Cadence Tutorial 3

Running Verilog-XL Simulation

EE577b Spring2000

In this tutorial, you will run a Verilog simulation on the function cellview of your 8-bit adder. You will read the functional cellview and begin Verilog Integration from this cellview.

1. Tutorial Setup

1. Finish the cadence tutorial 2 before you start this tutorial.

2. If you haven't finished the tutorial 2 or you want to use TA's design, follow the steps below.
   - `%cp ~ee577/cadence_ee577b_adder8.tar ~/cds` (or your working directory)
   - `%cd ~/cds`
   - `%tar xvf cadence_ee577b_adder8.tar`
   - Add the following line in your "cds.lib" file.
     `DEFINE Adder8 ~/cds/Adder8`
3. Even if you have finished tutorial 2, you need to make a symbol for adder8 before starting this tutorial.
4. Invoke "icfb"

2. Create the 8-bit Adder Functional Cellview

1. Open *adder8* schematic from *Adder8* library .
2. sch : **Design->Create Cellview->From Cellview**
3. In [Cellview from Cellview] window,
   Library Name : *Adder8*
   Cell Name : *adder8*
   From V iew Name :*schematic*
   To View Name : *functional*
   Tool/Data Type : *Verilog-Editor*
4. You will see the following codes opened in Text Editor with name "verilog.v"

```
module adder8 (Cout, S, A, B, Cin);
     output Cout;
     output [7:0] S;
     input [7:0] A;
     input [7:0] B;
     input Cin;
endmodule
```

   - This verilog.v file is located at
     `~/cds/Adder8/adder8/functional.` Use your favorite editor if you wish.

jsmoon@ISI.EDU

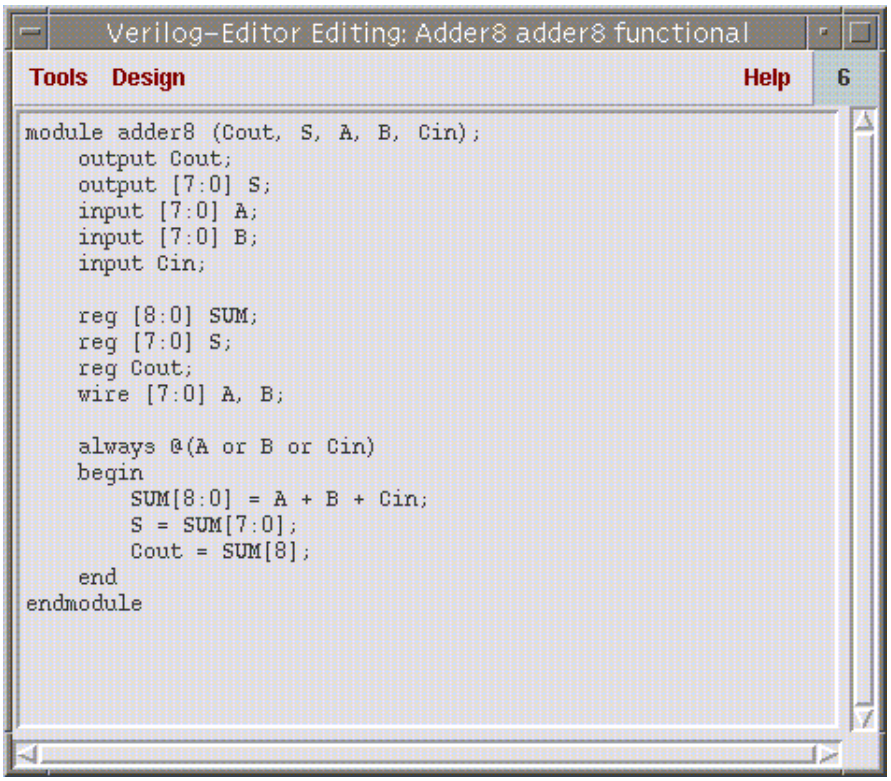
5. Type the following lines to finish functional description.

```
module adder8 (Cout, S, A, B, Cin);
     output Cout;
     output [7:0] S;
     input [7:0] A;
     input [7:0] B;
     input Cin;

     reg [8:0] SUM;
     reg [7:0] S;
     reg Cout;
     wire [7:0] A, B;

     always @(A or B or Cin)
     begin
          SUM[8:0] = A + B + Cin;
          S = SUM[7:0];
          Cout = SUM[8];
     end
endmodule
```

6. Save the file.

3. Open the Functional Cellview of the 8-bit Adder

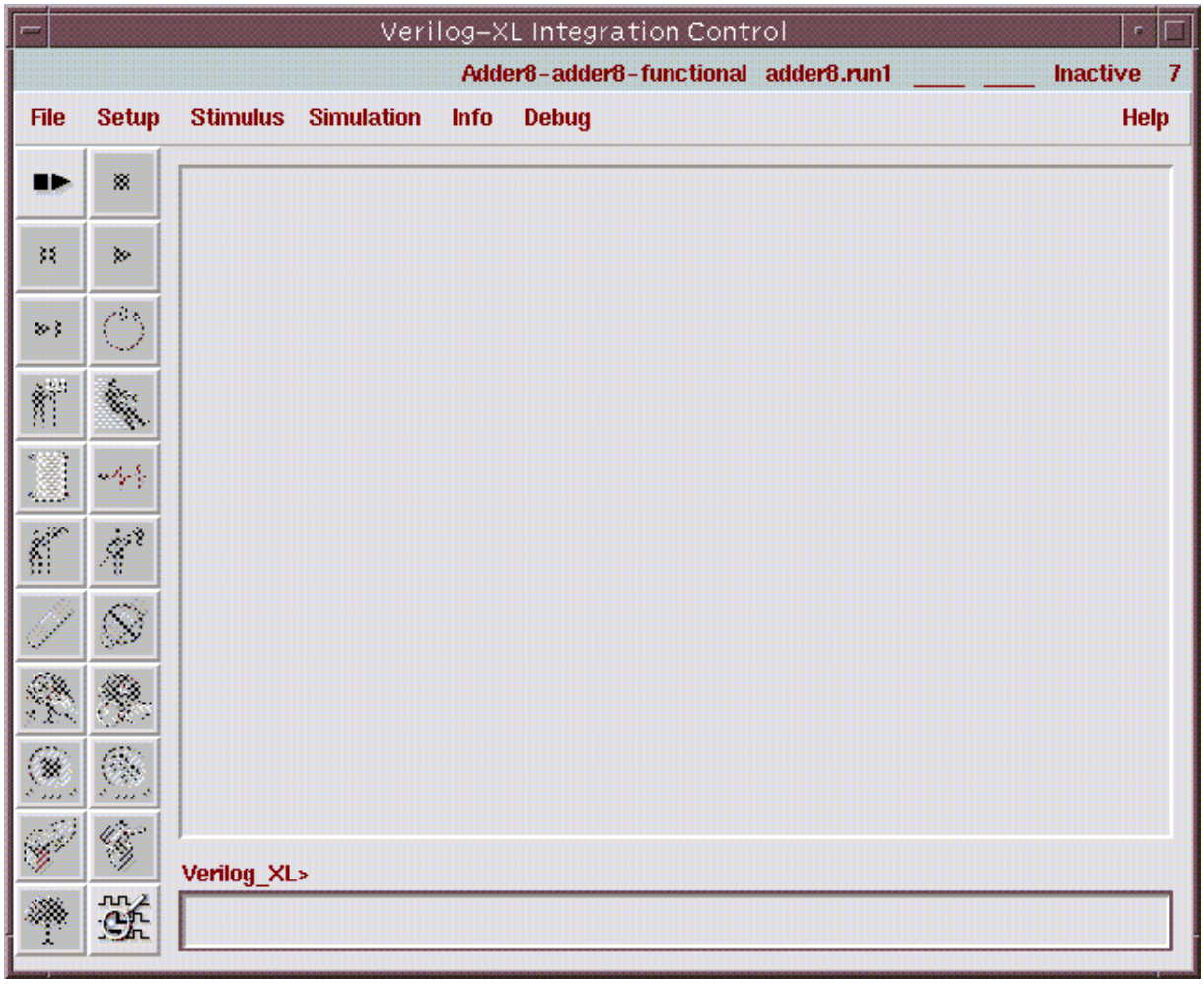
1. From the library browser, read the *adder8* functional cellview from the *Adder8* library.
   - The *adder8* functional cellview must be opened in read mode rather than edit.
2. The text window displays the *adder8* functional cellview.



jsmoon@ISI.EDU

4. Initializing Verilog Integration

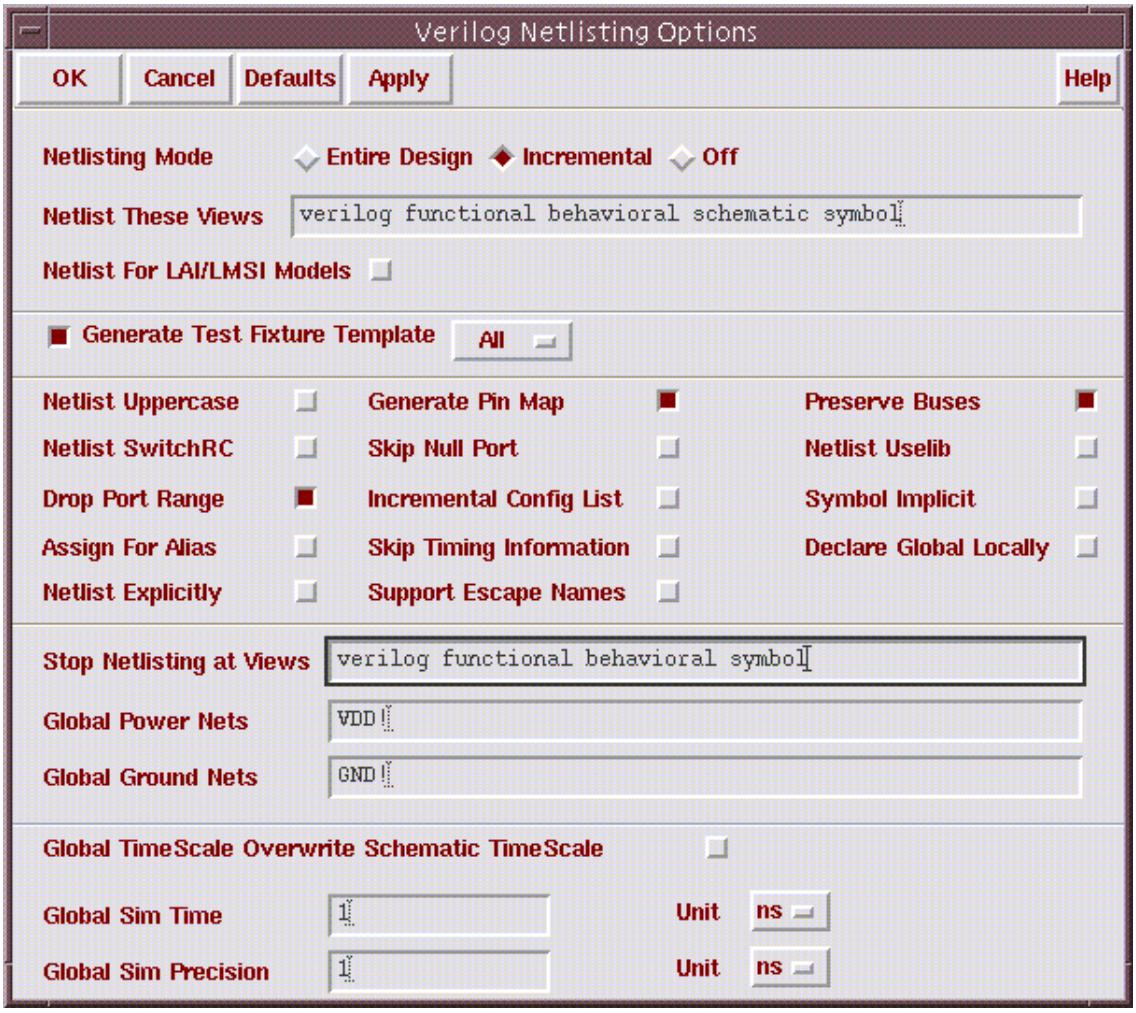
1. Start Verilog Integration by selecting **Tools->Verilog-XL** from the *adder8* window.
   The Verilog-XL Setup Environment form is displayed.
2. In the Verilog-XL Setup Environment form, enter *adder8.run1* for the run directory.
   All other default values are correct.
3. Click on OK.
4. The Verilog-XL Integration Control window is opened. The *adder8.run1* directory is created and the environment is initialized.



5. Setting the Netlist and Waveforms Options

1. In the Verilog window, set the Netlisting Options by selecting **Setup->Netlist**.
   The Netlisting Options form is displayed.
2. In the Netlisting Options form, set **Netlist These Views** list to:
   `verilog functional behavioral schematic symbol`
3. Click on **More**>>.
   Additional netlisting options are added to the form.
4. Define **Stop Netlisting at Views** as:
   `verilog functional behavioral symbol`
5. Click on **Generate Pin Map** to turn it on.
6. Click on OK.

jsmoon@ISI.EDU



6. Creating a Stimulus File

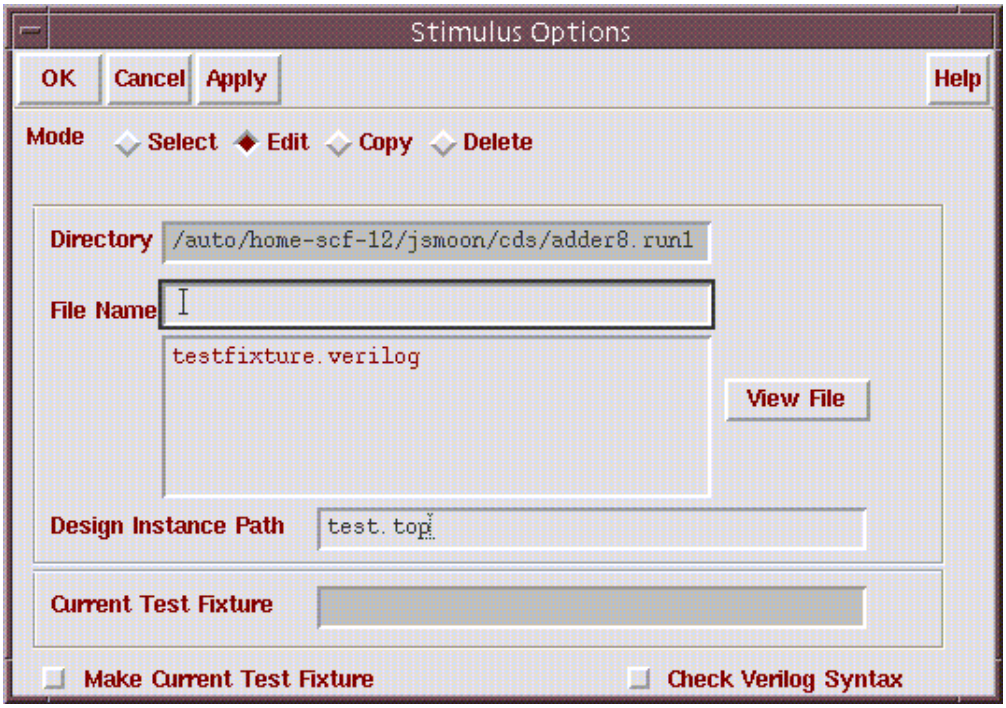
The stimulus file used for this simulation will nest two for loops which increment A and B from 0 to 256.

1. In the Verilog window, create a Test Fixture file by selecting **Stimulus->Verilog.**
2. Click on Yes in the dialog box that appears.

   This step produces a netlist of the design and a test fixture template in the run directory. The netlist is required prior to the creation of the template test fixture file. Then the Stimulus Option form is displayed.
3. Click on Edit for mode and testfixture.verilog for File Name in Stimulus Option form.
4. Click on OK.

   A text editor window displays the test fixture template. The test fixture template sets all input values to zero. You must add the simulation input vectors to this file. Complete the test fixture file as shown below.

jsmoon@ISI.EDU




```
integer counta, countb;
initial
begin
     A[7:0] = 8'b00000000;
     B[7:0] = 8'b00000000;

     Cin = 1'b0;

     for(counta=0;counta<=256;counta=counta+1)
     begin
          A=counta;
          for(countb=0;countb<=256;countb=countb+1)
               #20 B=countb;
     end

     Cin = 1'b1;

     for(counta=0;counta<=256;counta=counta+1)
     begin
          A=counta;
          for(countb=0;countb<=256;countb=countb+1)
               #20 B=countb;
     end
end
```

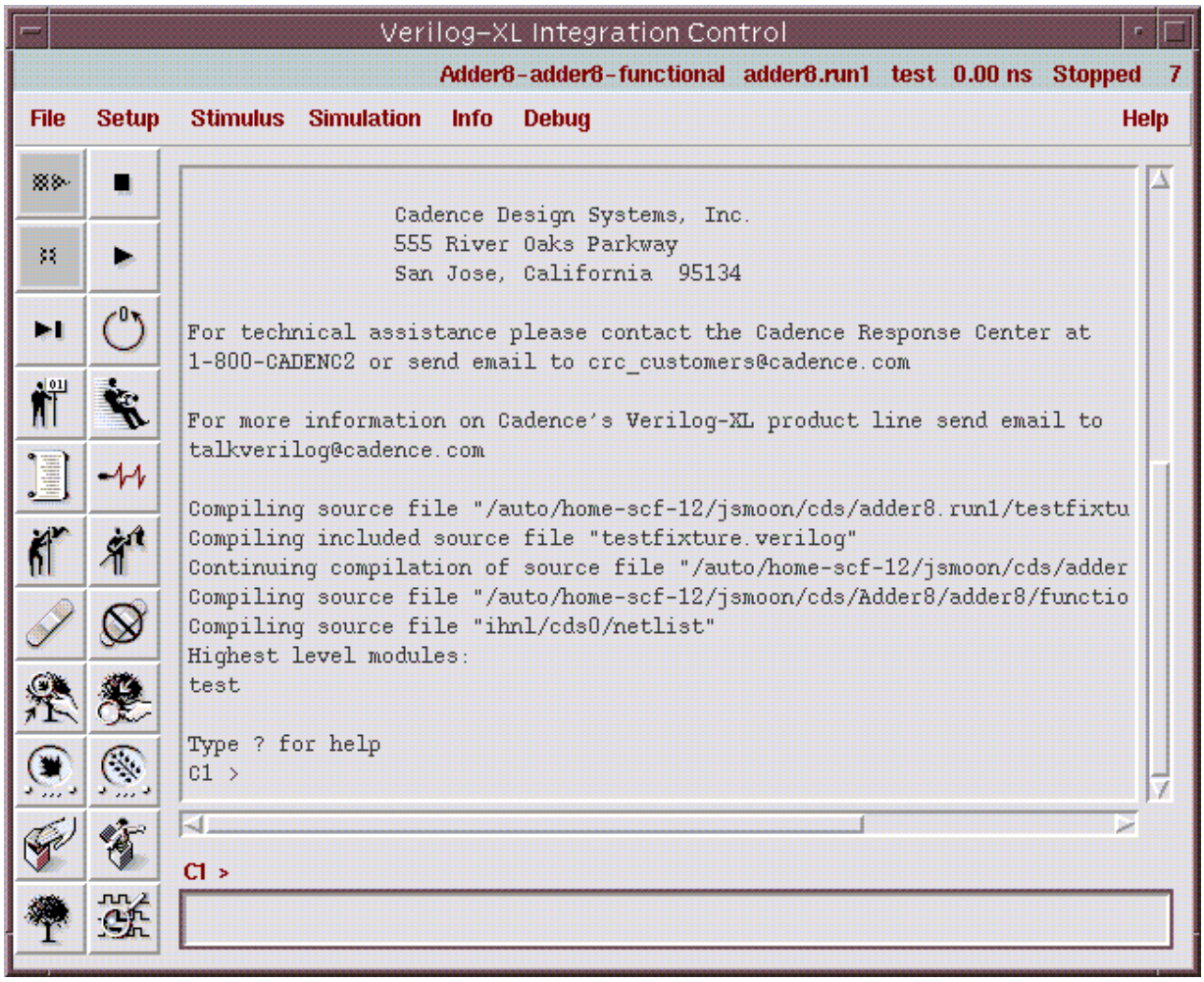
5. Save the file and quit the editor.

jsmoon@ISI.EDU

7. Starting the Simulation

1. Start the simulation in the interactive mode by selecting the **Start Interactive** fixed menu command.

   The Verilog-XL simulator is started and it compiles the files associated with this simulation run. The simulator is initialized in the interactive mode and stops at stimulation time=0.



8. Adding Breakpoints to the Simulation

Breakpoints are used to interrupt the simulation at predetermined times or events. In the steps below, youwill add breakpoints at specific simulation times.

1. Select **Debug->Add Breakpoints**
2. In the Set Breakpoint for, add absolute time breakpoints before the following times:

   ```
   100000 200000 300000
   ```

3. Click on OK to create the breakpoints.

9. Waveform Window

Refer to Simwave tutorial.

10. Continuing Simulation

1. In the Verilog-XL Integration window, select the fixed menu **Continue** command.
   The simulation continues to time 100000 (the first breakpoint).
2. Continue the simulation to time 300000.

11. Quitting the Verilog Integration

1. In the Verilog-XL Integration window, select **File->Quit**.
2. Close the adder8 function cellview window by selecting **Design->Close Window**.

jsmoon@ISI.EDU

Special Note : How to see waveform using SimWaves

1. Create *wave* directory under *adder8.run1* directory.

2. Copy all necessary verilog file from *adder8.run1* directory.
   - *testfixture.template* (topmost file)
   - *testfixture.verilog* (test bench file)

3. In testfixture.template file, include all necessary verilog description of design. In our example, we only need
   `~/cds/Adder8/adder8/functional/verilog.v`
   Include the following line before module declaration.
   `'include "../../Adder8/adder8/functional/verilog.v"`

   In testfixture.template file, you need to add lines `include ...
   after `timescale 1ns / 1ns line.
   For example, in your testfixture.template file for the functional simulation,

   `timescale 1ns / 1ns
   `include "../../Adder8/adder8/functional/verilog.v"

   If you swap the order, you will get the error message like

   Error! Module (test) has a `timescale directive
   but
   previous modules do not
   [Verilog-MODTDN]
   "testfixture.template", 5: module test;

4. In testfixture.verilog file, include the following initial block to save waveform so that we can see it using simWaves. (You know it already!!)

```
initial
     begin
          $shm_open("signals.shm");
          $shm_probe("AS");
#300000   $shm_close();
          $finish;
end
```

5. Compile *testfixture.template* file using "verilog".
   If succeeded, you will see signals.shm directory created.

6. You guys know how to use simWaves already, don't you?