown in Figure 4.
2. Click on the Other Files tab to reach the window displayed in Figure 5. Choose **Vector Waveform File** and click **OK**.

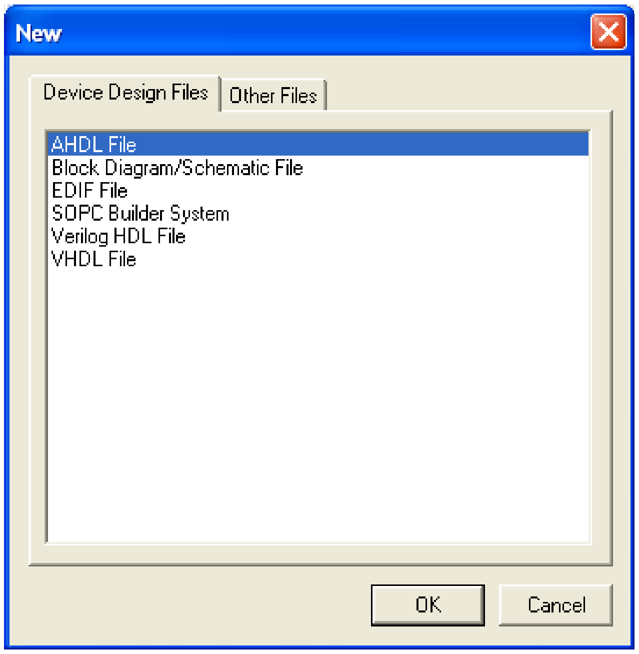


Figure 4. Need to prepare a new file.

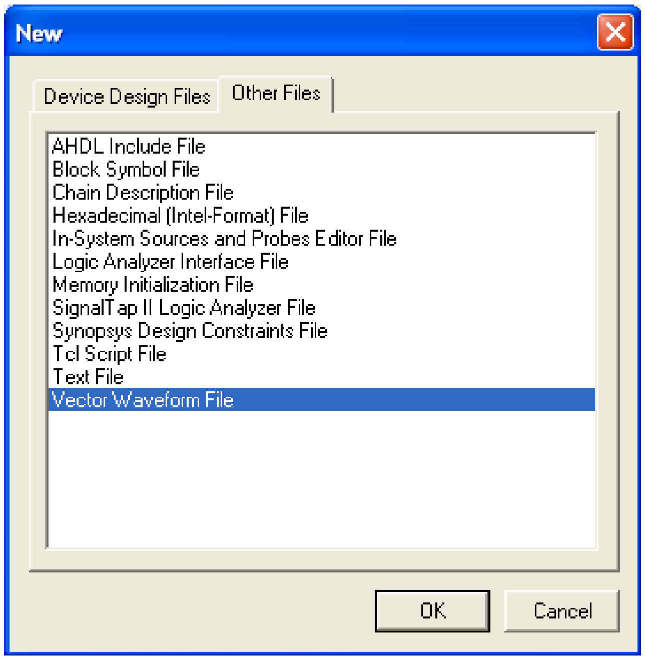

New
Device Design Files
Other Files
AHDL Include File
Block Symbol File
Chain Description File
Hexadecimal (Intel-Format) File
In-System Sources and Probes Editor File
Logic Analyzer Interface File
Memory Initialization File
SignalTap II Logic Analyzer File
Synopsys Design Constraints File
Tcl Script File
Text File
Vector Waveform File
OK
Cancel

Figure 5. Choose to prepare a test-vector file.

3. The Waveform Editor window is depicted in Figure 6. Save the file under the name *light.vwf*; note that this changes the name in the displayed window.
4. Set the desired simulation to run from 0 to 200 ns by selecting **Edit > End Time** and entering 200 ns in the dialog box that pops up.
5. Selecting **View > Fit in Window** displays the entire simulation range of 0 to 200 ns in the window, as shown in Figure 7. You may wish to resize the window to its maximum size.

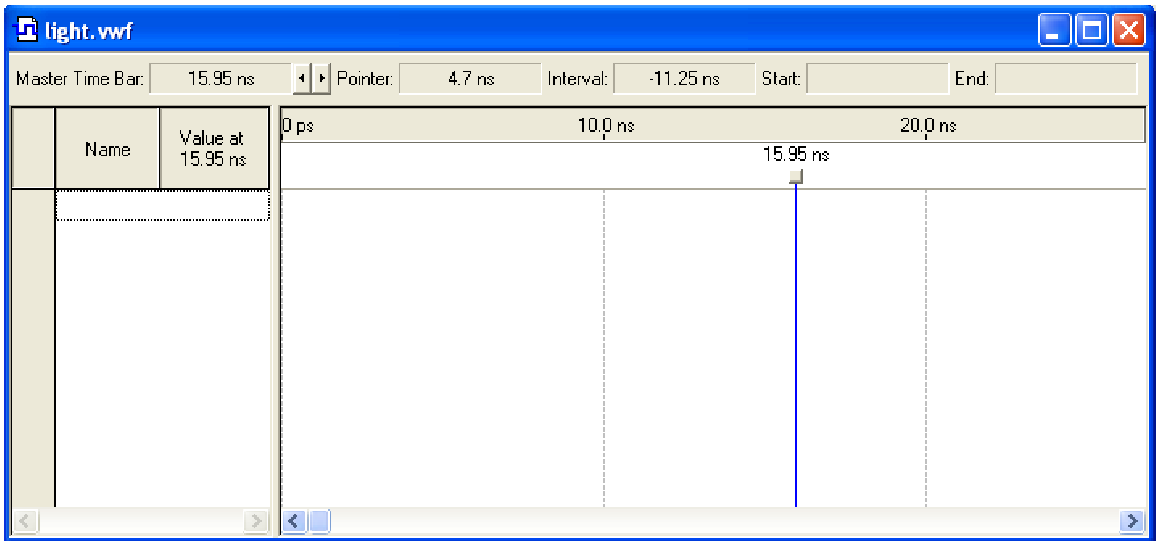


Figure 6. The Waveform Editor window.

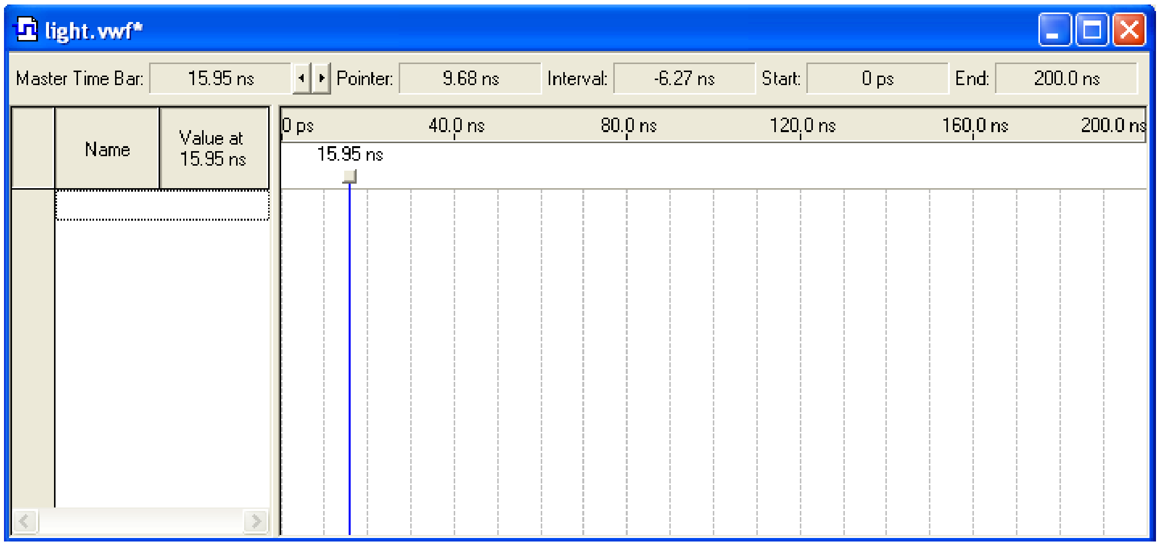
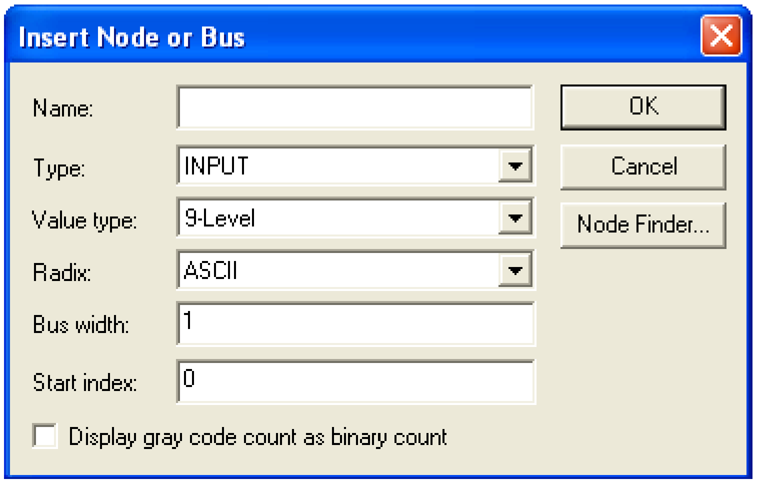


Figure 7. The augmented Waveform Editor window.

Next, we want to include the input and output nodes of the circuit to be simulated.

1. Click Edit **> Insert >Insert Node or Bus** to open the window in Figure 8.
2. Click on the button labeled **Node Finder** to open the window in Figure 9. The Node Finder utility has a filter used to indicate what type of nodes are to be found. Since we are interested in input and output pins, set the filter to **Pins: all**. Click the **List** button to find the input and output nodes as indicated on the left side of the figure. 22



Name:

Type: INPUT

Value type: 9-Level

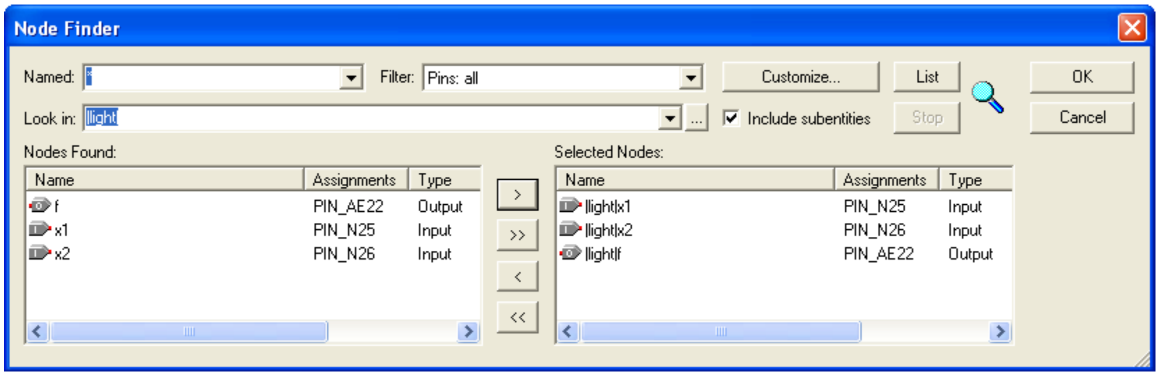
Radix: ASCII

Bus width: 1

Start index: 0

☐ Display gray code count as binary count

Figure 8 The Insert Node or Bus dialogue.



Named: *

Filter: Pins: all

Look in: |light|

☑ Include subentities

Nodes Found:

Name	Assignments	Type
f	PIN_AE22	Output
x1	PIN_N25	Input
x2	PIN_N26	Input

Selected Nodes:

Name	Assignments	Type
|light|x1	PIN_N25	Input
|light|x2	PIN_N26	Input
|light|f	PIN_AE22	Output

Figure 9. Selecting nodes to insert into the Waveform Editor.

3. Click on the *x1* signal in the **Nodes Found** box in Figure 9, and then click the **>** sign to add it to the Selected Nodes box on the right side of the figure.
4. Do the same for *x2* and *f*. Click **OK** to close the Node Finder window, and then click **OK** in the window of Figure 8. This leaves a fully displayed Waveform Editor window, as shown in Figure 10. If you did not select the nodes in the same order as displayed in Figure 10, it is possible to rearrange them. To move a waveform up or down in the Waveform Editor window, click on the node name (in the **Name** column) and release the mouse button.
5. The waveform is now highlighted to show the selection. Click again on the waveform and drag it up or down in the Waveform Editor. The nodes needed for simulation.

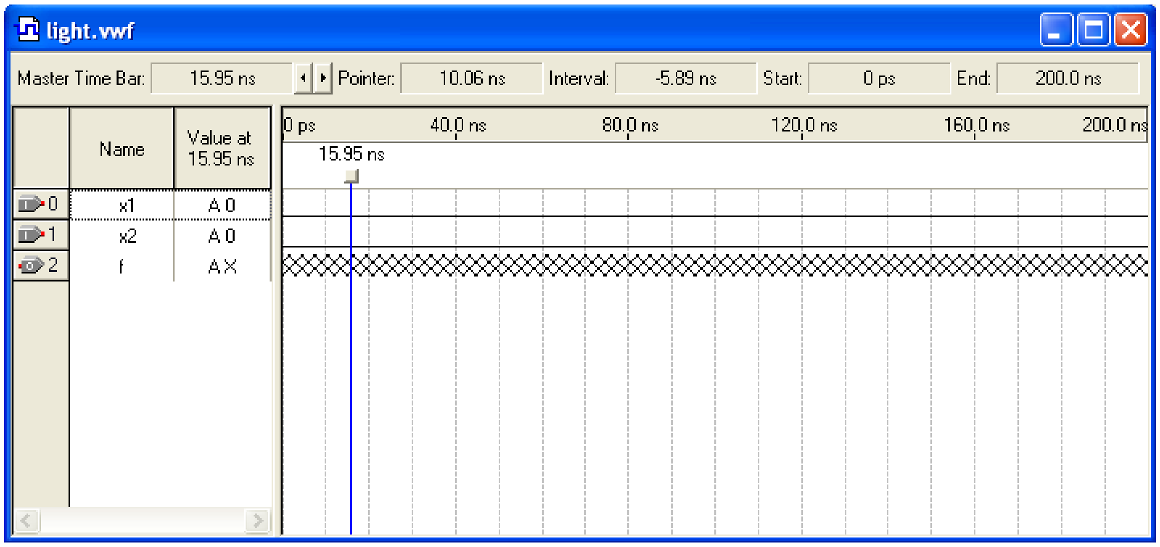


Figure 10. The nodes needed for simulation.

We will now specify the logic values to be used for the input signals *x1* and *x2* during simulation. The logic values at the output *f* will be generated automatically by the simulator. To make it easy to draw the desired waveforms, the Waveform Editor displays (by default) vertical guidelines and provides a drawing feature that snaps on these lines (which can otherwise be invoked by choosing **View > Snap to Grid**). Observe also a solid vertical line, which can be moved by pointing to its top and dragging it horizontally. This reference line is used in analyzing the timing of a circuit; move it to the *time* = 0 position. The waveforms can be drawn using the **Selection Tool**, which is activated by selecting the icon in the toolbar, or the **Waveform Editing Tool**, which is activated by the icon.
For our tiny circuit we can simulate all four input valuations and we will use four 50-ns time intervals to apply the four test vectors.

We can generate the desired input waveforms as follows.

1. Click on the waveform name for the *x1* node.
2. Once a waveform is selected, the editing commands in the Waveform Editor can be used to draw the desired waveforms. Commands are available for setting a selected signal to 0, 1, unknown (X), high impedance (Z), don't care (DC), inverting its existing value (INV), or defining a clock waveform. Each command can be activated by using the **Edit > Value** command, or via the toolbar for the Waveform Editor.
3. Set *x1* to 0 in the time interval 0 to 100 ns, which is probably already set by default.
4. Set *x1* to 1 in the time interval 100 to 200 ns. Do this by pressing the mouse at the start of the interval and dragging it to its end, which highlights the selected interval, and choosing the logic value 1 in the toolbar.
5. Make *x2* = 1 from 50 to 100 ns and also from 150 to 200 ns, which corresponds to the truth table in Figure 11. This should produce the image in Figure 11.
6. Observe that the output *f* is displayed as having an unknown value at this time, which is indicated by a hashed pattern; its value will be determined during simulation.
7. Save the file.

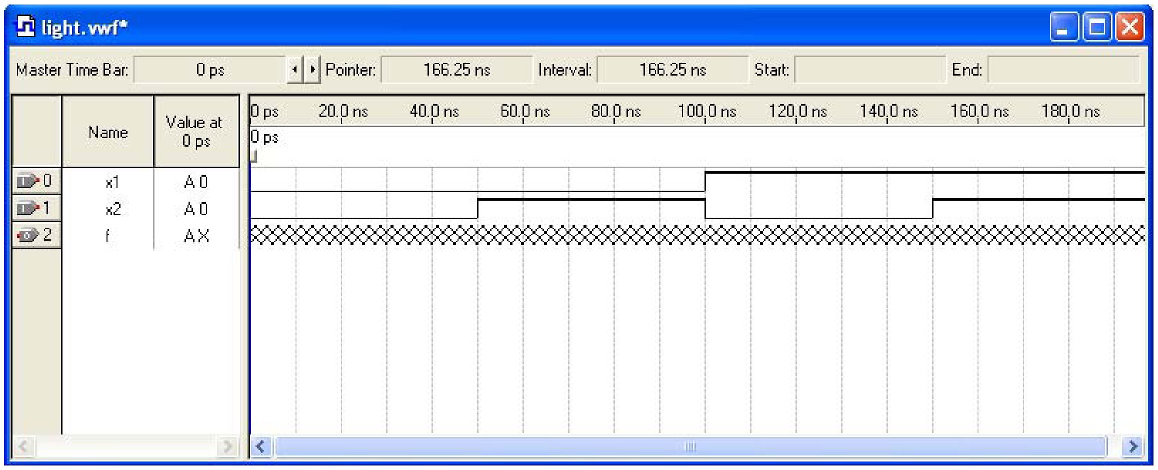


Figure 11. Setting of test values.

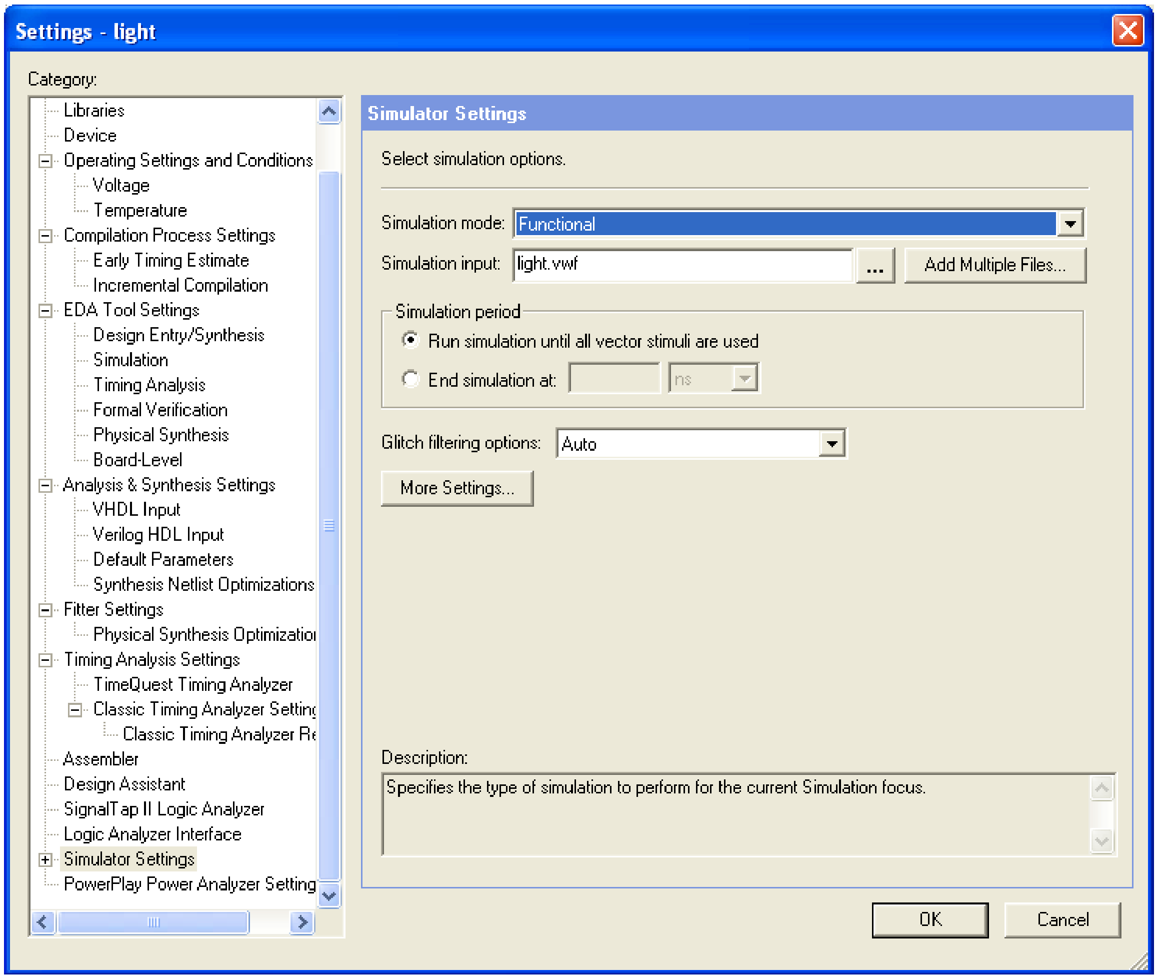
Performing the Simulation

A designed circuit can be simulated in two ways. The simplest way is to assume that logic elements and interconnection wires in the FPGA are perfect, thus causing no delay in propagation of signals through the circuit. This is called *functional simulation*. A more complex alternative is to take all propagation delays into account, which leads to *timing simulation*. Typically, functional simulation is used to verify the functional correctness of a circuit as it is being designed. This takes much less time, because the simulation can be performed simply by using the logic expressions that define the circuit.

Functional Simulation

To perform the functional simulation,

1. Select **Assignments > Settings** to open the **Settings** window.
2. On the left side of this window click on **Simulator Settings** to display the window in Figure 12, choose **Functional** as the simulation mode, and click **OK.**
3. The Quartus II simulator takes the inputs and generates the outputs defined in the **light.vwf** file.
4. Before running the functional simulation it is necessary to create the required netlist, which is done by selecting **Processing > Generate Functional Simulation Netlist.**
5. A simulation run is started by **Processing > Start Simulation**, At the end of the simulation, Quartus II software indicates its successful completion and displays a Simulation Report illustrated in Figure 13. If your report window does not show the entire simulation time range, click on the report window to select it and **choose View > Fit in Window.**
6. Observe that the output *f* is as specified in the truth table of Figure 13.



Settings - light

Category:

Simulator Settings

Select simulation options.

Simulation mode: Functional

Simulation input: light.vwf

Simulation period

(●) Run simulation until all vector stimuli are used

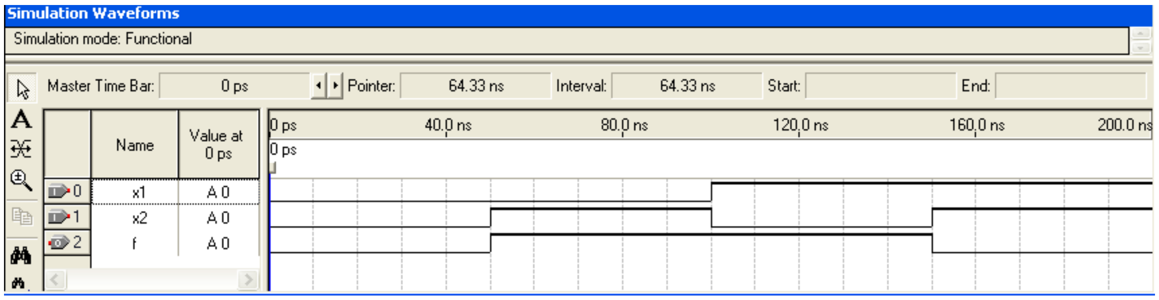
( ) End simulation at: ns

Glitch filtering options: Auto

Description:

Specifies the type of simulation to perform for the current Simulation focus.

Figure 12. Specifying the simulation mode.



Simulation Waveforms

Simulation mode: Functional

Master Time Bar: 0 ps   Pointer: 64.33 ns   Interval: 64.33 ns   Start: End:

	Name	Value at 0 ps
0	x1	A 0
1	x2	A 0
2	f	A 0

Figure 13. The result of functional simulation.

Part III –Simulation of a 3 bit adder

Experiment I

1. Create a new VHDL project.
2. Design a 3 bit adder using VHDL.

3. Simulate the circuit four using four different combinations of input vectors. When adding the inputs to simulate, add only the vectors (not the individual bits of each vector)
4. Include the simulation results in your report.

Post Lab Questions for lab 5

1) Explain the logic design flow with simulation.

Pre Lab Questions for lab 6

1) Draw the black box diagram for a 8:1 Multiplexer, and show its corresponding truth table.