Preliminary Report for EE894Z

Brian Dupaix
February 26, 2004

Introduction

The goal for my portion of the Design Center Group was to look at tools that provided mixed-mode simulation capability with the following constraints. First, the tool should work on text-based input files. Second, the tool should be one that is readily available at OSU. Third, the tool should have capabilities of simulating Verilog or VHDL along with Verilog-A, Verilog-AMS, and/or VHDL-AMS.

Investigation

The tools I chose to look into were the Cadence tools since the MISES group had already put a large amount of effort into creating a front-end capture and back end synthesis and layout flow using the Cadence tools. It was my assumption that by using a tool from the same vendor, some of the potential interfacing and integration problems could be avoided.

The tool I decided to explore operation of is the Cadence Analog Mixed-Signal Simulator(AMS). This tool allows an existing schematic design to be loaded into the simulator and connected to digital blocks and simulated as well as providing the capability of creating new analog blocks in Verilog-A or Verilog-AMS and simulating. The tool is also supposed to allow analog models written in VHDL-AMS to be simulated, but the current version of the Logic Design and Verification (LDV) package (version 4.1) does not allow for VHDL-AMS compilation and simulation.

In the process of creating a simulation, a basic understanding of how the tool works was discovered and those items are discussed in this document.

Design Creation

The AMS tool can be invoked on a text-based design netlist or a schematic based design. To use a text-based design, the design must be first compiled into a cadence library format so the Hierarchy Editor used to invoke the simulator will be able to recognize the required entities. Note that Cadence also provides an option to directly simulate a text based design using the **ncverilog** command, but I have not investigated that yet.

An existing design can be compiled into a Cadence-format library by using the following commands:

```
ncvhdl [–ams] -use5x X.vhd
ncvlog [-ams] -use5x X.v[ams]
```

The –ams switch is used to compile Verilog-A or Verilog-AMS files. These files should have a .vams extension. The –use5x switch tells the compiler to create a Cadence-format library element from the code being compiled. The library the new design is compiled into is determined by the hdl.var file in the directory the files are being compiled in. The hdl.var file contains a line defining libraries like the following:

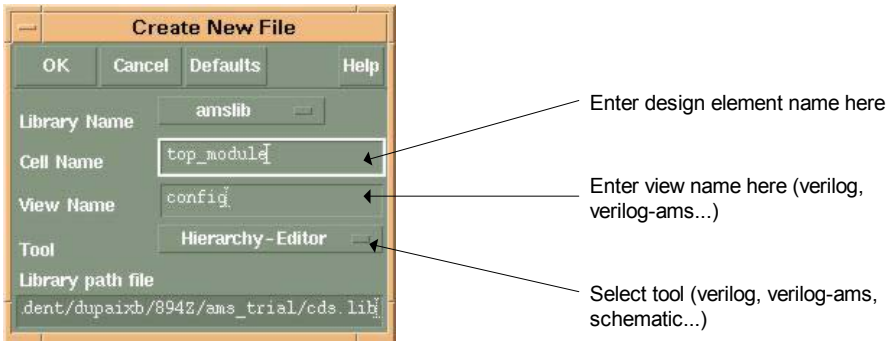
```
Define WORK diglib
```

'diglib' is a directory defined in the cds.lib file in the compile directory as shown below.

```
DEFINE diglib /rcc4/student/dupaixb/894Z/ams_trial/diglib
```

Once a design is compiled into a library, it can be instantiated in a schematic through the normal Cadence instantiation process.

A design can also be created from the **icfb** tool by using the File->Create New->Cellview pulldown menu, naming the design element, and selecting the corresponding view type (verilog, verilog-A, verilog-AMS, VHDL, schematic ...)



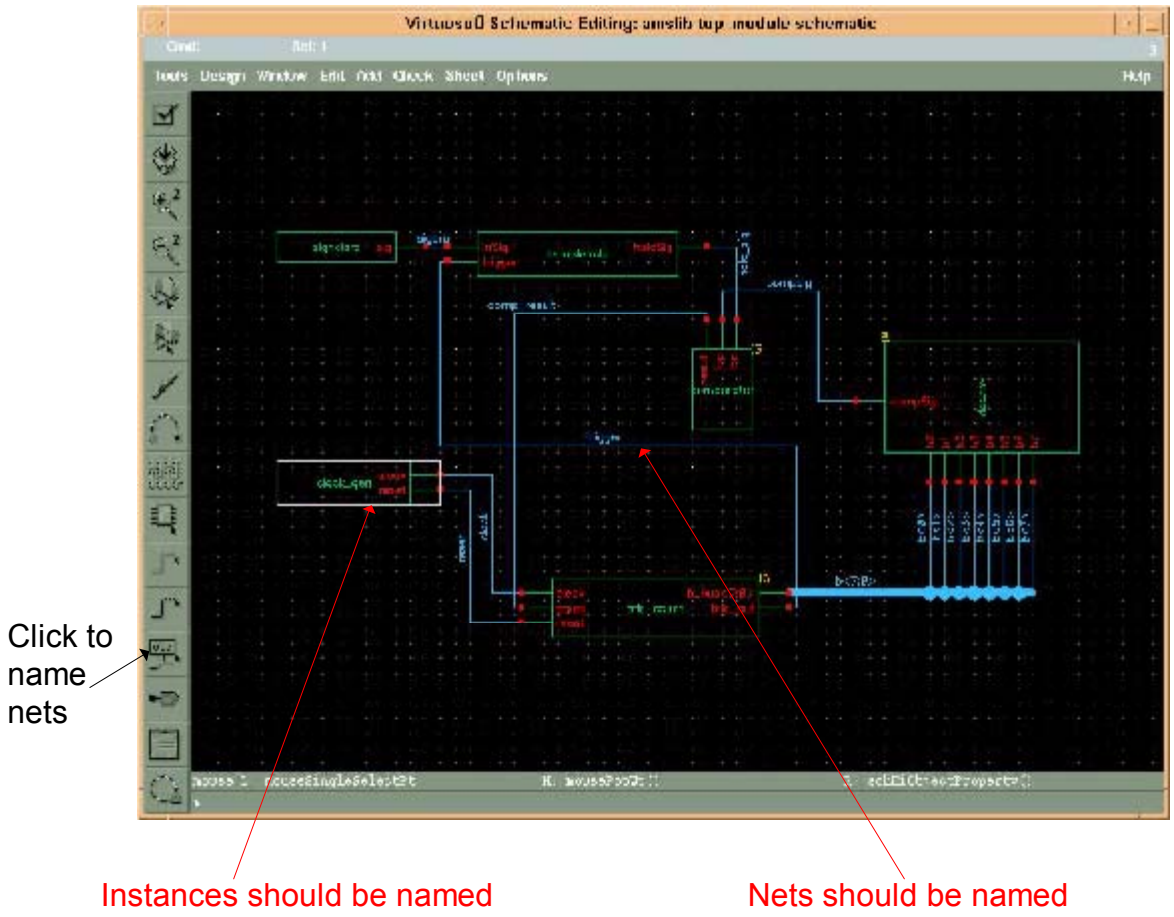
The design can then be entered as any of the available view types.

For this project, I created a schematic view based on the original Cadence tutorial. Not that if a schematic is being created, all nets and instances should be named. If instances

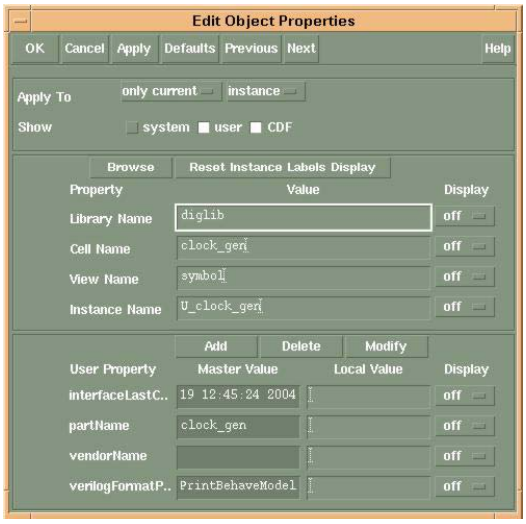
are not named, Cadence gives them a name starting with I making it harder to debug the traverse the design hierarchy and select waveforms to view(as will be shown later). The following netlist was generated by Cadence from a schematic and illustrates the problem. Instance names are shown in bold (to highlight them).

```
// Verilog-AMS netlist generated by the AMS netlister, version 5.0.32.57.
// Cadence Design Systems, Inc.

`include "disciplines.vams"

module top_module (  );
wire [7:0] b;


trig_count  (* integer library_binding = "diglib";  *) I6 ( .equal(
comp_result ), .clock( clock ), .reset( reset ), .trig_out( trigger ),
.b_bus( b[7:0] ) );

clock_gen  (* integer library_binding = "diglib";  *) U_clock_gen ( .clock(
clock ), .reset( reset ) );

comparator  (* integer library_binding = "amslib";  *) I3 ( .inn(
hold_sig ), .inp( compsig ), .result( comp_result ) );

samplehold  (* integer library_binding = "amslib";  *) U_samplehold (
.inSig( sigsrc ), .holdSig( hold_sig ), .trigger( trigger ) );

signalsrc  (* integer library_binding = "amslib";  *) U_signalsrc ( .sig(
sigsrc ) );

daconv  (* integer library_binding = "amslib";  *) I0 ( .b2( b[2] ), .b5(
b[5] ), .b6( b[6] ), .b3( b[3] ), .compSig( compsig ), .b7( b[7] ), .b0(
b[0] ), .b4( b[4] ), .b1( b[1] ) );

endmodule
```

The schematic for the design is shown below.



Instance names are changed by selecting an instance and selecting Edit->Properties after which the following dialog box will appear.

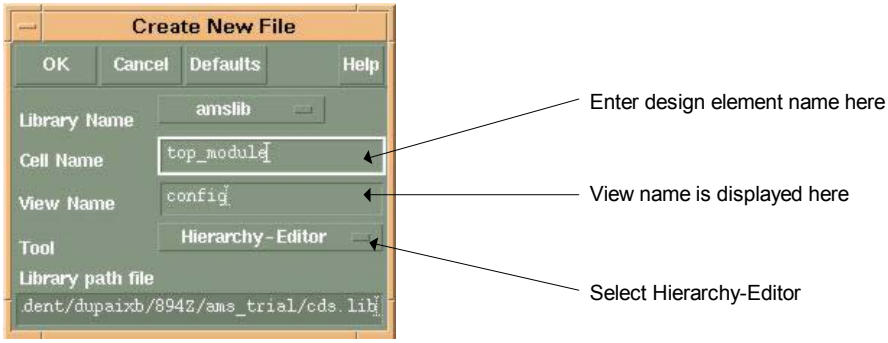


In this box, the instance name U_clock_gen is being added to the clock_gen symbol view and is reflected in the generated netlist:

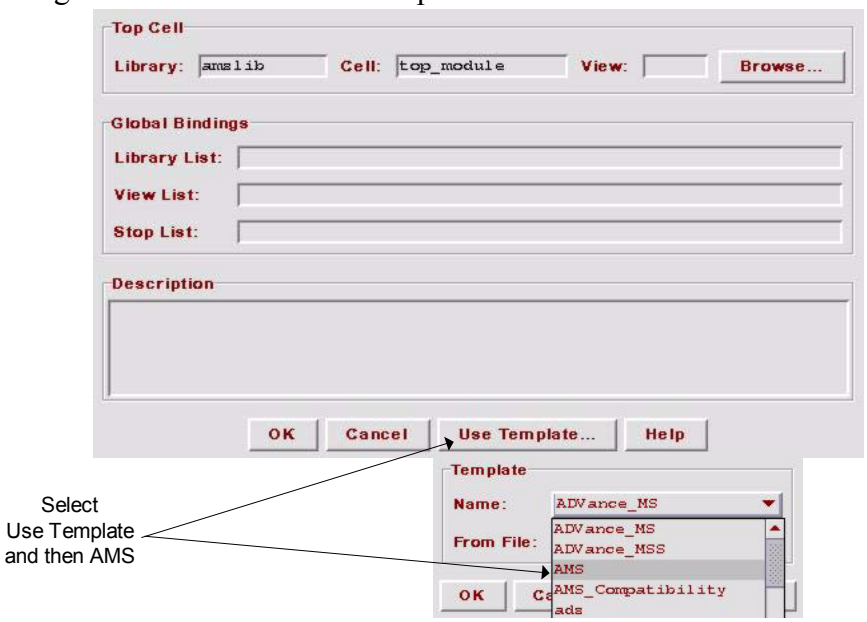
```
clock_gen  (* integer library_binding = "diglib";  *) U_clock_gen ( .clock(
clock ), .reset( reset ) );
```

Design Configuration

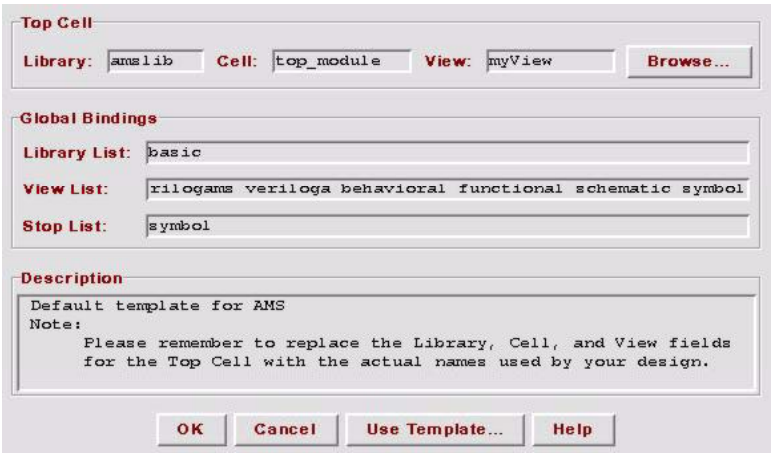
Once the design has been entered, it needs to be configured before it can be simulated. To configure the design, invoke the Hierarchy-Editor by using the File->New->Cell View pulldown from the **icfb** tool. The following dialog box will appear and you should enter the Cell Name and select the Hierarchy-Editor tool as shown in the following picture.



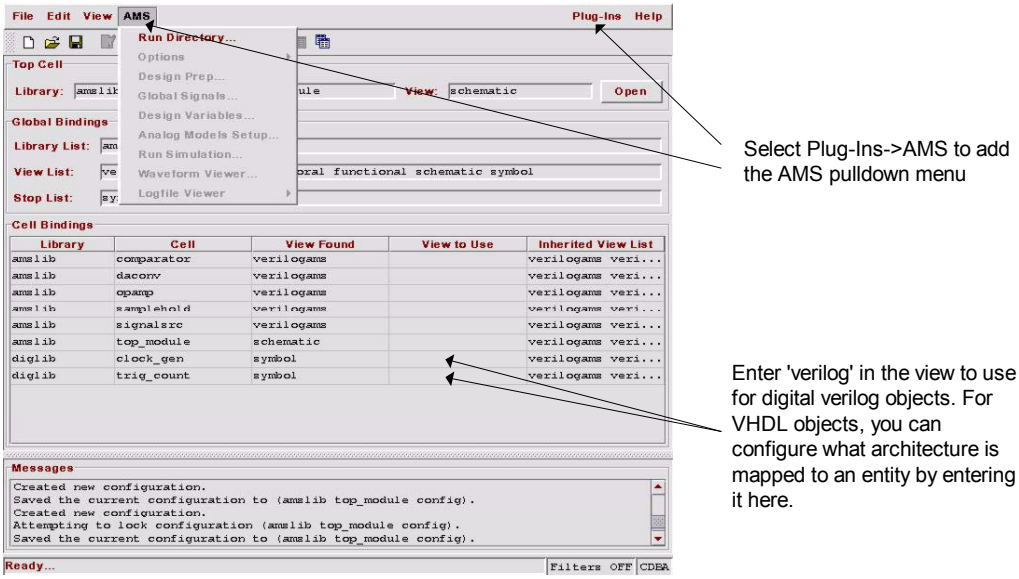
Once the Hierarchy-Editor has started, it will popup a window asking for configuration for a target tool. Select the Use Template button and select the AMS tool.



The resulting configuration setup should look like the following diagram. If it does not, select File-> New in the Hierarchy-Editor and create a new configuration.



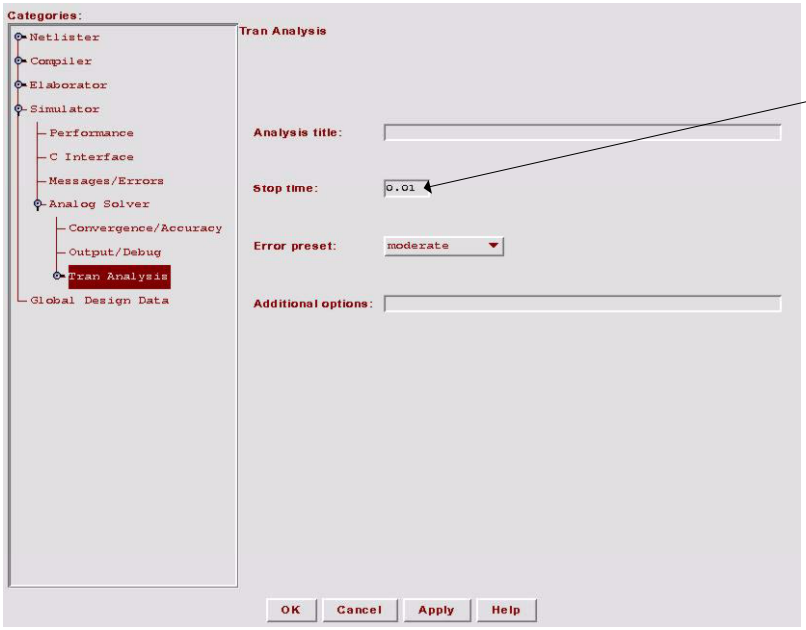
Select OK and you will enter the Hierarchy-Editor tool configured to run AMS. However, configuration is not complete. The Hierarchy Editor allows the user to select what view to simulate and we need to specify this for Verilog and VHDL files. Also, the Hierarchy-Editor does not automatically load the AMS menu required to run simulation. The menu is loaded by selecting the Plug-Ins pulldown on the toolbar and selecting AMS. An AMS menu is added to the pulldown menu list and looks like the following:



Select Plug-Ins->AMS to add the AMS pulldown menu

Enter 'verilog' in the view to use for digital verilog objects. For VHDL objects, you can configure what architecture is mapped to an entity by entering it here.

After selecting Run Directory and entering a directory path build the simulation database and run simulations in, the design is now ready to have simulation parameters set up. One critical parameter is the stop time for transient analysis. By default, this value is 0.0 and the design will not simulate, so it must be set by selecting AMS->Options and traversing the options hierarchy simulation like the following:

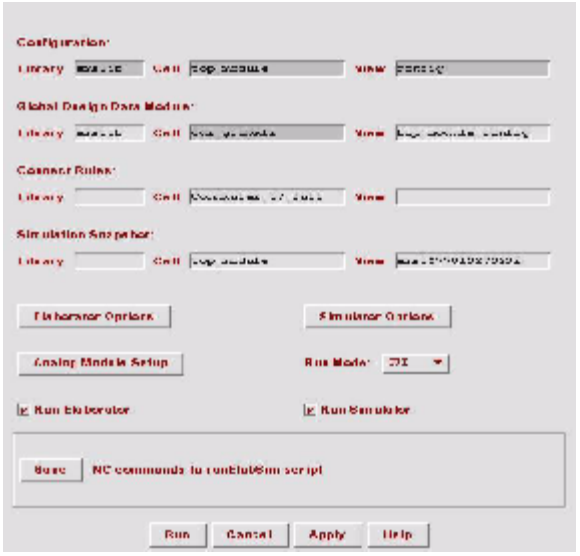


Enter a non-zero value for Transient Analysis stop time.

Once the Transient Analysis stop time is configured, the design is ready for simulation. First, run Design Prep by selecting AMS->Design Prep... from the pulldown menu. Click OK on the dialog box that comes up and Design Prep will run. The result should look like the Design Prep Results below.

Design Prep Dialog Box

Design Prep Results

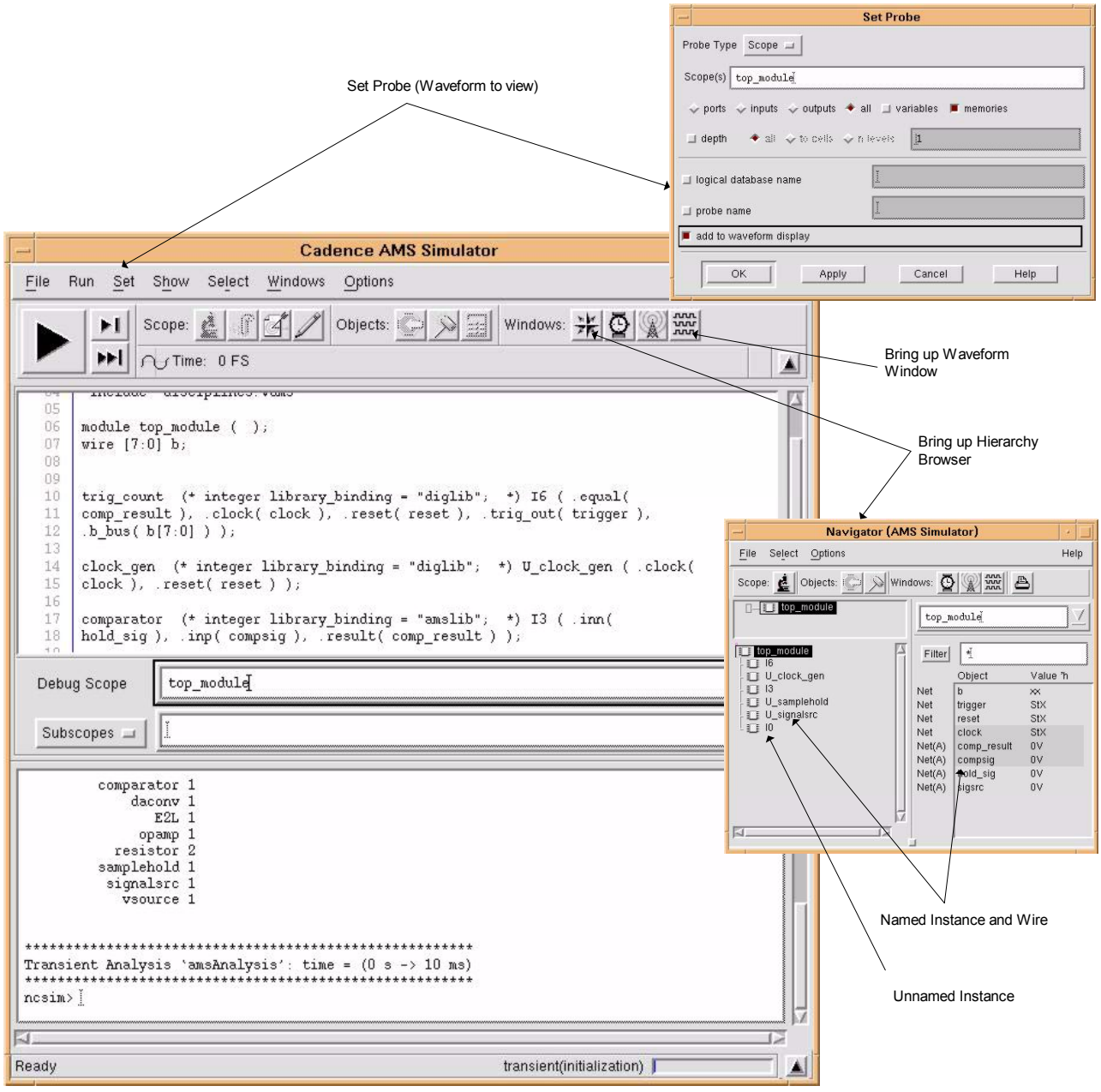


Run Simulation Dialog Box

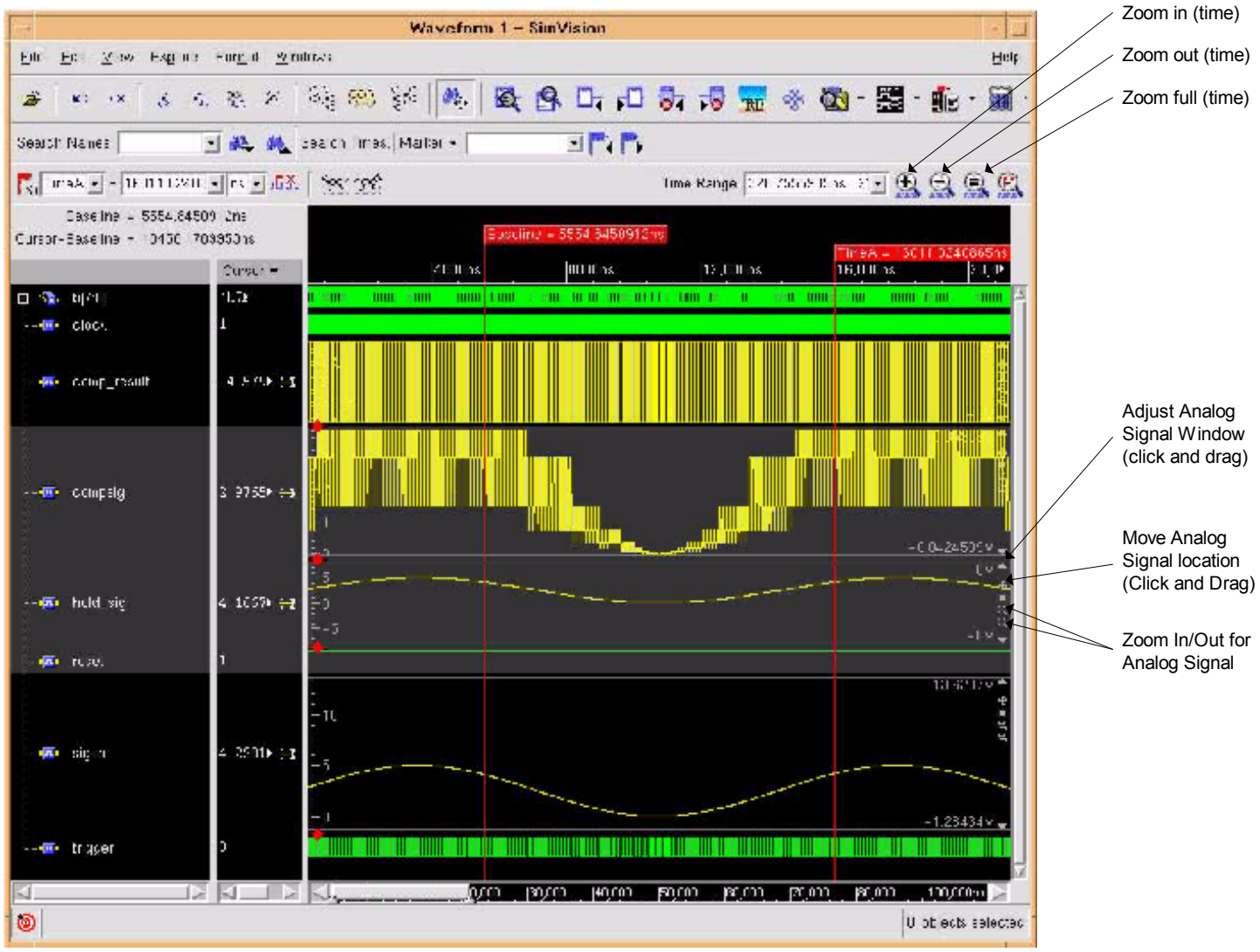
Once Design Prep has run successfully, select AMS->Run Simulation from the pulldown menu and click on the Run button as shown in the dialog box above. The simulator will start and bring up its GUI.

Simulation

The simulator GUI looks like the figure below. Before simulation is run, the simulator needs to be configured to save off waveforms to view for debugging the design. To accomplish this, probes are set on ports and/or wires in the design hierarchy for the simulator to capture waveform for. This is accomplished by using the Set->Probe pulldown in the GUI dialog box.



The waveform window looks like the following diagram. Some button functions are annotated.



Problems

The first problem I encountered was that the Startup.EE file which sets up the environment for the Cadence tools was not able to support AMS. Also, in the process of getting AMS to run, a variable set to limit 64-bit simulation was found to cause the ncsim tool (used for verilog, AMS, and VHDL simulations) to run in 64-bit mode. Since 64-bit mode is not currently supported by the AMS tool, this variable was removed from the Startup.EE file.

Once the environment was set up correctly, a tutorial was run to test the functionality of the AMS simulator. The tutorial included Verilog, Verilog-AMS, VHDL and VHDL-AMS code and a Cadence Schematic. The tutorial did not work because the AMS tool was unable to compile the VHDL-AMS code provided. This issue will be resolved in the pending release of the LDV5.1 toolset.

Future Work

There are several issues still to be resolved.

- First is the proper use of connection modules (which may have to wait until we receive LDV5.1).
- Second is running an actual useful design through the tool and using ADS to debug the design.
- Third is capturing the simulator setup including probe setting, allowing multiple restarts of the simulator while recording the same signals in the waveform database.
- Fourth is exploring invoking the simulator from the command line.