

## VeriLogger Tutorial: Basic Verilog Simulation

This tutorial demonstrates the basic simulation features of VeriLogger Pro. It teaches you how to create and manage a project and how to build, simulate, and debug your design.

This is a stand alone tutorial which you should be able to complete without reading any of the other tutorials. You will compile and simulate a 4-bit adder and a test bench module contained in files add4.v and add4test.v.

```
//*****  
// VeriLogger Pro: Basic Verilog Simulation  
//*****  
  
//add4.v File *****  
/** Full Adder *****  
  
module fulladder(sum, c_out, x, y, c_in);  
    output sum, c_out;  
    input x, y, c_in;  
    wire a, b, c;  
    xor (a, x, y);  
    xor (sum, a, c_in);  
    and (b, x, y);  
    and (c, a, c_in);  
    or (c_out, c, b);  
endmodule  
  
/** 4-Bit Adder *****  
module FourBitAdder(sum, c_out, x, y, c_in);  
    output [3:0] sum;  
    output c_out;  
    input [3:0] x, y;  
    input c_in;  
    wire c1, c2, c3;  
    fulladder fa0(sum[0], c1, x[0], y[0], c_in);  
    fulladder fa1(sum[1], c2, x[1], y[1], c1);  
    fulladder fa2(sum[2], c3, x[2], y[2], c2);  
    fulladder fa3(sum[3], c_out, x[3], y[3], c3);  
endmodule
```

```

//*****
// VeriLogger Pro: Basic Verilog Simulation
//*****

//add4test.v File *****
module testbed();
    reg c_in;
    reg [3:0] y;
    reg [3:0] x;
    wire c_out;
    wire [3:0]sum;
    FourBitAdder A1(sum, c_out, x, y, c_in);
initial
begin
    x = 4'b0001; y = 4'b0001; c_in = 1'b0;
    #25 x = 4'b0001; y = 4'b0010;
    #25 x = 4'b0010; y = 4'b0011;
    #25 x = 4'b0001; y = 4'b1111;
    #25 x = 4'b0001; y = 4'b1111; c_in = 1'b1;
    #25 x = 4'b1000; y = 4'b1111; c_in = 1'b0;
    #25 x = 4'b0001; y = 4'b0001; c_in = 1'b1;
    #25 x = 4'b0001; y = 4'b0010;
    #25 x = 4'b0010; y = 4'b0011;
    #25 x = 4'b0011; y = 4'b1111;
    #25;
end
initial
    #250 $finish;
endmodule

```

In this tutorial, you will create, build, and simulate a project. VeriLogger uses a project to control all aspects of simulation and design including specifying the files to be simulated, controlling simulation options, and setting watches on signals. The project also stores the hierarchical structure of the Verilog components contained in the design and displays this information on the tree control in the Project window.

## 1. Add Files to the Project

In project pull-down menu select New Project. To add a file to the project:

- Choose Add Files from the menu
- Select the add4.v and add4test.v files. To select multiple files at the same time, select the first file then hold down the <CTRL> key while using the mouse to select any additional files.

- Press the Open button to add the files to the project. Both file names should be visible on the project tree. If you do not see both files then repeat the instructions and add the missing file to the project.

## **2. Build the Tree and use the Editor Windows**

In this section we will build the project tree and use the Editor windows to view the source code. When files are first added to the project, you can see the file name but you cannot see a hierarchical view of the modules inside the files. To view the internal modules on the project tree you must first build or run a simulation. The build command compiles the Verilog files and builds the Verilog tree. It does not run a simulation. For large projects build lets you quickly construct the tree without having to wait for a simulation to run. To build a project:

- Press the yellow Build button on the simulation button bar.
- Notice that the modules inside each source file are displayed on the project tree. One module, testbed, is surrounded by brackets to indicate that it is the top-level module (the highest-level instantiated component).
- Notice that the Diagram window now has several signals listed in the waveform display. By default, VeriLogger sets watches on all the top level modules internal signals. Later we will learn to set watches on the other signals.

All sub-modules can be viewed by descending the top-level module's tree. When the tree is expanded it can display the signals, ports, and components contained in each module. Expand the tree by using the + buttons or by using the node context menu Expand Item. Try both methods for expanding the tree:

- Press the + button to the left of <<testbed>> to expand the project tree. Notice that the sub-nodes of Signals and Components are not expanded.
- Right click on <<testbed>> and choose Expand Item from the context menu. This expands the node and all the sub-nodes.

Double clicking on a particular module or node in the Project Tree causes an editor to open and display the relevant source code. Let's view some source code:

- Double click on the add4test.v file name to open an editor loaded with that file. Notice that the editor is scrolled to the top of the file.

### 3. Simulate the Project

When we built the project in the last section, the names of the internal signals in the top-level module were automatically added to the Diagram window. This feature allows you to quickly set up a project and start simulating and debugging without having to stop and specify a set of signals. For large projects you may want to turn off this feature by choosing the Project > Project Settings menu and un-checking the Grab top level signals check box. For small projects the automatic signal watches save a lot of time so we will leave it on for the tutorial. First, let's simulate with the default signals:

- Press the green Run button on the simulation button bar. This causes a simulation to start and run until the end of the simulation time or until a breakpoint is reached. The Diagram window should contain purple waveforms.
- Verify that the sum and c\_out are correctly being computed as  $x + y + c\_in$ .

### 4. Watch and View Internal Signals

With VeriLogger you can watch any combination of signals listed under the top-level module tree. To demonstrate this we will set watches on the sum outputs for the full adders sub-modules that make up the 4-bit adder:

- In the Project window, expand the top-level module tree and find the fa0 component.
- Right click on the sum port for fa0 to open a context menu.
- Choose the Watch Connection menu option. This adds the testbed.A1.fa0.sum signal to the Diagram window.
- Press the green Run button to run another simulation. Verify that the testbed.A1.fa0.sum signal is the 0 bit of the testbed.sum[3:0] signal.

To remove signals from the watch list, delete them from the Diagram window:

- In the Diagram window, left click on the testbed.A1.fa0.sum signal name to highlight it.
- Press the Delete key on the keyboard to remove the signal from the Diagram window.
- Press the green Run button to run another simulation. Verify that the testbed.A1.fa0.sum signal was not watched.

Next we will experiment with different ways to view waveforms in the Diagram window:

- In the time line above the signals in the Diagram window, left click down and hold to show a marker that displays the value of each signal. Release the mouse button without dragging.
- Left click and drag the marker about 50ns in the time line window. When you release the mouse button, the window will zoom to display the time range that the mouse was dragged over.
- Right click in the time line to zoom out on the waveforms.
- Press the Zoom Full button on the Diagram window to return the zoom level to the entire simulation range.

## **5. Save the Project and Source Code**

Next we will learn to save the project and source code. The project saves the simulation options and the names of the files contained on the project tree. It does not save the source code or the watched signals. To save the project:

- Choose the Project > Save HDL Project menu option to open the Save Project As dialog.
- Type add4test.hpj into the File name edit box.
- Click the Save button to close the dialog. Notice that the name of the project is displayed in the titled bar of the project window. Note that, in the evaluation version of VeriLogger you cannot save the waveforms, but you can save the HDL code and the project.