Introduction to AMS Designer Simulation

Abstract

This example uses a simple test case consisting of an inverter chain to show the setup and use of AMS Designer. Both the GUI driven flow with ADE L and the text based command line flow are shown. The steps to setup an AMS simulation in ADE L are discussed. The ADE AMS Error Explanation tool will be used to show how to resolve simulation setup errors. The steps needed to build the config view used in AMS simulation are illustrated. The schematic based test case will be migrated to command line simulation to give an overview of both the runams and irun AMS command line based simulation flows.

Prerequisites

- Experience running simulations with ***Virtuoso Analog Design Environment*** (ADE L).

Software and License Requirements

To run this example the following software packages are required:

- **INCISIV 13.2** (or higher) for the AMS simulator.
- **IC616 ISR 6** (or higher) for ADE L and VSE L.

License Requirements

- AMS Simulator
- ADE L
- VSE L (schematic editor)

AMS can be licensed in one of two ways

1) MMSIM tokens (Cadence product # 90003)
2) AMS Designer with Flexible Analog Simulation (Cadence product # 70020) + Virtuoso Spectre Simulator (Cadence product # 38500)

Analog Design Environment L is also used in this example which requires one of the following licenses to be available:

- Virtuoso Analog Design Environment L (Cadence product # 95200)

- Virtuoso Analog Design Environment XL (Cadence product # 95210)
- Virtuoso Analog Design Environment GXL (Cadence product # 95220)

VSE L can be licensed with either the “L” or “XL” license

- Virtuoso Schematic Editor L (Cadence product # 95100)
- Virtuoso Schematic Editor XL (Cadence product # 95115)

While we are discussing software and licensing, there is an important distinction between the MMSIM license (token license) and the MMSIM installation package.

The MMSIM license enables simulation with AMS, Ultrasim, APS, Spectre and SpectreRF. However, the MMSIM software is not required to be installed to run AMS.

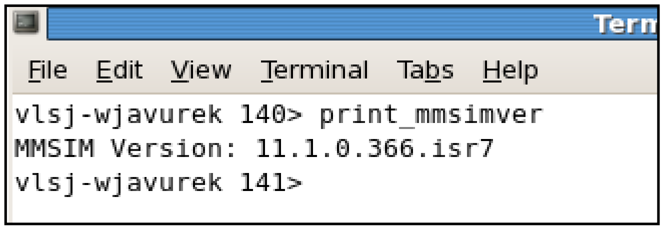
To run AMS, the INCISIV software needs to be installed and includes analog simulator engines (Spectre, APS and Ultrasim). AMS is a single executable simulator not a co-simulation. Therefore, the Spectre, APS and Ultrasim solvers are complied into the AMS simulator and the MMSIM installation is not required.

The MMSIM installation, on the other hand, contains the stand-alone executables for Spectre, APS and Ultrasim and is used for analog simulation from ADE or from the command line.

The takeaways are

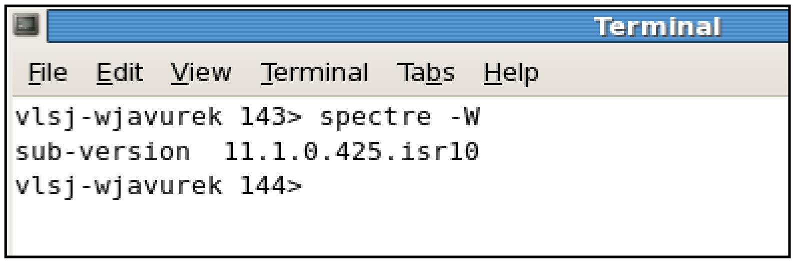
1) MMSIM license works for AMS, Spectre, APS, SpectreRF and Ultrasim.
2) The MMSIM installation is used for analog simulation.
3) The INCISIV installation is used for mixed signal simulation with AMS. AMS does not use the MMSIM installation.

Since the INCISIV package contains the analog solvers (Spectre, Ultrasim and APS), it is possible for the version of analog solver used by AMS to be different than the version used for analog simulation. To see the analog solver version used by AMS, use the ***print_mmsimver*** command from the terminal window.



```
vlsj-wjavurek 140> print_mmsimver
MMSIM Version: 11.1.0.366.isr7
vlsj-wjavurek 141>
```

The Spectre version used for analog simulation can be checked with “spectre –W”. In my case, it is a slightly newer release than the one used for AMS. This is typical since the analog solvers have to be compiled into the AMS simulator after they are developed.



```
vlsj-wjavurek 143> spectre -W
sub-version  11.1.0.425.isr10
vlsj-wjavurek 144>
```

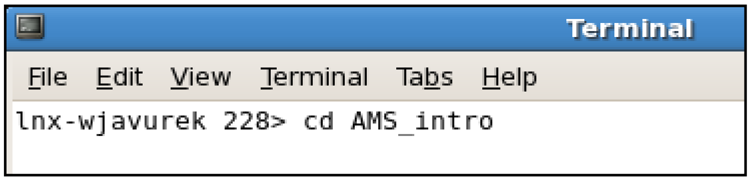
One final word on software installation; to install AMS it is not required to install the full INCISIV package which is quite large. Only products 29630, 70001, and 70020 are needed for AMS simulation. See Support Solution # 11593850

http://support.cadence.com/wps/mypoc/cos?uri=deeplinkmin:ViewSolution;solutionNumber=11593850

Table of Contents

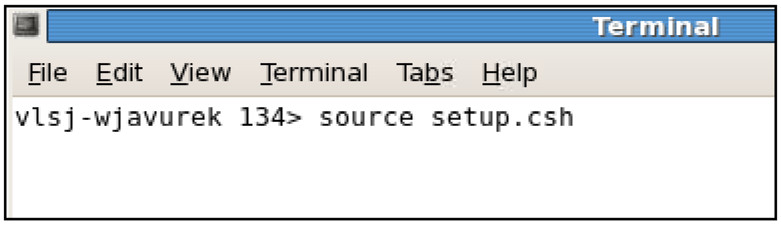
I. Running AMS from ADE

1) **Untar the database and change directory to the** `AMS_intro` **directory**



```
lnx-wjavurek 228> cd AMS_intro
```

2) **Source the** `setup.csh` **script** as shown below



```
vlsj-wjavurek 134> source setup.csh
```

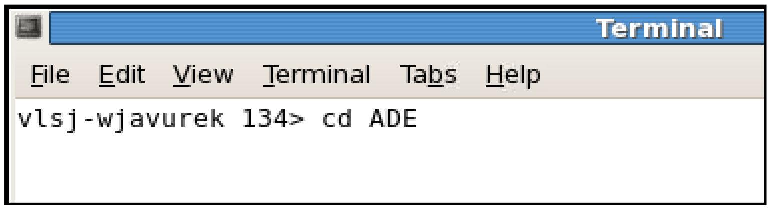
The `setup.csh` script sets 2 Unix environment variables that are referenced by the `cds.lib`

```
    setenv AMSHOME `cds_root irun`
    setenv CDSHOME `cds_root virtuoso'
```

It also sets the variable needed to run OSS UNL:

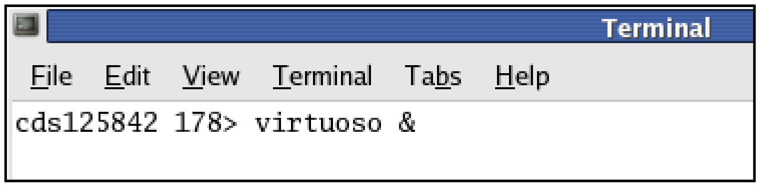
```
    setenv AMS_UNL YES
```

3) **Change directory to "ADE"**



```
vlsj-wjavurek 134> cd ADE
```

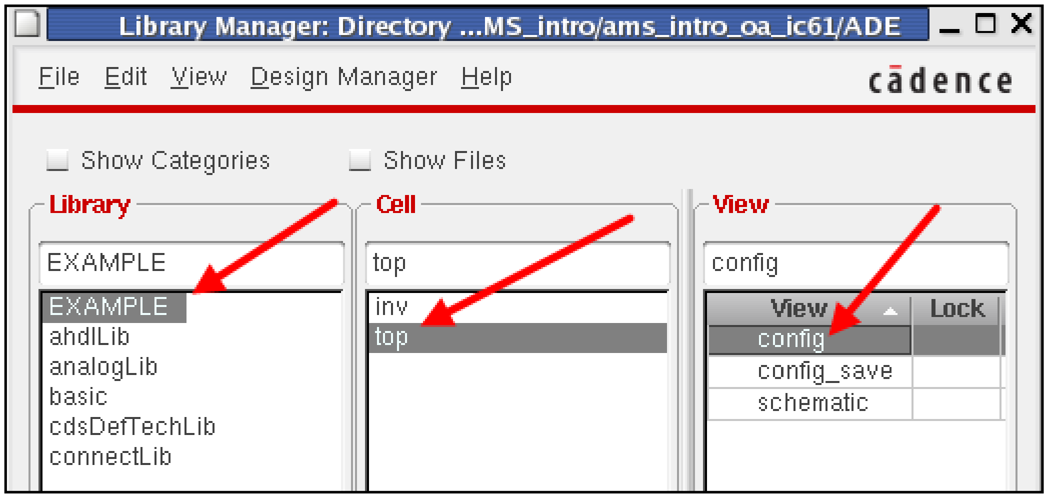
4) **Start virtuoso**



```
cds125842 178> virtuoso &
```

5) **Open the example design**

Library = EXAMPLE
Cell = top
View = config


Library Manager: Directory ...MS_intro/ams_intro_oa_ic61/ADE
File Edit View Design Manager Help
cadence
Show Categories
Show Files
Library
Cell
View
Lock

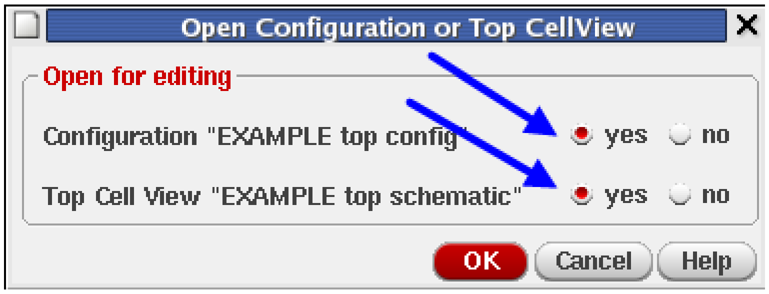
Library: EXAMPLE
EXAMPLE
ahdlLib
analogLib
basic
cdsDefTechLib
connectLib

Cell: top
inv
top

View: config

View	Lock
config	
config_save	
schematic	

6) **Choose both the config view and schematic** when prompted.



Open for editing

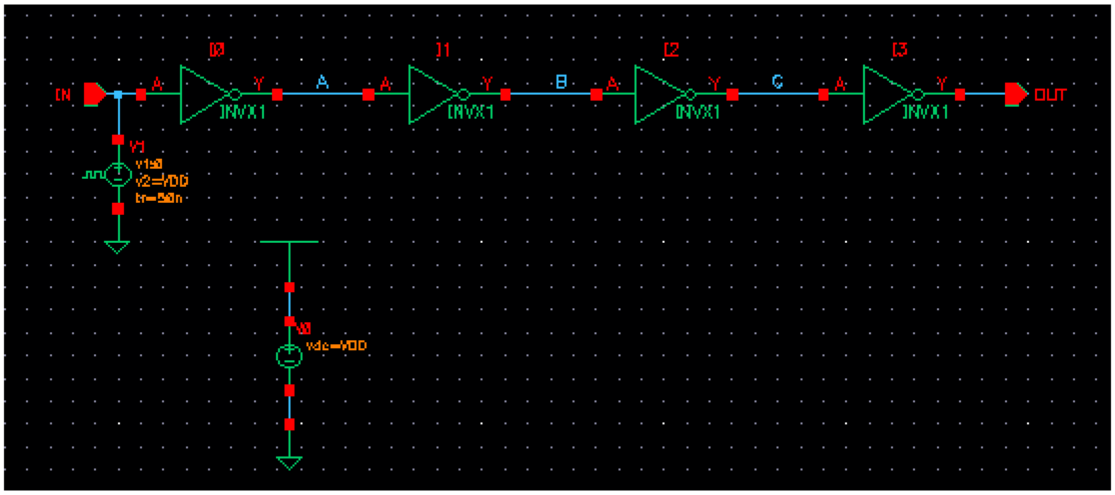
Configuration "EXAMPLE top config" [✓ yes] no

Top Cell View "EXAMPLE top schematic" [✓ yes] no

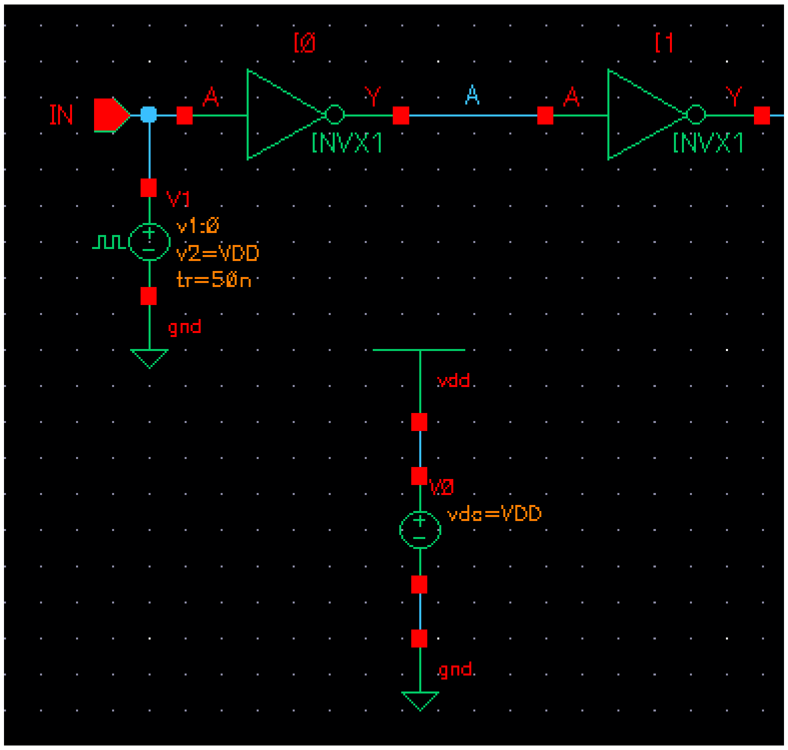
- **Choose "yes" for both and click OK**

7) **The schematic and config view will open**

The test bench schematic is an inverter chain with power and input stimulus.

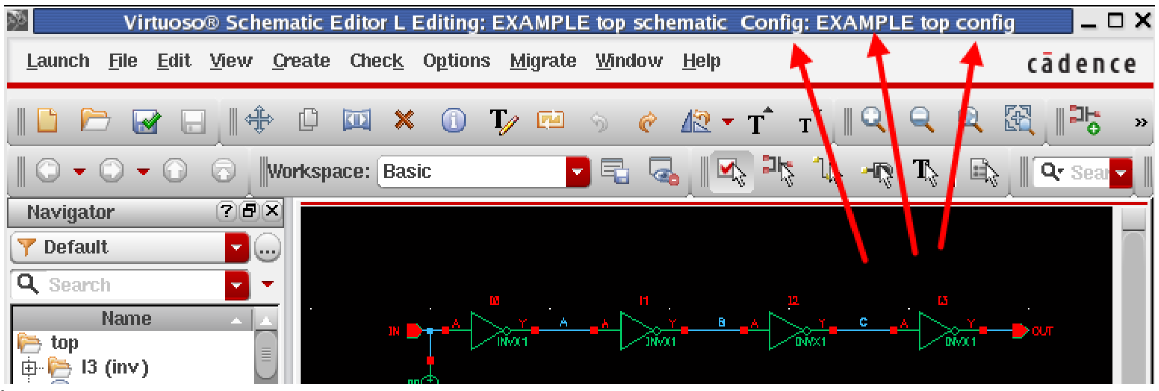


Zooming in on the sources,



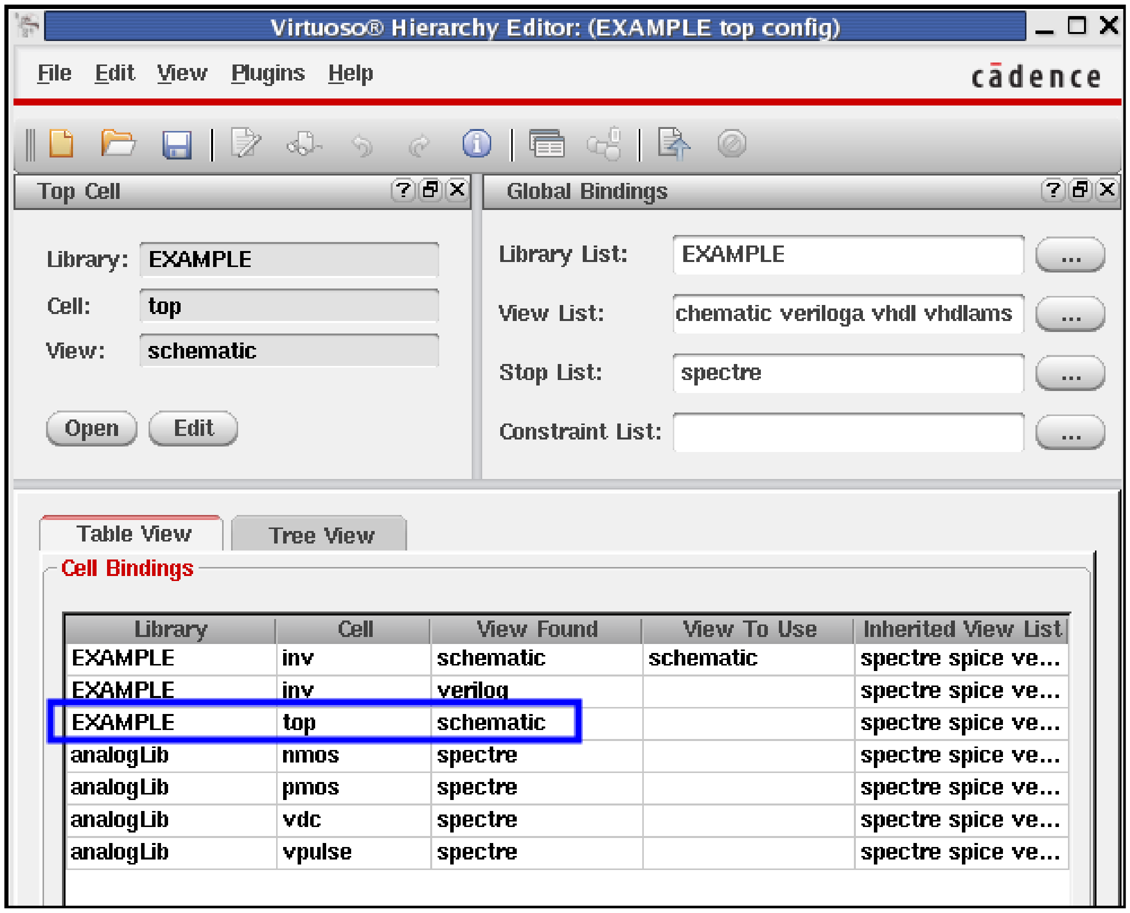
- The power supply, `vdd,` uses a design variable, `VDD`, to set the supply voltage.
- The stimulus is a pulse source varying from `vdd to 0`

Note the banner in the schematic editor (shown below) refers to a "config" view. You may have to expand the size of the schematic window horizontally to see the banner.


Virtuoso® Schematic Editor L Editing: EXAMPLE top schematic Config: EXAMPLE top config
Launch File Edit View Create Check Options Migrate Window Help
cādence
Workspace: Basic
Navigator
Default
Search
Name
top
I3 (inv)
IN
OUT
INVX1

When a schematic is opened with a config view, it is called a "configured" schematic. Running AMS from ADE requires a config view.

The other window that opened along with the schematic window is the ***Hierarchy Editor*** which shows the **config view.**


Virtuoso® Hierarchy Editor: (EXAMPLE top config)
File Edit View Plugins Help
cādence
Top Cell
Global Bindings
Open
Edit
Table View
Tree View
Cell Bindings

Library:	EXAMPLE
Cell:	top
View:	schematic

Library List:	EXAMPLE
View List:	chematic verilog vhdl vhdlams
Stop List:	spectre
Constraint List:	

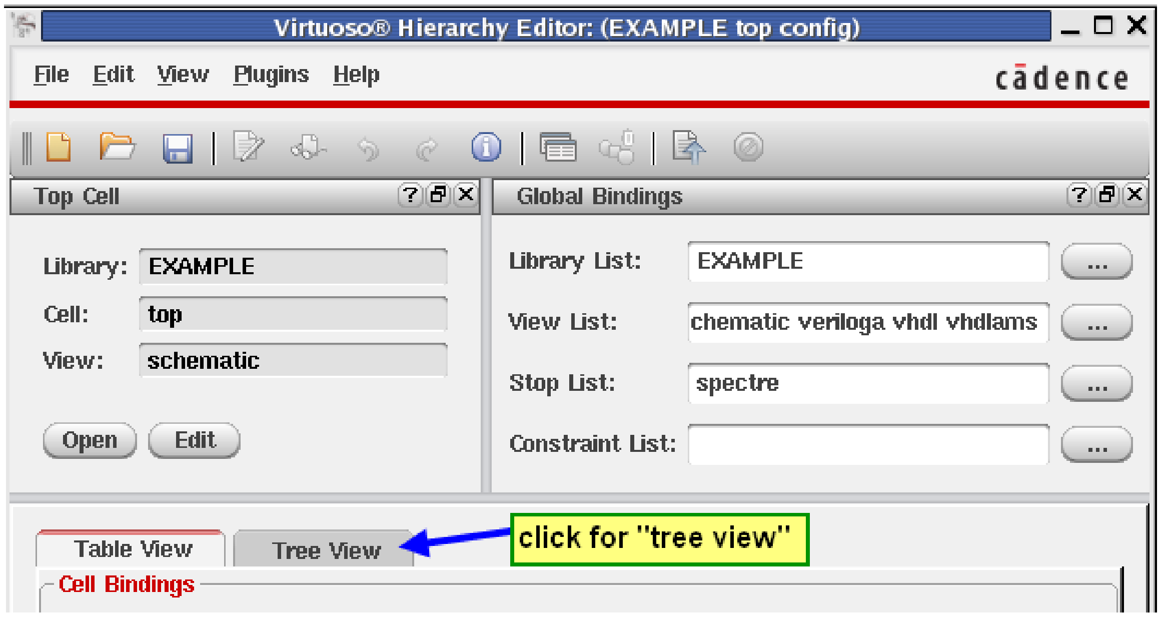
Library	Cell	View Found	View To Use	Inherited View List
EXAMPLE	inv	schematic	schematic	spectre spice ve...
EXAMPLE	inv	verilog		spectre spice ve...
EXAMPLE	top	schematic		spectre spice ve...
analogLib	nmos	spectre		spectre spice ve...
analogLib	pmos	spectre		spectre spice ve...
analogLib	vdc	spectre		spectre spice ve...
analogLib	vpulse	spectre		spectre spice ve...

The config view is used by the netlister to control how the design is represented in the simulation netlist. For example in the config view shown above, the "top" cell is represented by a schematic.

The Hierarchy Editor has 2 display modes

a) Table view (previous page)
b) Tree view

8) **Change the Hierarchy Editor to "tree view"** by **clicking the "tree view" tab** as shown below

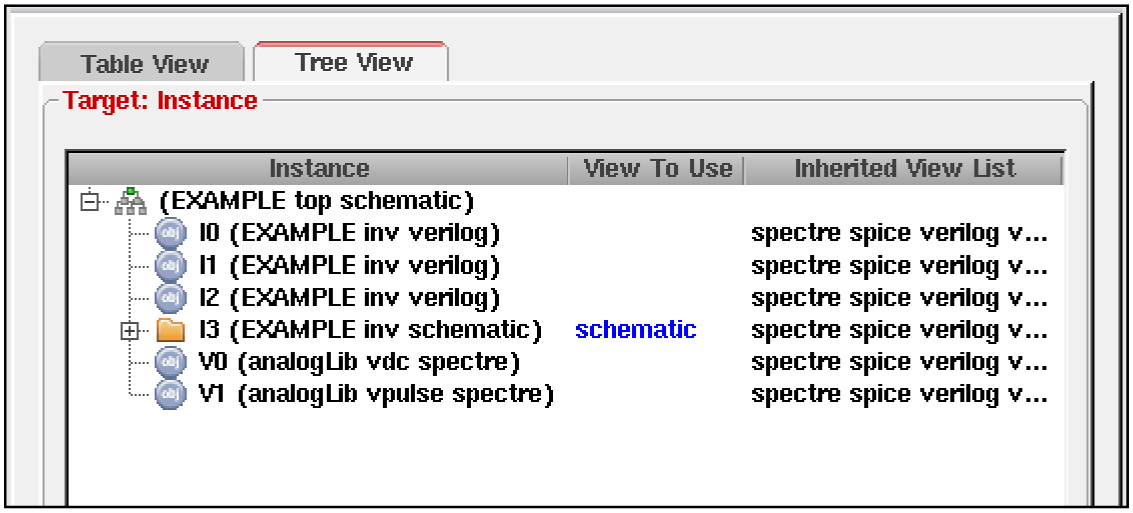


Library: EXAMPLE
Cell: top
View: schematic

Library List: EXAMPLE
View List: chematic veriloga vhdl vhdlams
Stop List: spectre
Constraint List:

Alternatively, tree mode can be set from the menu with **view → tree**

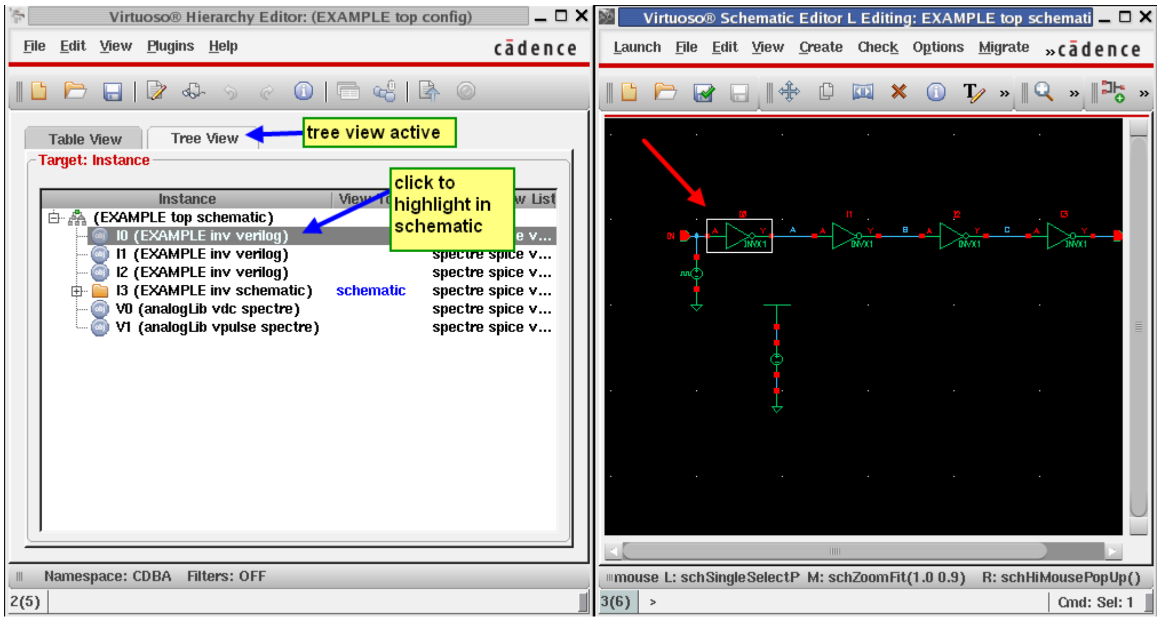
The "tree view" shows a hierarchical tree representation of what is contained in the schematic.



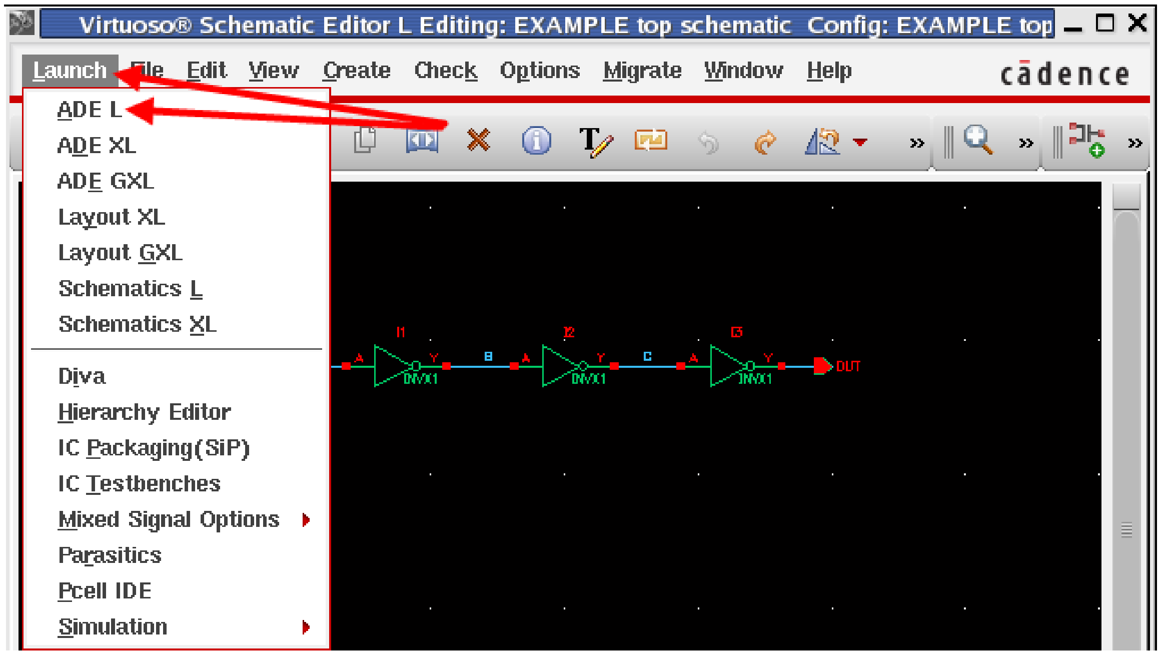
Instance	View To Use	Inherited View List
(EXAMPLE top schematic)		
I0 (EXAMPLE inv verilog)		spectre spice verilog v...
I1 (EXAMPLE inv verilog)		spectre spice verilog v...
I2 (EXAMPLE inv verilog)		spectre spice verilog v...
I3 (EXAMPLE inv schematic)	schematic	spectre spice verilog v...
V0 (analogLib vdc spectre)		spectre spice verilog v...
V1 (analogLib vpulse spectre)		spectre spice verilog v...

Here we can see that inverters I0, I1 and I2 are represented as **verilog** and inverter I3 is represented by a schematic. In other words**, I0, I1 and I2 are digital**, and **I3 is analog**.

The hierarchy editor supports cross selection with the schematic. Clicking on an instance in the hierarchy editor tree view will highlight the corresponding instance in the schematic.

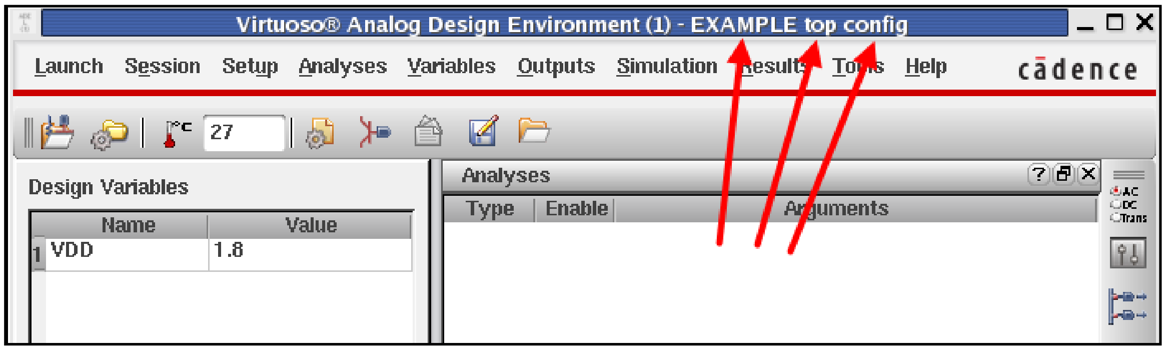

Virtuoso® Hierarchy Editor: (EXAMPLE top config)
File Edit View Plugins Help
Table View
Tree View
tree view active
Target: Instance
Instance
View
click to highlight in schematic
(EXAMPLE top schematic)
I0 (EXAMPLE inv verilog)
I1 (EXAMPLE inv verilog)
I2 (EXAMPLE inv verilog)
I3 (EXAMPLE inv schematic)
schematic
V0 (analogLib vdc spectre)
V1 (analogLib vpulse spectre)
spectre spice v...
Namespace: CDBA Filters: OFF
Virtuoso® Schematic Editor L Editing: EXAMPLE top schemati
Launch File Edit View Create Check Options Migrate
mouse L: schSingleSelectP M: schZoomFit(1.0 0.9) R: schHiMousePopUp()
Cmd: Sel: 1

9) **Start ADE L (Analog Design Environment)** with **Launch → ADE L** from the schematic



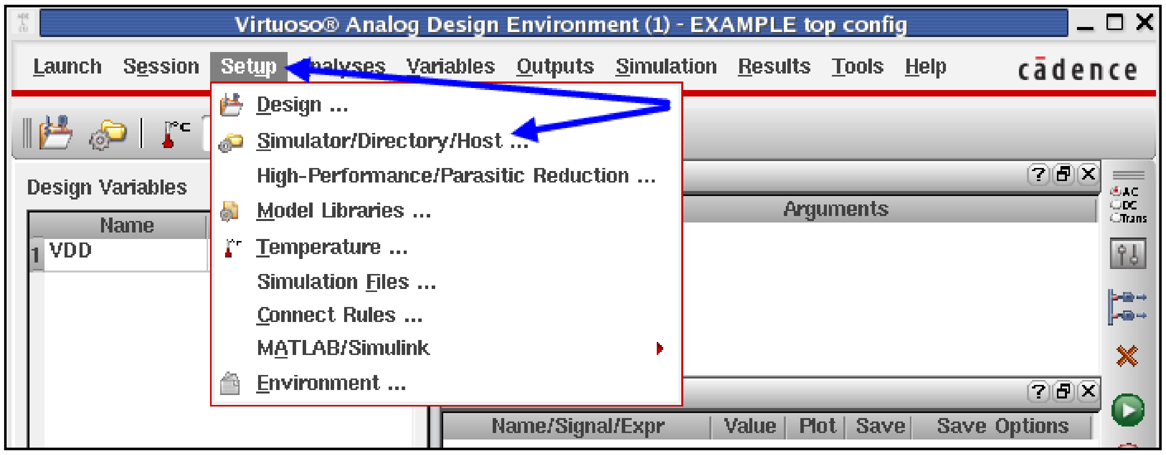
The ADE L window will open.

10) Verify in the title bar of the ADE L window that the design to simulate is the **EXAMPLE > top > config** as shown below.

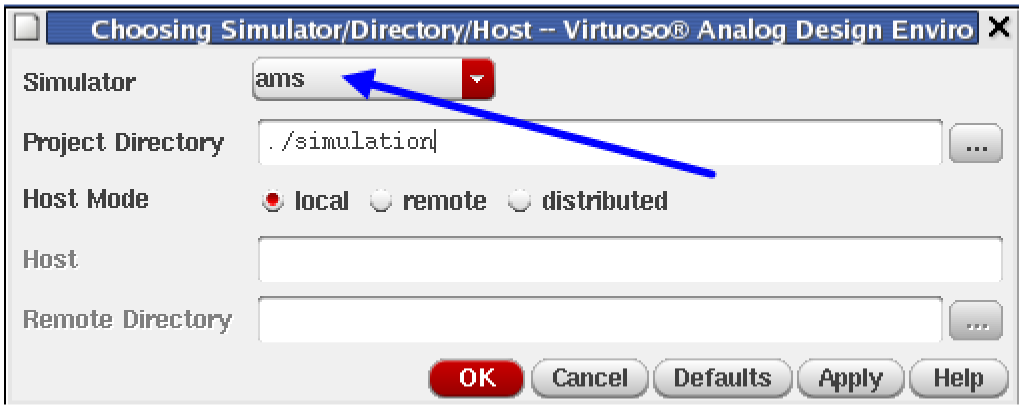

Virtuoso® Analog Design Environment (1) - EXAMPLE top config
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
27
Design Variables
Name
Value
VDD
1.8
Analyses
Type
Enable
Arguments

For AMS simulation, the config view is required. If the design to simulate is not the config view, it will not be possible to set the simulator to be AMS. For example, if the schematic view was opened instead of the config, attempting to set the simulator to be AMS will report an error in the CIW.

11) In ADE L, pick **Setup → Simulator / Directory / Host**


Virtuoso® Analog Design Environment (1) - EXAMPLE top config
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
Design ...
Simulator/Directory/Host ...
High-Performance/Parasitic Reduction ...
Model Libraries ...
Temperature ...
Simulation Files ...
Connect Rules ...
MATLAB/Simulink
Environment ...
Design Variables
Name
VDD
Arguments
Name/Signal/Expr
Value
Plot
Save
Save Options

12) **Verify the simulator is set to AMS** as shown below

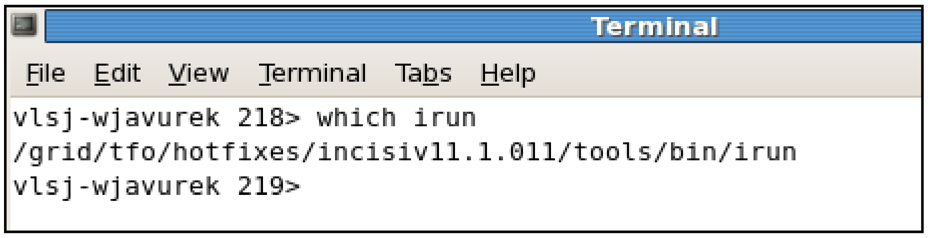

Choosing Simulator/Directory/Host -- Virtuoso® Analog Design Enviro
Host
Remote Directory
OK
Cancel
Defaults
Apply
Help

Simulator: ams
Project Directory: ./simulation
Host Mode: [✓ local] remote distributed

The simulator has been preset to AMS in the `.cdsinit` with the following command
`envSetVal( "asimenv.startup" "simulator" 'string "ams")`

If you cannot set AMS as the simulator,

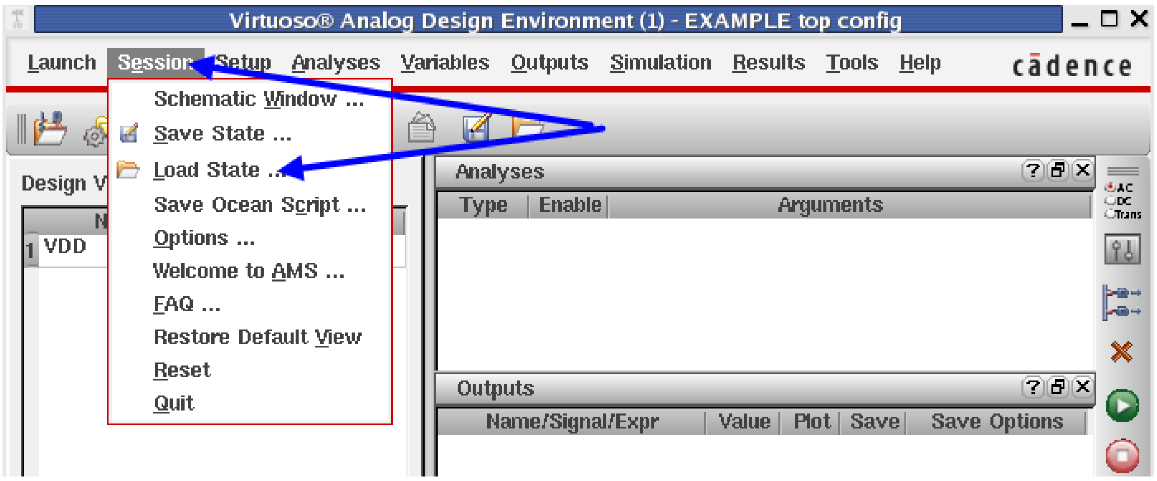
- Double check the config view is the simulation view since that is required by AMS.
- Verify that INCISIVE is in `$path`. Use "`which irun`" from the terminal window where `virtuoso` was started, for example



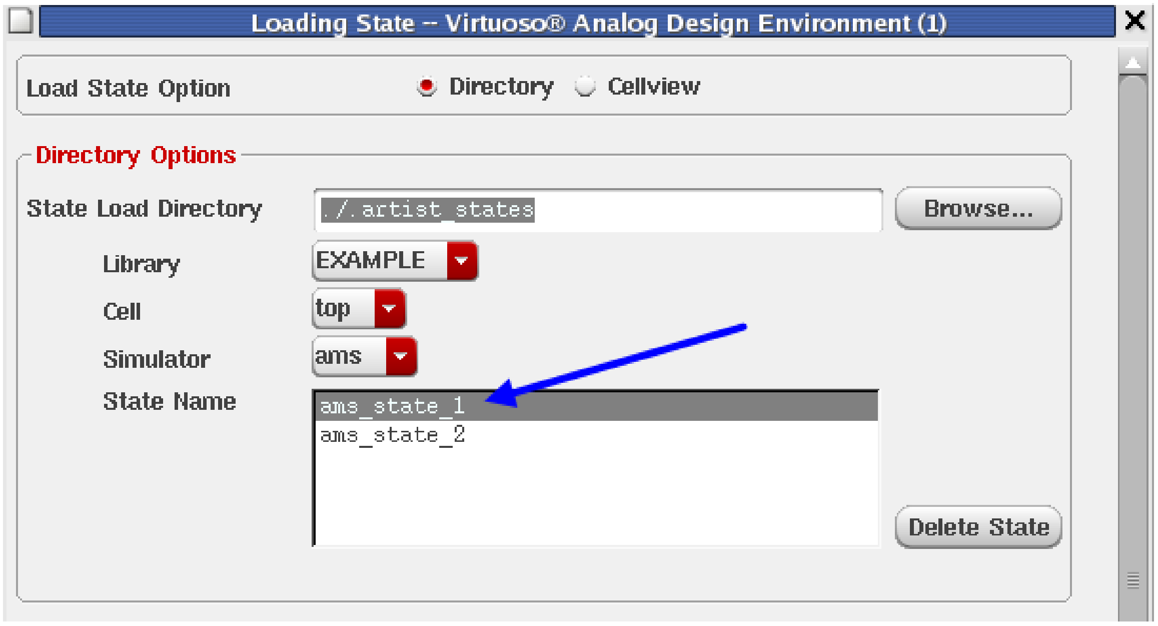
```
vlsj-wjavurek 218> which irun
/grid/tfo/hotfixes/incisiv11.1.011/tools/bin/irun
vlsj-wjavurek 219>
```

13) **Click OK to close the ADE simulation setup dialog**

14) From ADE, pick **Session → Load State**



15) In the load state window, choose **ams_state_1** and **click OK** at the bottom of the form.

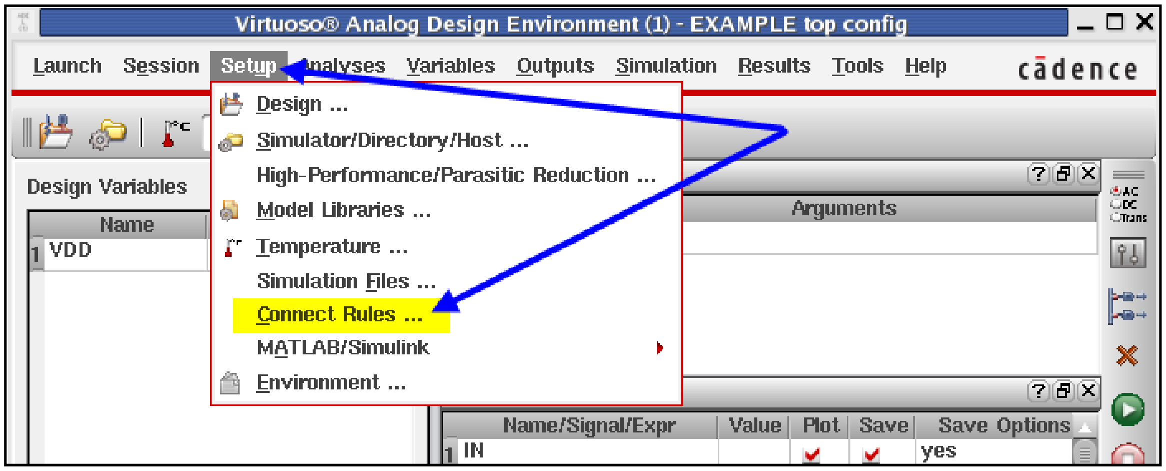


Load State Option [✓ Directory] Cellview

Directory Options

State Load Directory	./.artist_states
Library	EXAMPLE
Cell	top
Simulator	ams
State Name	ams_state_1 ams_state_2

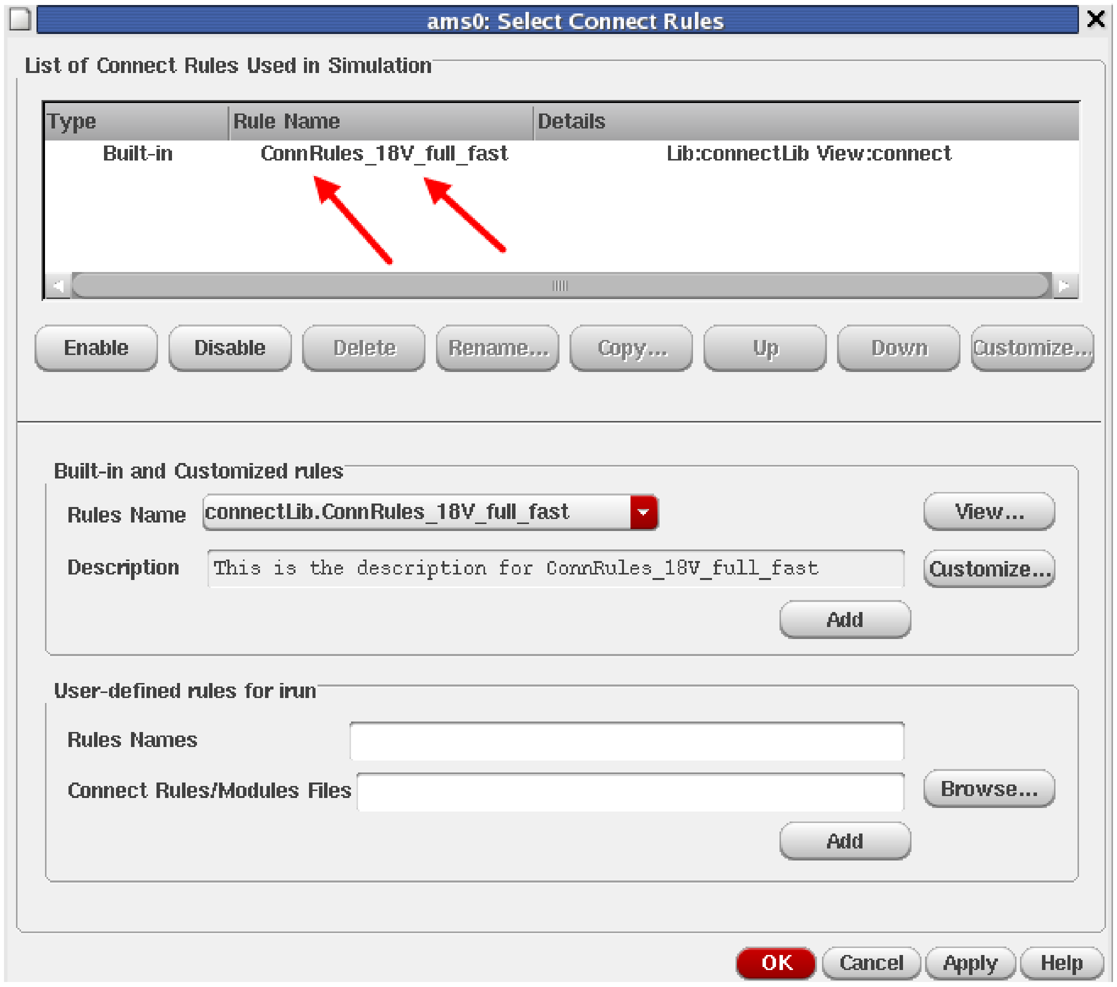
16) Pick **Setup → Connect Rules**


Virtuoso® Analog Design Environment (1) - EXAMPLE top config
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
Design ...
Simulator/Directory/Host ...
High-Performance/Parasitic Reduction ...
Model Libraries ...
Temperature ...
Simulation Files ...
Connect Rules ...
MATLAB/Simulink
Environment ...
Design Variables
Name
1 VDD
Arguments
Name/Signal/Expr Value Plot Save Save Options
1 IN yes

The connect rules are Verilog-AMS modules used for the analog/digital signal conversion between the digital and analog blocks.

The connect rules are set to be ***ConnRules_18V_full_fast*** from the loaded ADE state.

The ***full_fast*** rules are recommended for AMS simulation. The "18V" refers to the voltage for logic levels. In the case of ***ConnRules_18V_full_fast***, logic1 will be represented as 1.8V in the analog domain.


ams0: Select Connect Rules

List of Connect Rules Used in Simulation

Type	Rule Name	Details
Built-in	ConnRules_18V_full_fast	Lib:connectLib View:connect

Built-in and Customized rules

Rules Name: connectLib.ConnRules_18V_full_fast

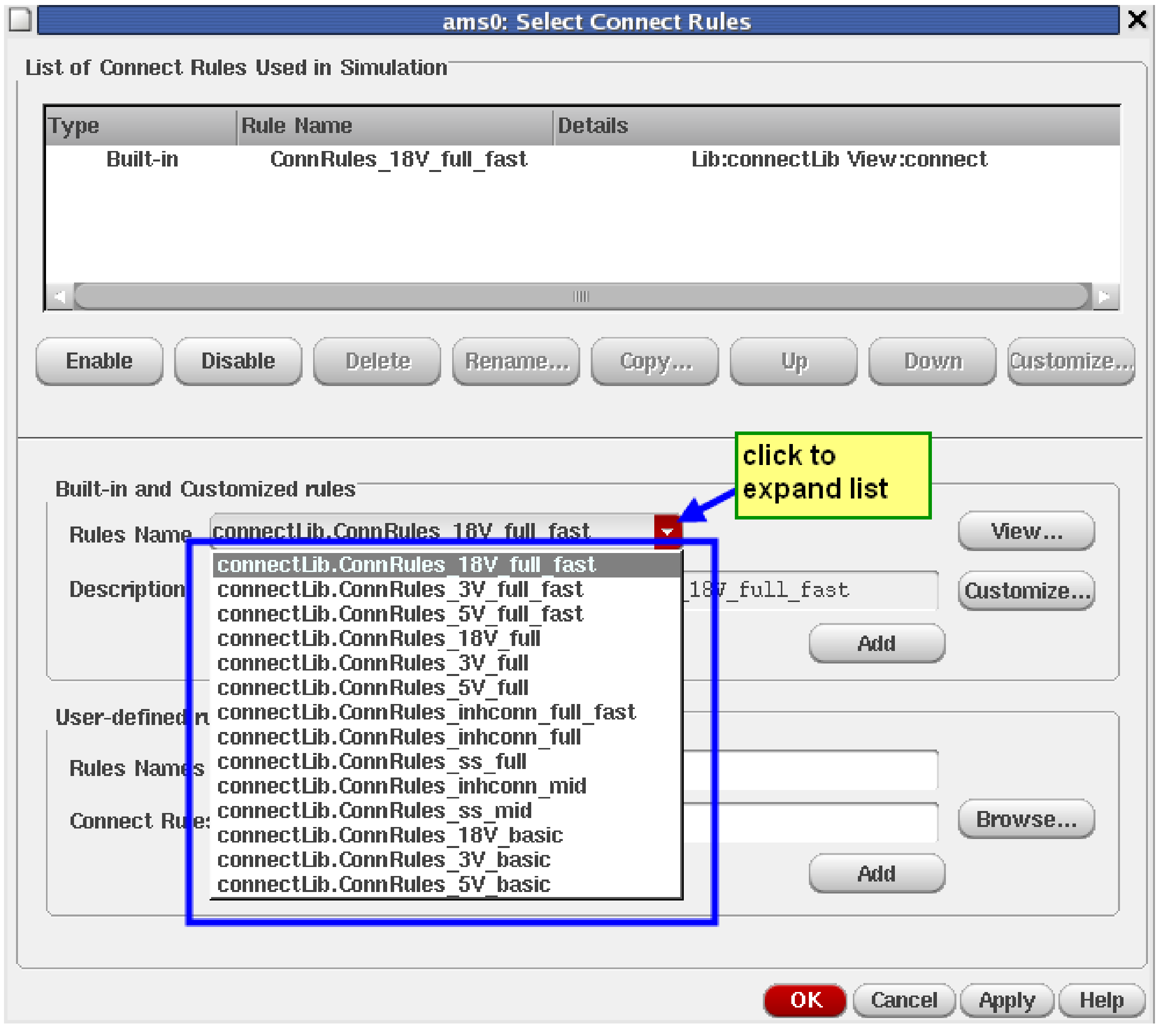
Description: This is the description for ConnRules_18V_full_fast

User-defined rules for irun

Rules Names:

Connect Rules/Modules Files:

AMS Designer has a variety of connect rules built in which can be seen by clicking on the **Rules Name** drop down list as shown below.



Type	Rule Name	Details
Built-in	ConnRules_18V_full_fast	Lib:connectLib View:connect

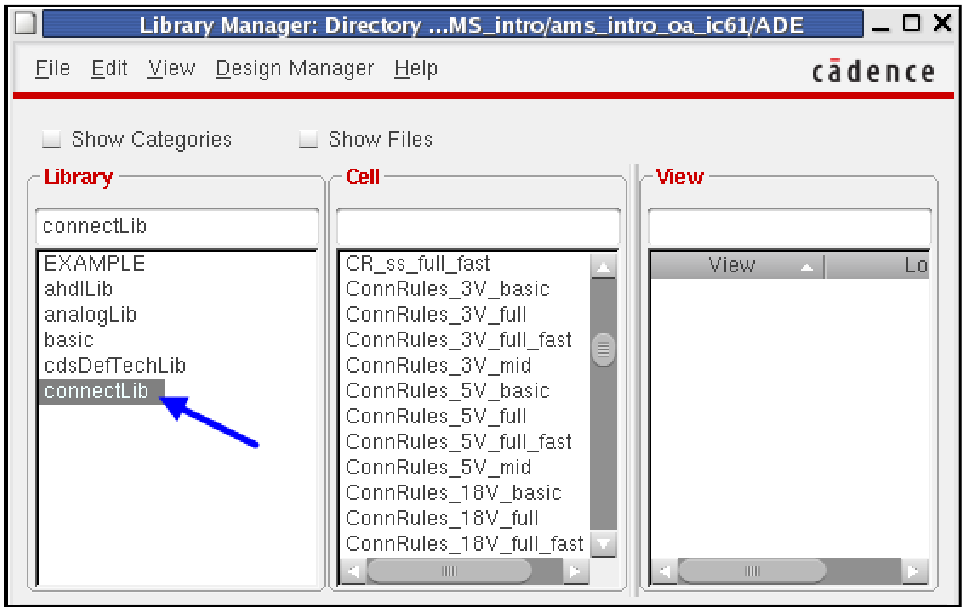
Rules Name: connectLib.ConnRules_18V_full_fast

Description: ..._18V_full_fast

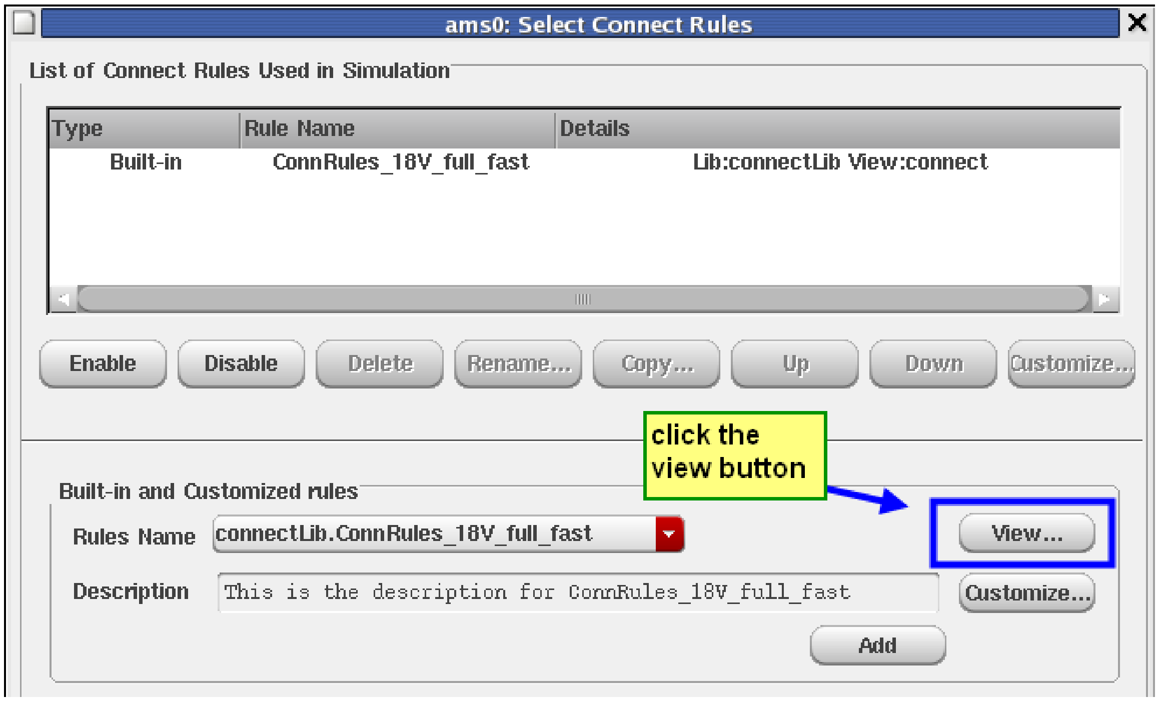
- connectLib.ConnRules_18V_full_fast
- connectLib.ConnRules_3V_full_fast
- connectLib.ConnRules_5V_full_fast
- connectLib.ConnRules_18V_full
- connectLib.ConnRules_3V_full
- connectLib.ConnRules_5V_full
- connectLib.ConnRules_inhconn_full_fast
- connectLib.ConnRules_inhconn_full
- connectLib.ConnRules_ss_full
- connectLib.ConnRules_inhconn_mid
- connectLib.ConnRules_ss_mid
- connectLib.ConnRules_18V_basic
- connectLib.ConnRules_3V_basic
- connectLib.ConnRules_5V_basic

After looking at the rules, **click on the anywhere in the connect rule setup window to collapse the list.** Leave the connect rule setup window open.

The built-in connect rule list is from the ***connectLib*** in the library manager



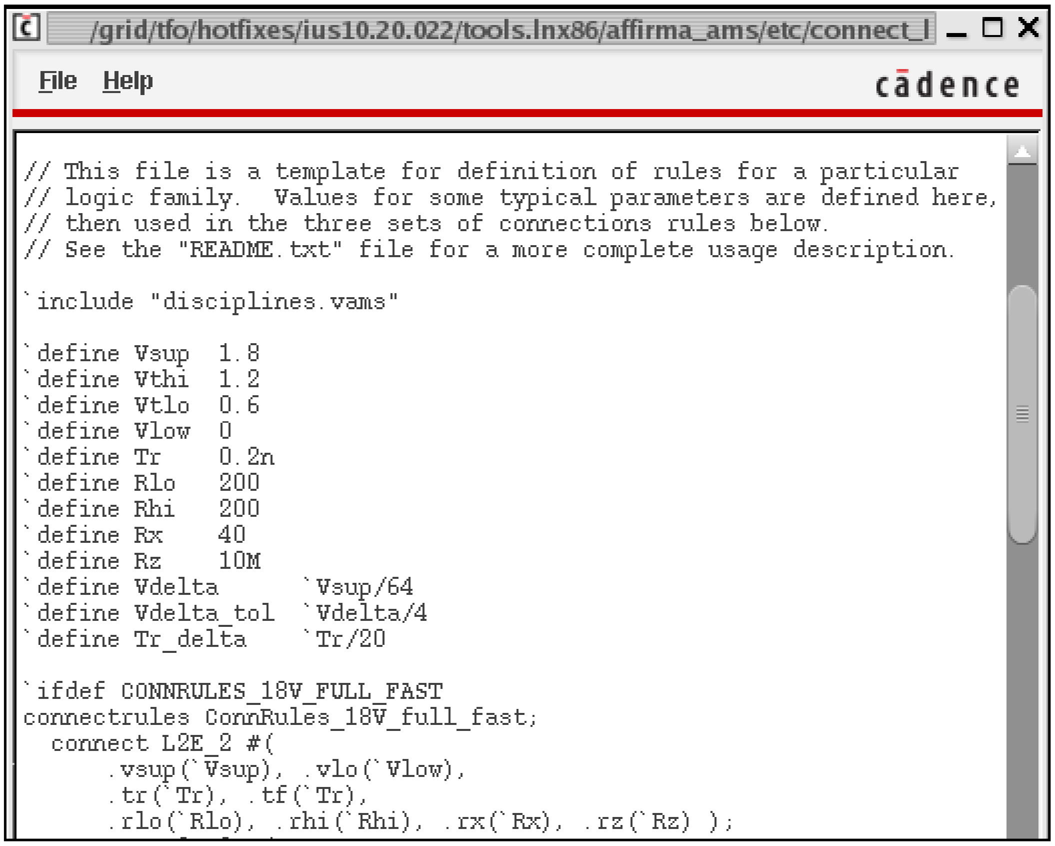
17) **In the connect rule setup dialog**, click the **view button** as shown below to view the source code for the connect module.



Type	Rule Name	Details
Built-in	ConnRules_18V_full_fast	Lib:connectLib View:connect

Rules Name: connectLib.ConnRules_18V_full_fast

Description: This is the description for ConnRules_18V_full_fast

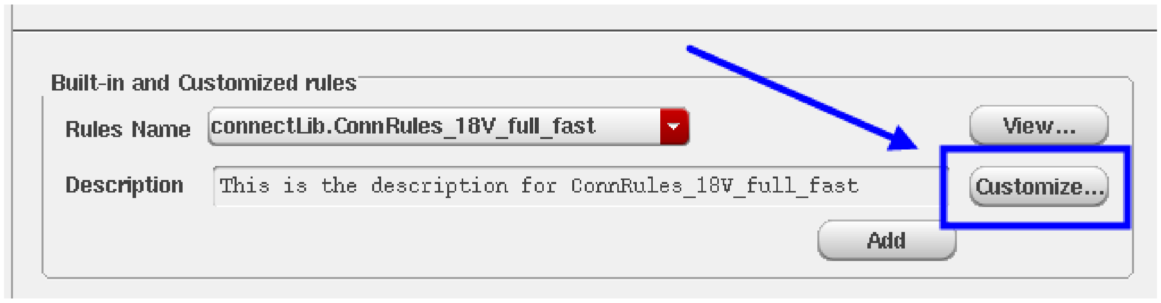

/grid/tfo/hotfixes/ius10.20.022/tools.lnx86/affirma_ams/etc/connect_l
File
Help
cādence

```
// This file is a template for definition of rules for a particular
// logic family.  Values for some typical parameters are defined here,
// then used in the three sets of connections rules below.
// See the "README.txt" file for a more complete usage description.

`include "disciplines.vams"

`define Vsup  1.8
`define Vthi  1.2
`define Vtlo  0.6
`define Vlow  0
`define Tr    0.2n
`define Rlo   200
`define Rhi   200
`define Rx    40
`define Rz    10M
`define Vdelta      `Vsup/64
`define Vdelta_tol  `Vdelta/4
`define Tr_delta    `Tr/20

`ifdef CONNRULES_18V_FULL_FAST
connectrules ConnRules_18V_full_fast;
  connect L2E_2 #(
      .vsup(`Vsup), .vlo(`Vlow),
      .tr(`Tr), .tf(`Tr),
      .rlo(`Rlo), .rhi(`Rhi), .rx(`Rx), .rz(`Rz) );
```

Near the top of the connect rule file, the power supply, `Vsup`, can be seen along with the threshold voltages for `logic1` and `logic0`.

18) **Close the connect rule source code window.**

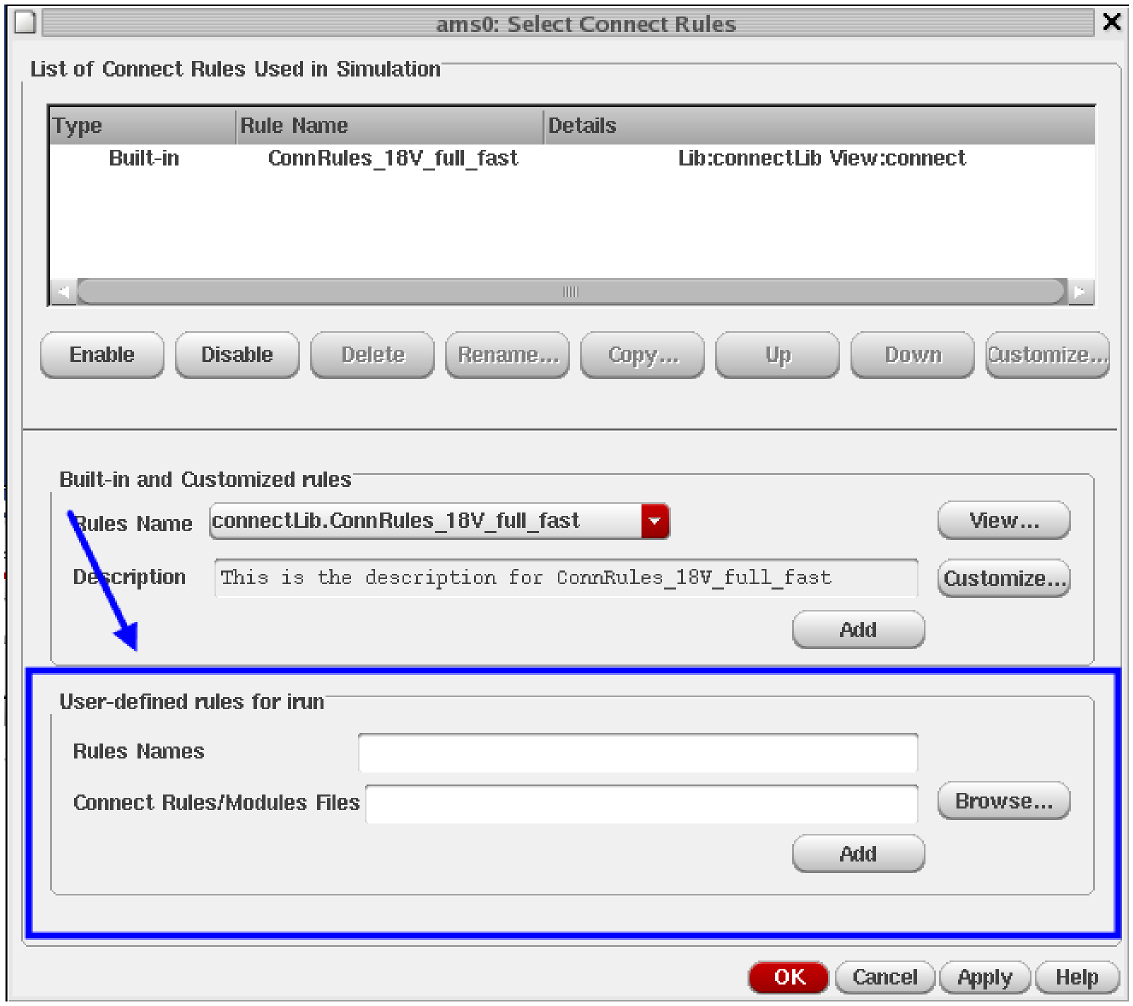
The **Customize** button, provides a mechanism for customizing the connect rules from within ADE.



Rules Name: connectLib.ConnRules_18V_full_fast

Description: This is the description for ConnRules_18V_full_fast

In addition, AMS designer also supports **user defined connect rules** which can be setup in the lower part of the connect rule form.


ams0: Select Connect Rules
List of Connect Rules Used in Simulation
Enable
Disable
Delete
Rename...
Copy...
Up
Down
Customize...
Built-in and Customized rules
View...
Customize...
Add
User-defined rules for irun
Browse...
Add
OK
Cancel
Apply
Help

Type	Rule Name	Details
Built-in	ConnRules_18V_full_fast	Lib:connectLib View:connect

Rules Name: connectLib.ConnRules_18V_full_fast

Description: This is the description for ConnRules_18V_full_fast

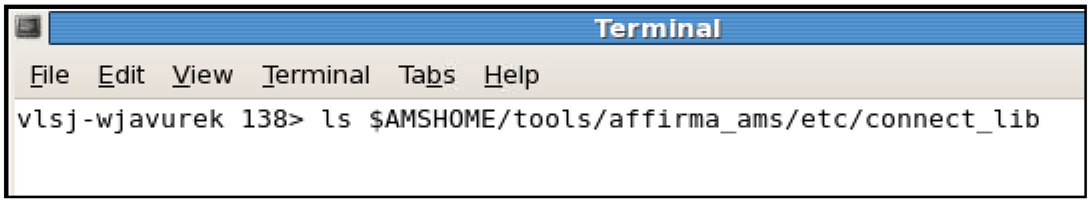
Rules Names:

Connect Rules/Modules Files:

Sometimes it easier to copy the connect rules from the Incisive installation to the project directory and make changes to the source code opposed to using the GUI to customize the rules.

The connect rules are located in

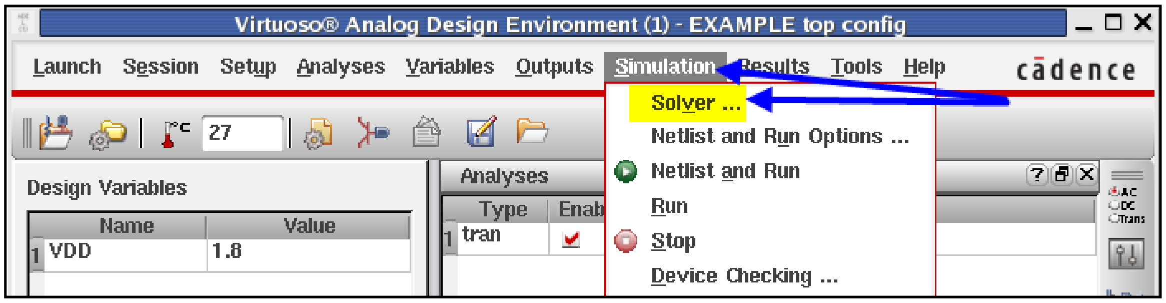
`<IUS_install_dir>/tools/affirma_ams/etc/connect_lib`



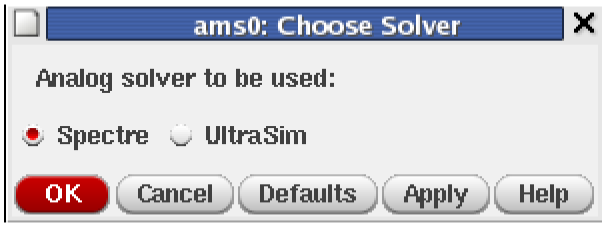
```
vlsj-wjavurek 138> ls $AMSHOME/tools/affirma_ams/etc/connect_lib
```

19) **Click Cancel** on the connect rule setup dialog

20) From ADE, pick **Simulation → Solver**


Virtuoso® Analog Design Environment (1) - EXAMPLE top config
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
Solver ...
Netlist and Run Options ...
Netlist and Run
Run
Stop
Device Checking ...
27
Design Variables
Name Value
1 VDD 1.8
Analyses
Type Enab
1 tran

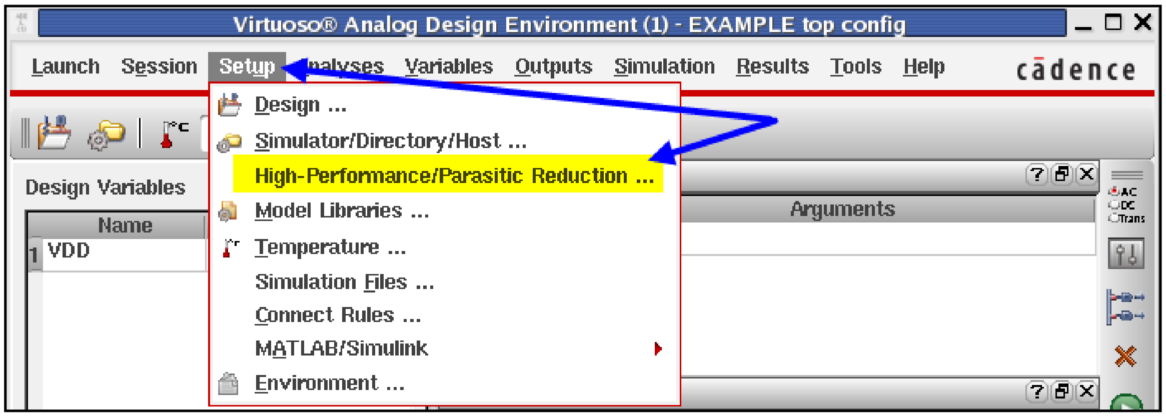
This will open a window which sets up the solver (or simulation engine) to be used for the analog.



For this test case, the design is very small and Spectre will be used for the analog engine. For designs with large analog content, Ultrasim (fast spice) can be used. High performance options (described below) can also be applied to Spectre for larger designs to increase performance.

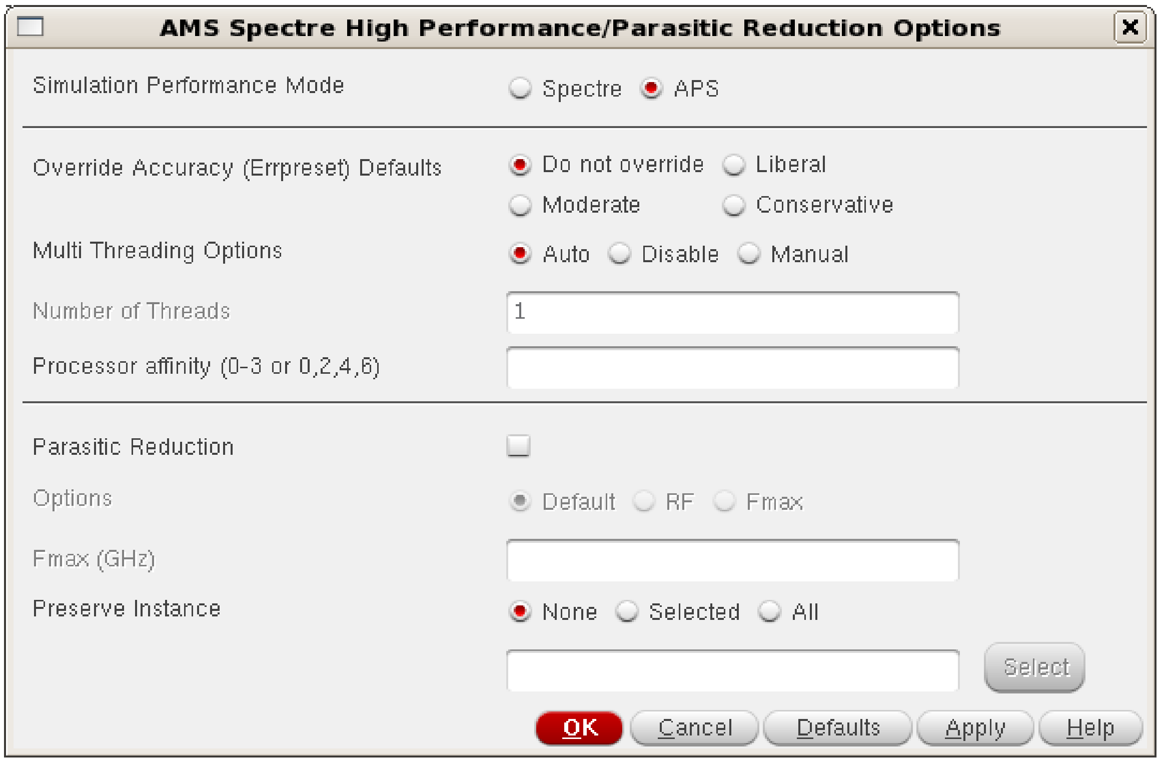
21) **Click Cancel** on the solver setup window

22) Pick **Setup → High-Performance/Parasitic Reduction** in ADE as shown below


Virtuoso® Analog Design Environment (1) - EXAMPLE top config
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
Design ...
Simulator/Directory/Host ...
High-Performance/Parasitic Reduction ...
Model Libraries ...
Temperature ...
Simulation Files ...
Connect Rules ...
MATLAB/Simulink
Environment ...
Design Variables
Name
1 VDD
Arguments

Here is where the High Performance Spectre options can be set

- APS mode
- Spectre multi threading options
- Parasitic reduction


AMS Spectre High Performance/Parasitic Reduction Options

Simulation Performance Mode: Spectre [✓ APS]

Override Accuracy (Errpreset) Defaults: [✓ Do not override] Liberal Moderate Conservative

Multi Threading Options: [✓ Auto] Disable Manual

Number of Threads: 1

Processor affinity (0-3 or 0,2,4,6):

Parasitic Reduction: ☐

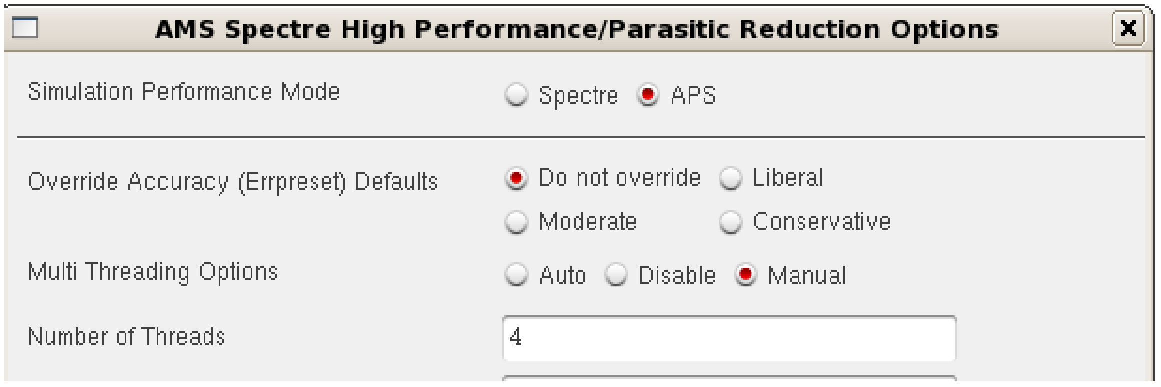
Options: [✓ Default] RF Fmax

Fmax (GHz):

Preserve Instance: [✓ None] Selected All

APS mode enables multithreading support to Spectre for faster simulations on designs with large analog content. The default is for multithreading is “auto mode” which will automatically detect the number CPUs and set the number threads equal to the number of CPUs. For example, if the machine has 8 CPUs, “auto mode” would use 8 threads.

The number of threads can be manually specified by choosing “manual” and specifying the number of threads. For example


AMS Spectre High Performance/Parasitic Reduction Options

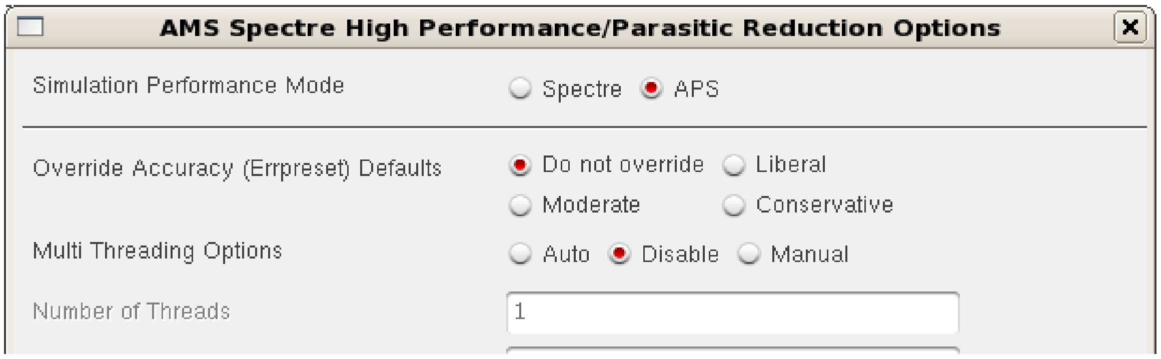
Simulation Performance Mode: Spectre [✓ APS]

Override Accuracy (Errpreset) Defaults: [✓ Do not override] Liberal Moderate Conservative

Multi Threading Options: Auto Disable [✓ Manual]

Number of Threads: 4

One other point on APS before we move on. The multithreaded matrix solver in APS is more efficient than the standard Spectre matrix solver, and APS with 1 CPU will be faster than Spectre. For 1 CPU APS, choose APS mode and disable multithreading as shown below


AMS Spectre High Performance/Parasitic Reduction Options

Simulation Performance Mode: Spectre [✓ APS]

Override Accuracy (Errpreset) Defaults: [✓ Do not override] Liberal Moderate Conservative

Multi Threading Options: Auto [✓ Disable] Manual

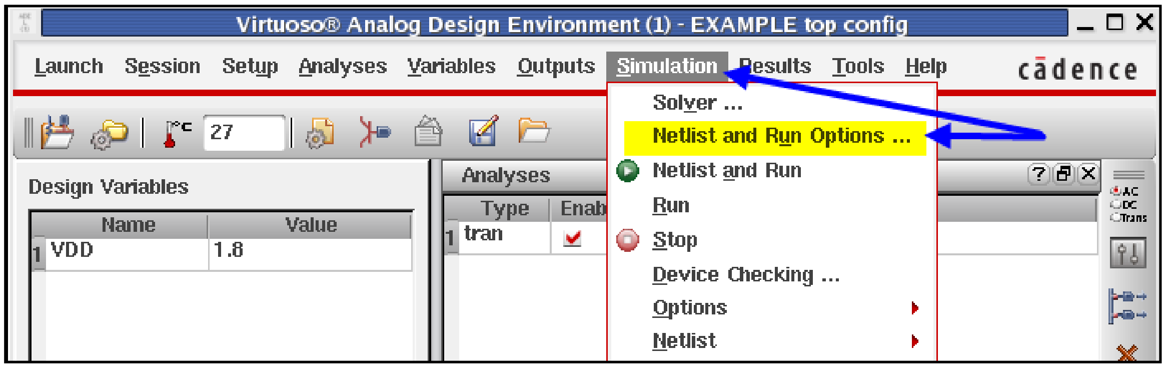
Number of Threads: 1

The chosen analog solver and Spectre high performance options are saved in the ADE state file and will be restored when loading the state.

The design for this example is not large enough to benefit from APS, and we will simulate with Spectre.

23) Click **Cancel** at the bottom of the window to **close the Spectre High Performance setup window**

24) Pick **Simulation → Netlister and Run Options** from ADE

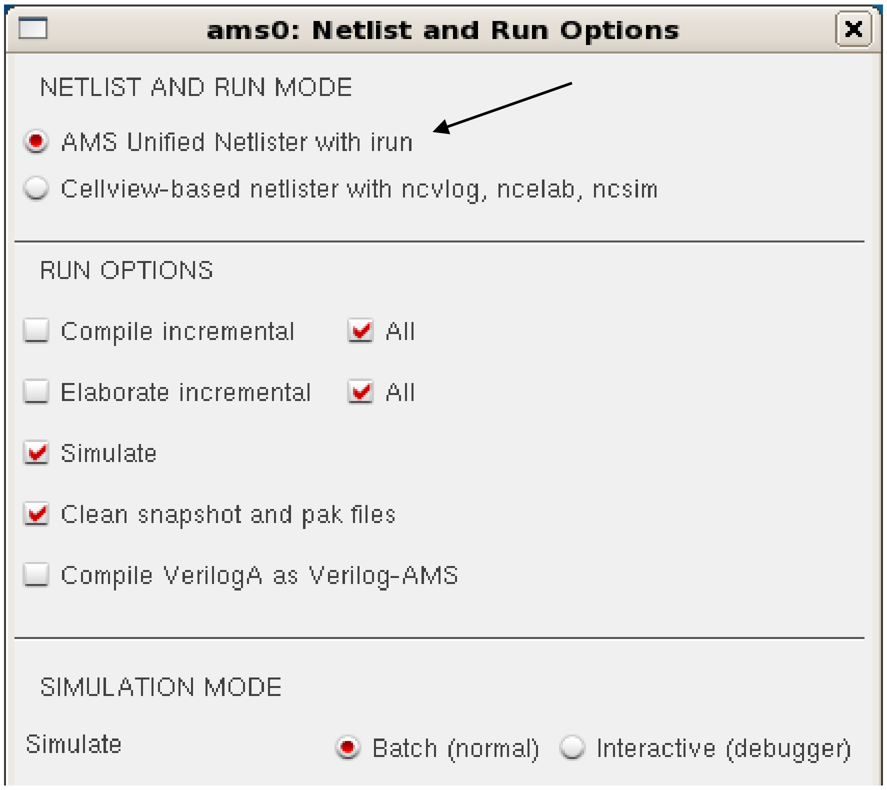


	Name	Value
1	VDD	1.8

	Type	Enab
1	tran	✓

This will open the options dialog that sets up netlisting.

25) **Verify that the netlisting mode is set to “AMS Unified Netlister with irun” as shown below.**



NETLIST AND RUN MODE

[● AMS Unified Netlister with irun] ○ Cellview-based netlister with ncvlog, ncelab, ncsim

RUN OPTIONS

- ☐ Compile incremental ☑ All
- ☐ Elaborate incremental ☑ All
- ☑ Simulate
- ☑ Clean snapshot and pak files
- ☐ Compile VerilogA as Verilog-AMS

SIMULATION MODE

Simulate [● Batch (normal)] ○ Interactive (debugger)

The netlisting mode was set to UNL from the loaded state.

Leave the simulation options window open for now.

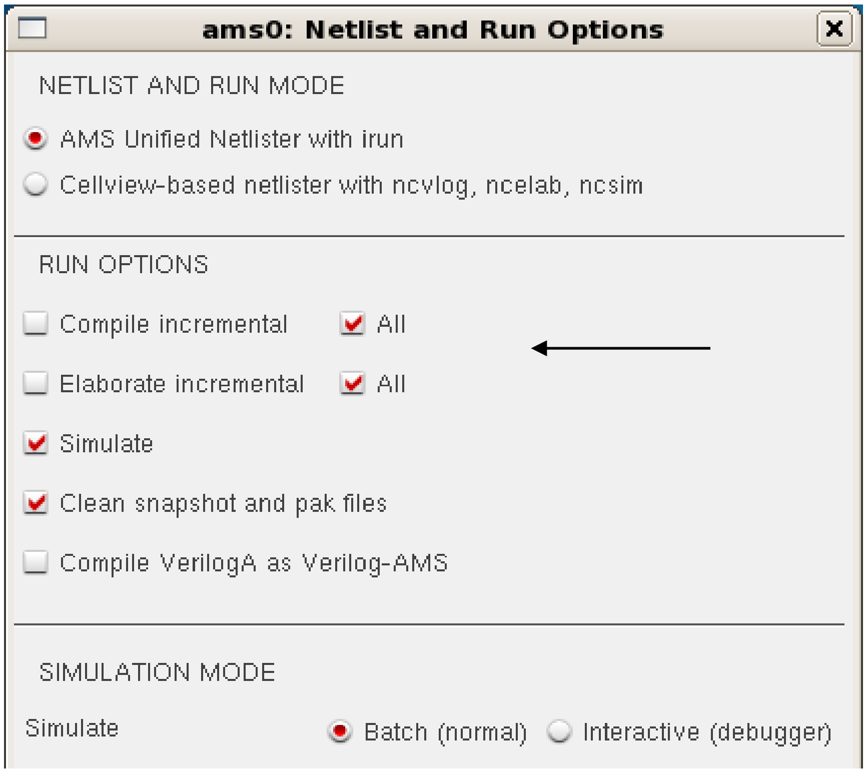
Important

The **default is the cellview-based netlister (CBN) and needs to be manually set to OSS**. After setting OSS, save the state and the netlisting mode will be read in from the saved state when the state is loaded. To set UNL as the default, put this in your .cdsenv file:

```
ams.envOpts  netlisterMode   string  "OSS-based"
```

AMS Run Options

The netlister and run option forms also has options to control how the simulation will be run. AMS follows the NC-Sim methodology with compile, elaborate and simulate.



ams0: Netlist and Run Options

NETLIST AND RUN MODE

● AMS Unified Netlister with irun
○ Cellview-based netlister with ncvlog, ncelab, ncsim

RUN OPTIONS

☐ Compile incremental ✓ All
☐ Elaborate incremental ✓ All
✓ Simulate
✓ Clean snapshot and pak files
☐ Compile VerilogA as Verilog-AMS

SIMULATION MODE

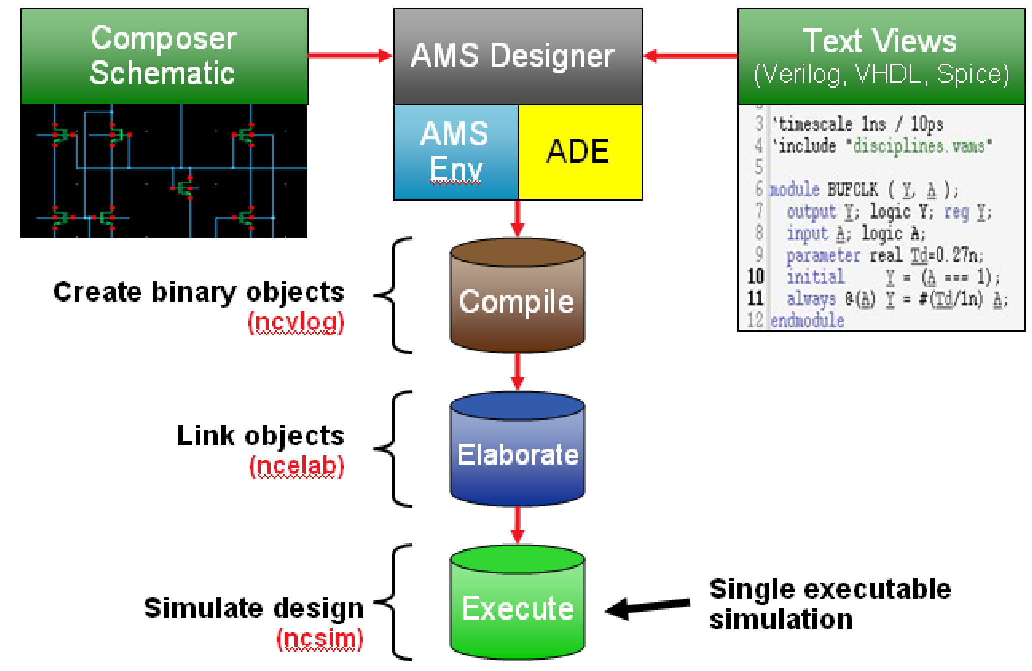
Simulate [✓ Batch (normal)] Interactive (debugger)

- The ***compile*** step compiles the sources into binary `.pak` files
- The ***elaboration*** step links the compiled code from the pak files together <u>and</u> inserts the connect modules to create the ***simulation snapshot***
- The ***simulation*** step runs the simulation snapshot

One way to think of how AMS runs is to think about programming. If you have a programming project with several source code modules, for example written in 'C'.

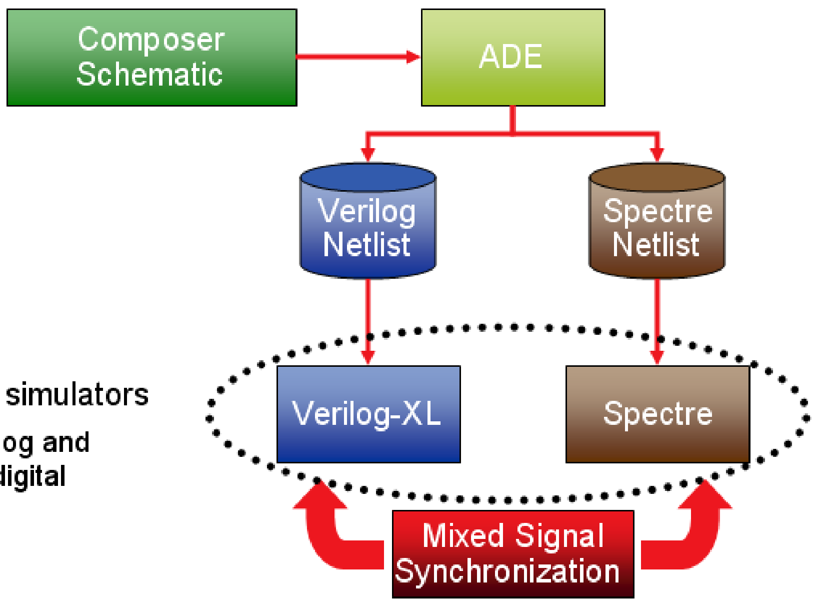
- Each 'C' source code module would be complied to create binary object (.obj) files. This is equivalent to the AMS "compile" step. In the case of AMS, *.pak files are created.
- Once the 'C' source files are complied, a linker is used to link the objects together to create the executable (.exe) file. For AMS designer, this is the "simulation snapshot"
- The last step in creating a 'C' program is to run the executable. The equivalent in AMS is "run simulation". When the AMS simulation is run, the simulator actually runs the complied design. This is why it is referred to as "native complied" and is a key advantage for AMS over co-simulation approaches. AMS is a single executable <u>not</u> a co-simulation that has separate analog and digital simulators running at the same time.

AMS Designer Architecture



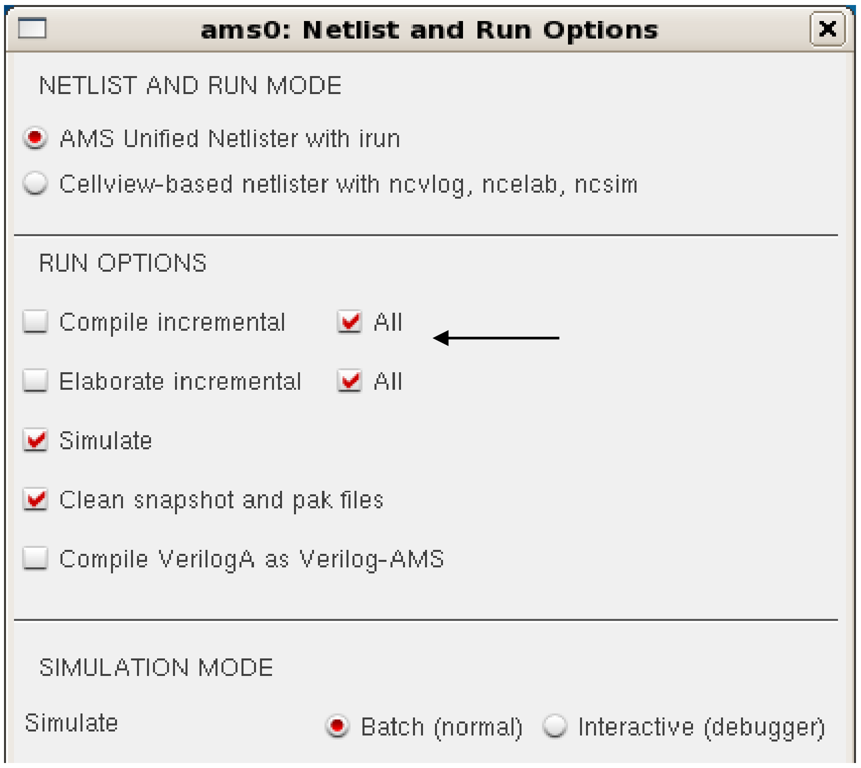
```
'timescale 1ns / 10ps
'include "disciplines.vams"

module BUFCLK ( Y, A );
  output Y; logic Y; reg Y;
  input A; logic A;
  parameter real Td=0.27n;
  initial      Y = (A === 1);
  always @(A) Y = #(Td/1n) A;
endmodule
```

Mixed Signal Simulation with Co-Simulation



- 2 separate simulators
- One for analog and another for digital

Incremental compile and elaborate in the Netlister and Run Options form shown below mean what they say – only the changes in the simulation will be complied / elaborated. The "all" option forces the entire design to compile / elaborate even if the design (config view or schematic) has not changed.


ams0: Netlist and Run Options

NETLIST AND RUN MODE

- ● AMS Unified Netlister with irun
- ○ Cellview-based netlister with ncvlog, ncelab, ncsim

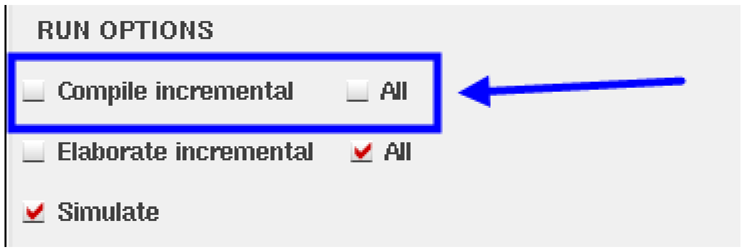
RUN OPTIONS

- ☐ Compile incremental ☑ All
- ☐ Elaborate incremental ☑ All
- ☑ Simulate
- ☑ Clean snapshot and pak files
- ☐ Compile VerilogA as Verilog-AMS

SIMULATION MODE

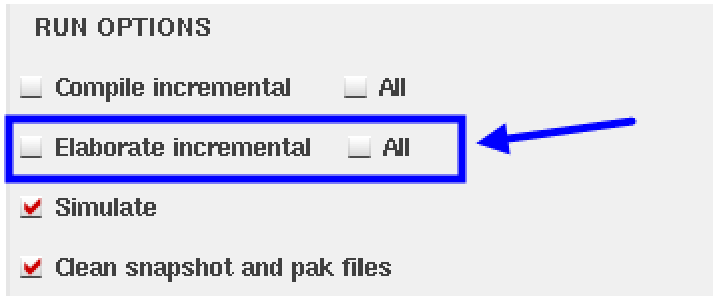
Simulate ● Batch (normal) ○ Interactive (debugger)

- Turning off both options for compile will tell AMS not to run the compile step at all.


RUN OPTIONS

- ☐ Compile incremental ☐ All
- ☐ Elaborate incremental ☑ All
- ☑ Simulate

- Likewise, turning off both incremental and all for elaboration will tell AMS not to run the elaboration step.

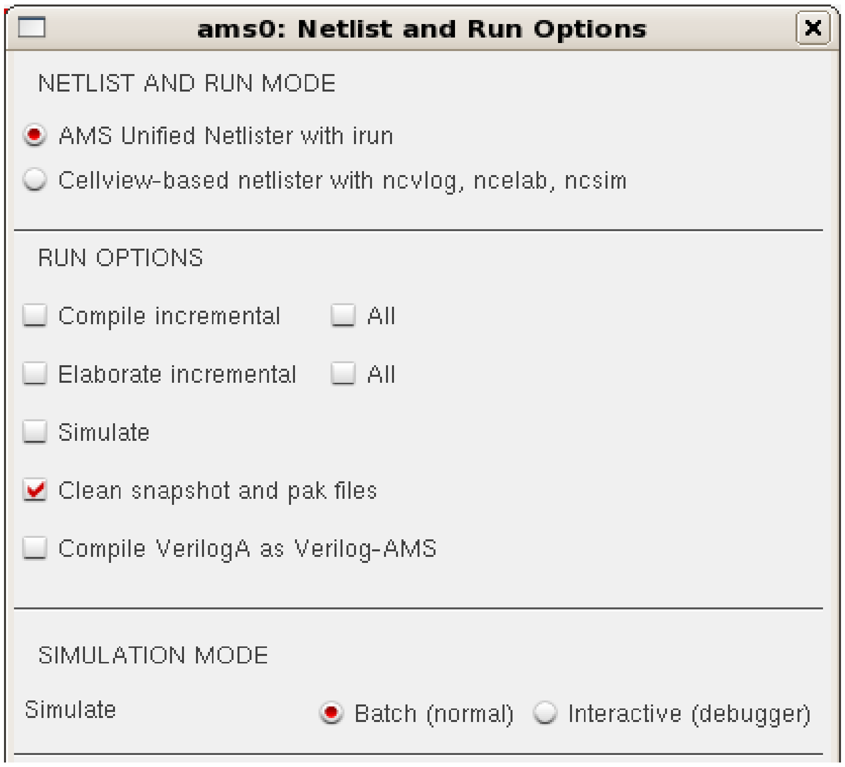


RUN OPTIONS

- [ ] Compile incremental [ ] All
- [ ] Elaborate incremental [ ] All
- [✓] Simulate
- [✓] Clean snapshot and pak files

- The "simulation" option tells AMS whether or not to run the simulation.

Q: What happens when all three steps are disabled as shown below and the simulation is run?



ams0: Netlist and Run Options

NETLIST AND RUN MODE

- [✓] AMS Unified Netlister with irun
- [ ] Cellview-based netlister with ncvlog, ncelab, ncsim

RUN OPTIONS

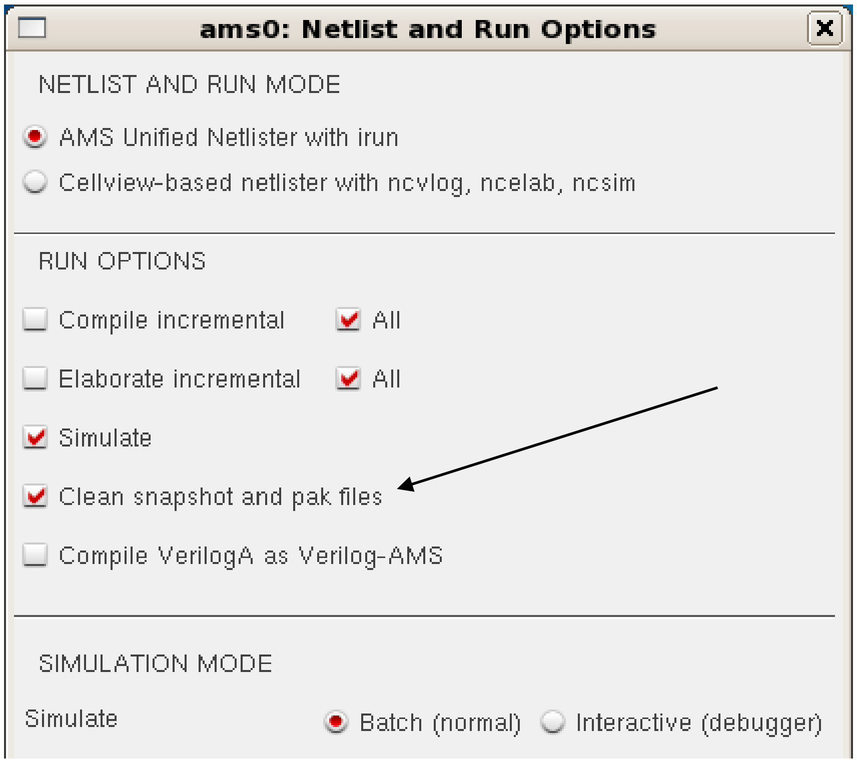
- [ ] Compile incremental [ ] All
- [ ] Elaborate incremental [ ] All
- [ ] Simulate
- [✓] Clean snapshot and pak files
- [ ] Compile VerilogA as Verilog-AMS

SIMULATION MODE

Simulate [✓ Batch (normal)] Interactive (debugger)

A: This mode will only generate the netlist.

The ***Clean snapshot and pak files*** option tells AMS to remove all the data from the previous run



ams0: Netlist and Run Options

NETLIST AND RUN MODE

● AMS Unified Netlister with irun
○ Cellview-based netlister with ncvlog, ncelab, ncsim

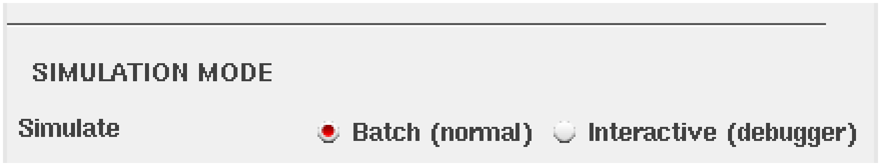
RUN OPTIONS

☐ Compile incremental ✓ All
☐ Elaborate incremental ✓ All
✓ Simulate
✓ Clean snapshot and pak files
☐ Compile VerilogA as Verilog-AMS

SIMULATION MODE

Simulate ● Batch (normal) ○ Interactive (debugger)

The next option in the run mode form is the ***simulation mode***



SIMULATION MODE

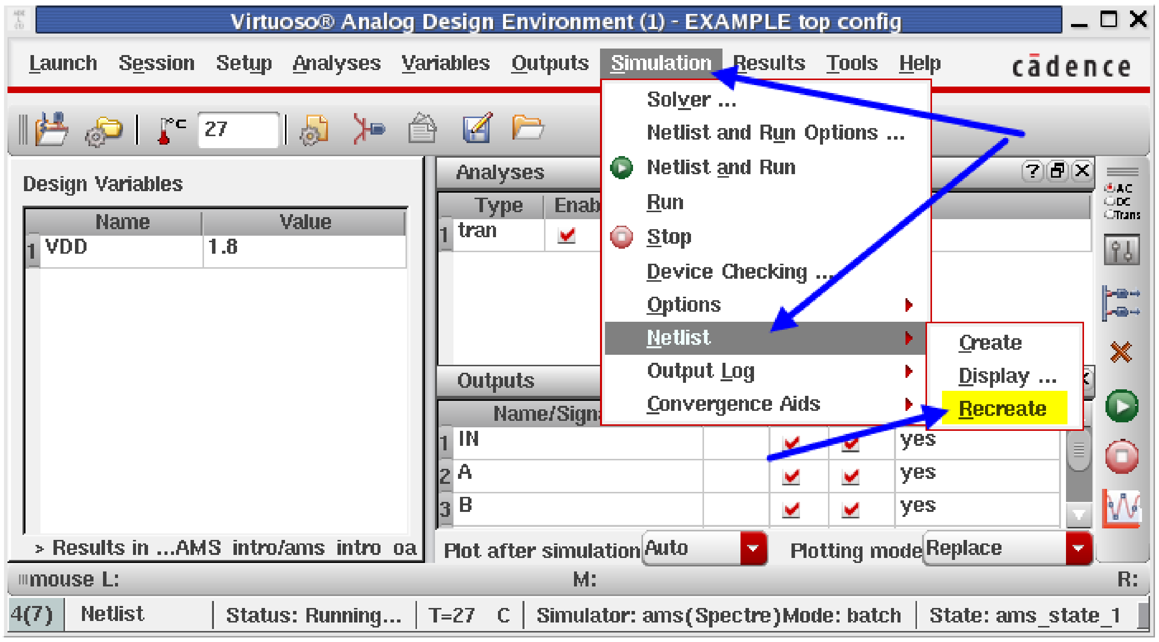
Simulate ● Batch (normal) ○ Interactive (debugger)

- **Batch** runs the simulation in ADE just like a spectre simulation would run. Hit the run button and the ViVA waveform window opens.
- **Interactive** runs the simulation in the ***Simvision Debug Environment*** which provides full debugging capabilities for the digital portion of the design as well as probing and plotting waveforms from the analog.

The final set of options in the run mode form are for save/restart. These options work the same with AMS as they do for regular Spectre simulations.

26) **Click cancel to close the Netlister and Run Options dialog**

27) In ADE, pick **Simulation → Netlist → Recreate**


Virtuoso® Analog Design Environment (1) - EXAMPLE top config
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
Solver ...
Netlist and Run Options ...
Netlist and Run
Run
Stop
Device Checking ...
Options
Netlist
Output Log
Convergence Aids
Create
Display ...
Recreate
Design Variables
Analyses
Outputs
Plot after simulation Auto
Plotting mode Replace

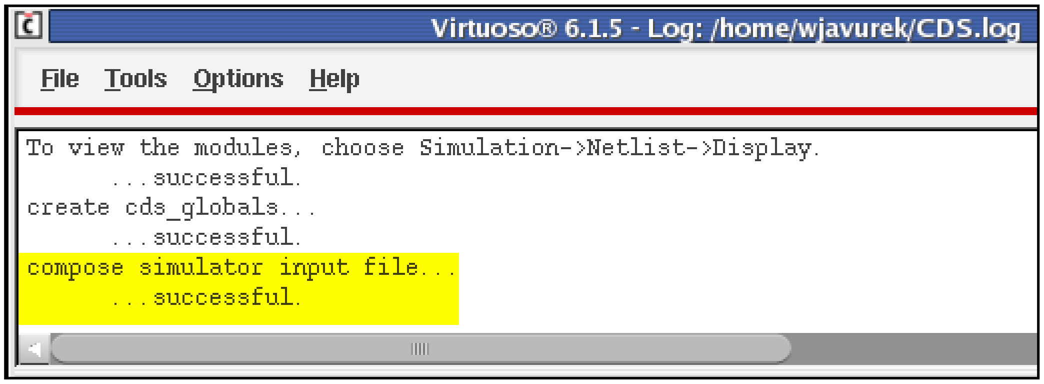
	Name	Value
1	VDD	1.8

Results in ...AMS intro/ams intro oa

mouse L: M: R:

4(7) Netlist | Status: Running... | T=27 C | Simulator: ams(Spectre)Mode: batch | State: ams_state_1

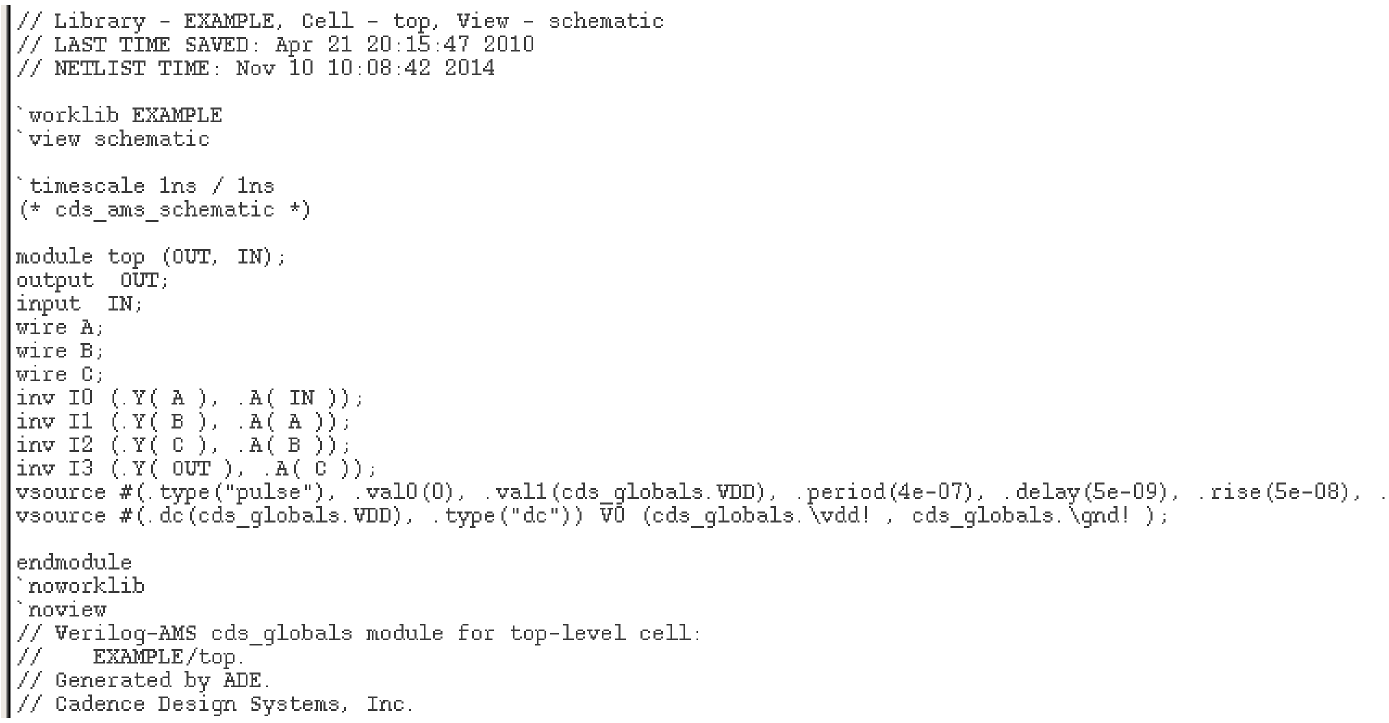
After the netlisting completes, you will see a message in the CIW.



```
To view the modules, choose Simulation->Netlist->Display.
        ...successful.
create cds_globals...
        ...successful.
compose simulator input file...
        ...successful.
```

28) Pick **Simulation → Netlist → Display**

This will display the OSS netlist. Scroll down to see the module for the top level schematic. This is called **module top** in the netlist.



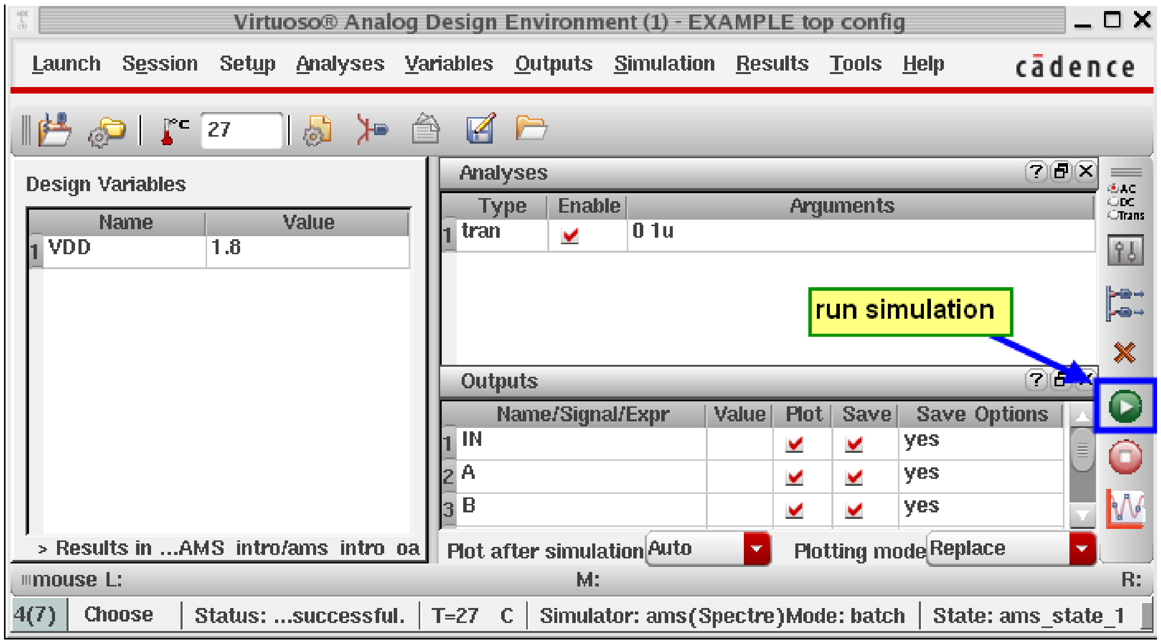
```
// Library - EXAMPLE, Cell - top, View - schematic
// LAST TIME SAVED: Apr 21 20:15:47 2010
// NETLIST TIME: Nov 10 10:08:42 2014

`worklib EXAMPLE
`view schematic

`timescale 1ns / 1ns
(* cds_ams_schematic *)

module top (OUT, IN);
output  OUT;
input  IN;
wire A;
wire B;
wire C;
inv I0 (.Y( A ), .A( IN ));
inv I1 (.Y( B ), .A( A ));
inv I2 (.Y( C ), .A( B ));
inv I3 (.Y( OUT ), .A( C ));
vsource #(.type("pulse"), .val0(0), .val1(cds_globals.VDD), .period(4e-07), .delay(5e-09), .rise(5e-08), .
vsource #(.dc(cds_globals.VDD), .type("dc")) V0 (cds_globals.\vdd! , cds_globals.\gnd! );

endmodule
`noworklib
`noview
// Verilog-AMS cds_globals module for top-level cell:
//    EXAMPLE/top.
// Generated by ADE.
// Cadence Design Systems, Inc.
```

The format of the netlist is ***structural Verilog-AMS***.

29) **Close the netlist window**

30) **Run the simulation**

- Pick **Simulation → Netlist and Run** or use the "green run" icon on the lower right of the ADE GUI


Virtuoso® Analog Design Environment (1) - EXAMPLE top config
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
run simulation
AC
DC
Trans

Design Variables

	Name	Value
1	VDD	1.8

> Results in ...AMS intro/ams intro oa

Analyses

	Type	Enable	Arguments
1	tran	✓	0 1u

Outputs

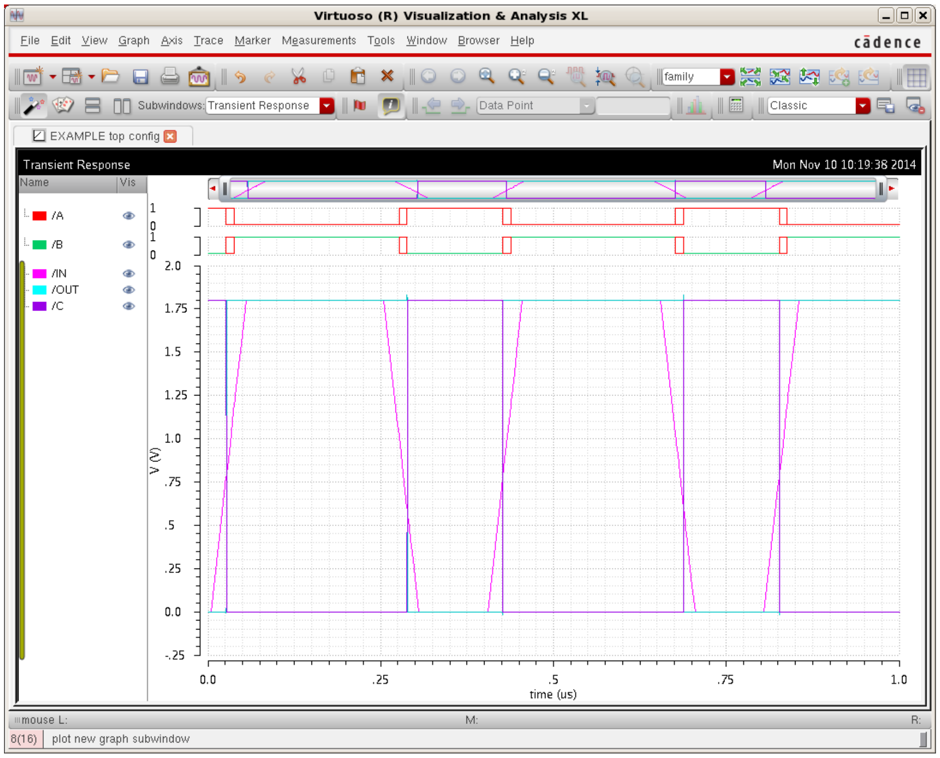
	Name/Signal/Expr	Value	Plot	Save	Save Options
1	IN		✓	✓	yes
2	A		✓	✓	yes
3	B		✓	✓	yes

Plot after simulation Auto   Plotting mode Replace

mouse L: M: R:

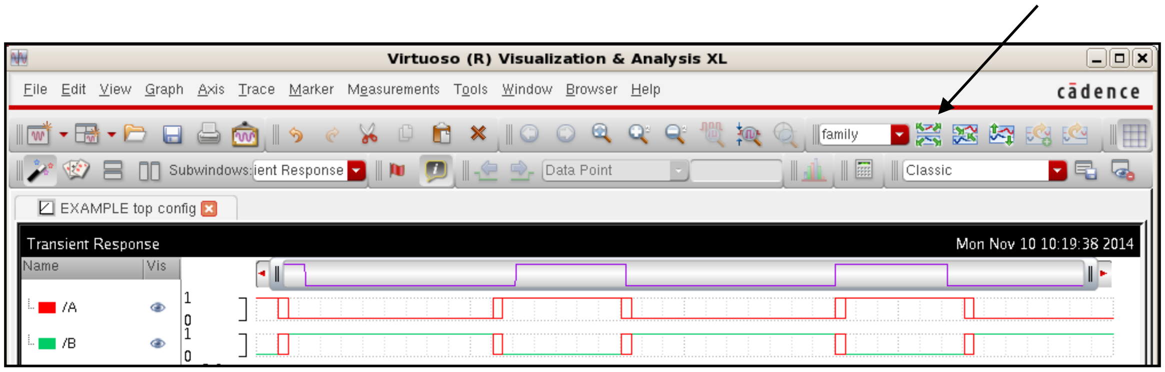
4(7) | Choose | Status: ...successful. | T=27 C | Simulator: ams(Spectre)Mode: batch | State: ams_state_1

ViVA will open with the waveforms when the simulation finishes.

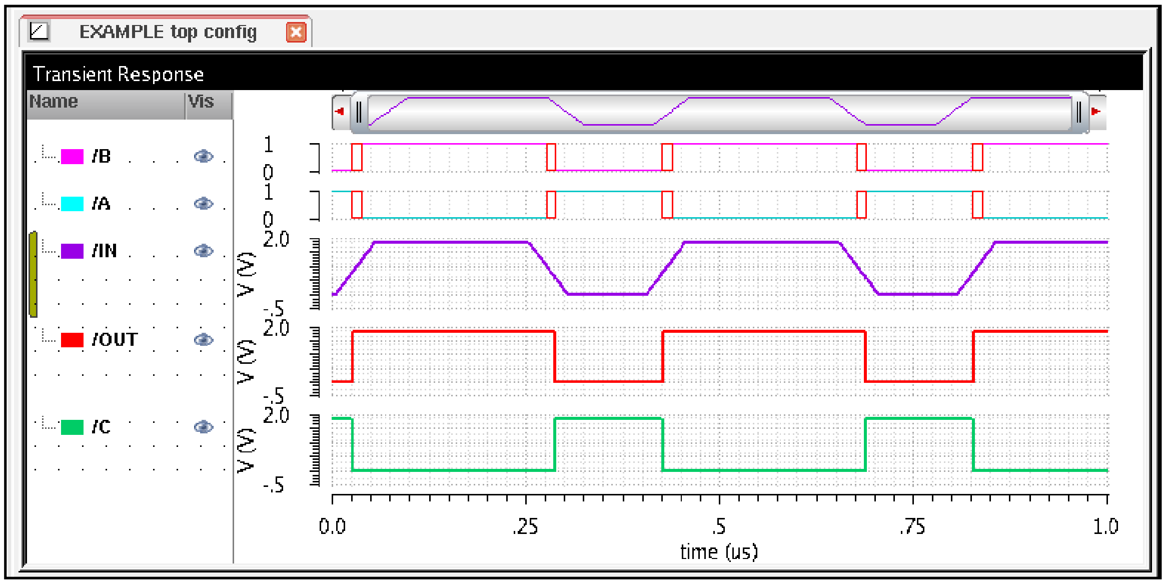


31) **Split the analog traces in to separate strips with the strip mode icon in ViVA**

Note: The function of the split to strips icon is to split the current strip. Therefore, the strip with the analog traces has to be active to before clicking the strip mode icon.

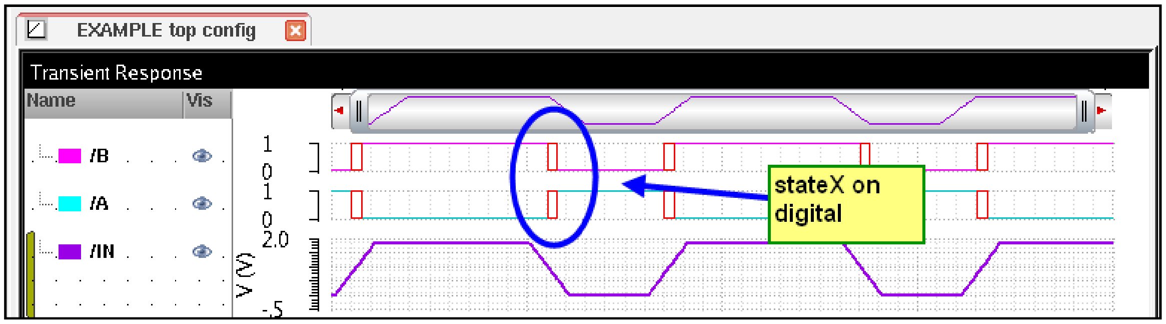

Virtuoso (R) Visualization & Analysis XL
File Edit View Graph Axis Trace Marker Measurements Tools Window Browser Help
cādence
family
Subwindows: ient Response
Data Point
Classic
EXAMPLE top config
Transient Response
Mon Nov 10 10:19:38 2014
Name
Vis
/A
/B

Activating strip mode will show the analog as separate strips in the plot window.



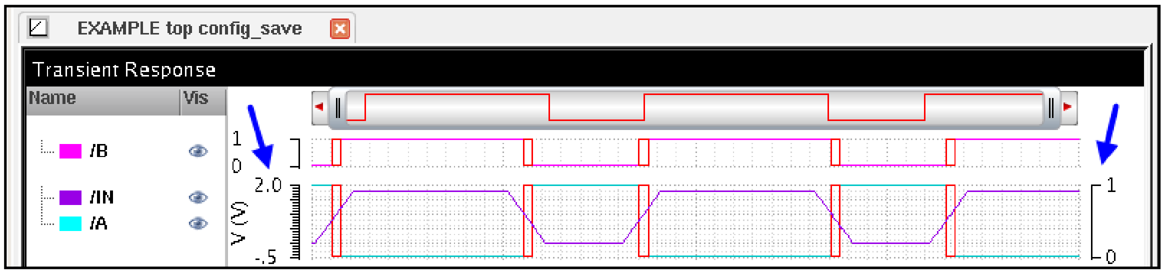
ViVA supports drag-and-drop to rearrange strips and which can be used to organize the waveforms. Just click on the waveform and drag it to the desired location.

The slow moving edges on the input signal causes the unknown region (state = x) in the digital waveforms.



ViVA also allows dragging and dropping of digital waveforms on top of analog traces.

32) **Drag the** `/A` **digital trace on top of** `/IN` **as shown below**



Note the vertical axis for the digital trace is on the right side.

33) **Close ViVA**

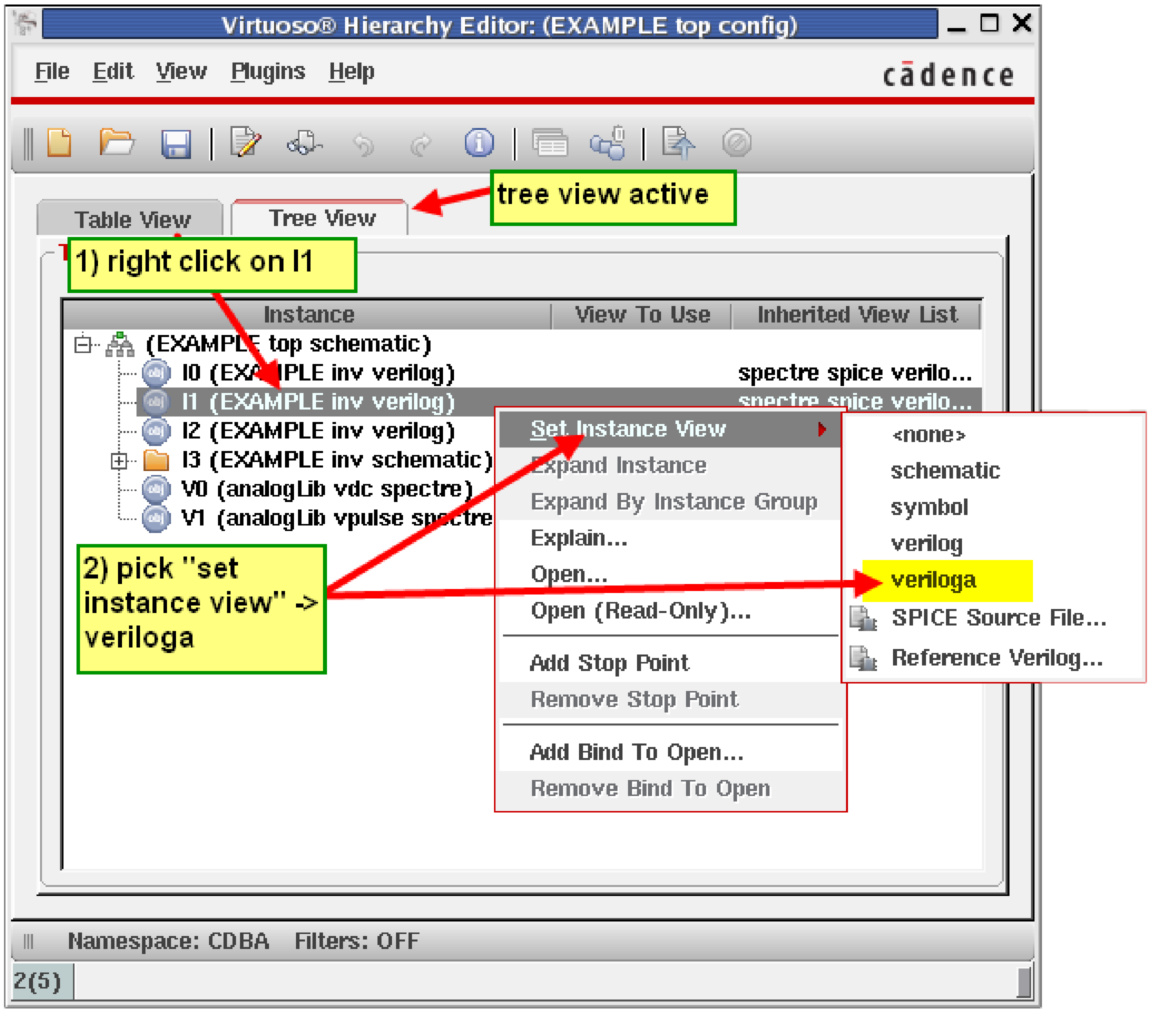
34) **Scroll to the top of the simulation log to see the irun command line. Near the end of the irun command line you will see where the design data is referenced (yellow highlight below).**

```
irun: 13.20-s021: (c) Copyright 1995-2014 Cadence Design Systems, Inc.
TOOL:   irun    13.20-s021: Started on Nov 10, 2014 at 10:19:02 EST
irun
        -f irunArgs
                -clean
                -UNBUFFERED
                -noupdate
                -errormax 50
                -status
                ...
                ./netlist.vams
                -f ./textInputs -amscompilefile
"file:/usr2/AMS_RAK/AMS_intro/ADE/EXAMPLE/inv/verilog/verilog.v
lib:EXAMPLE cell:inv view:verilog"
```

The `netlist.vams` file is the netlist for the schematic. The Verilog module used in the simulation is referenced from the design library with
`$cwd/<library>/<cell>/<view>/verilog.v`

35) **Close the simulation run log.**

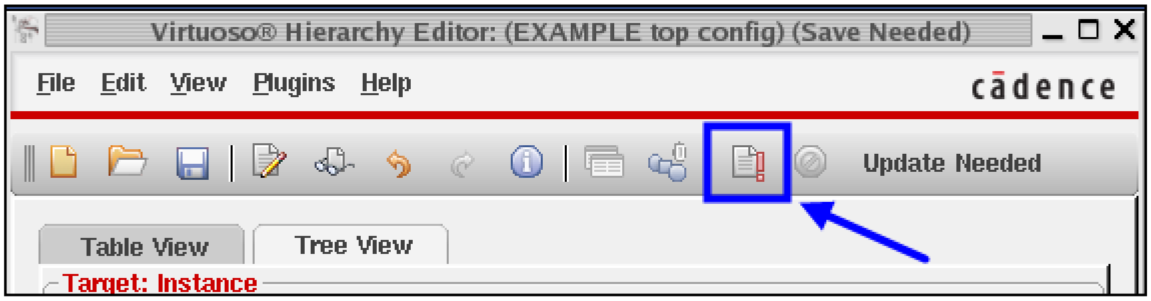
36) In the hierarchy editor, instance I1 is set to be "verilog". **Right click on I1** and **pick "set instance view" to be veriloga**.



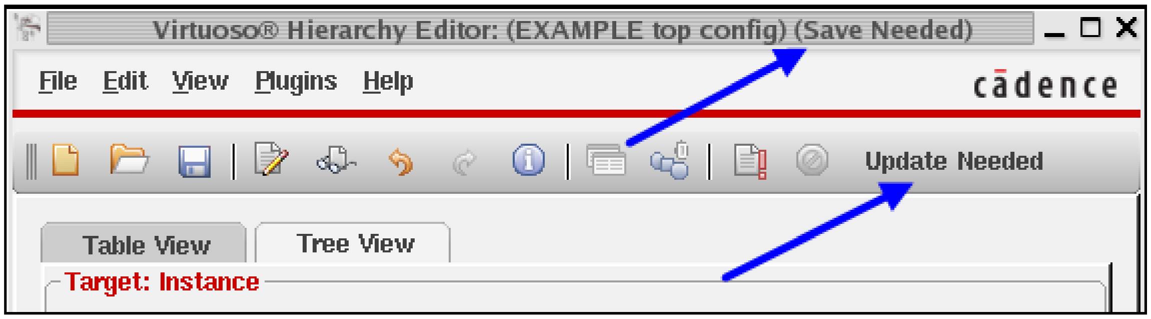
Very Important

After changing the configuration, the config view has to be saved to take effect.

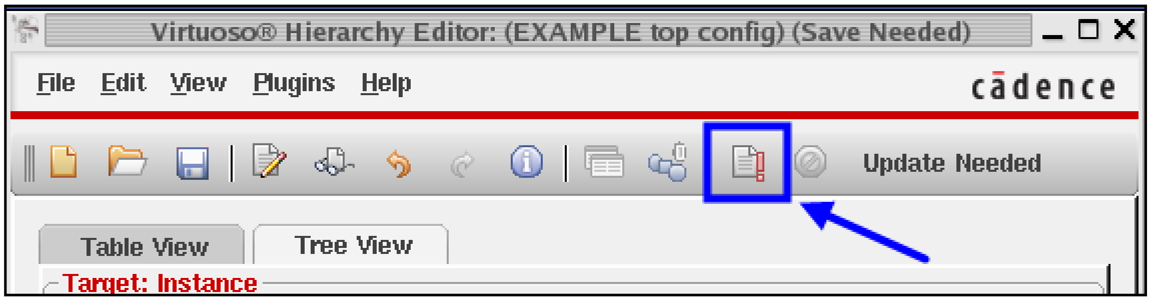
- The update icon near the top of the window will turn red


Virtuoso® Hierarchy Editor: (EXAMPLE top config) (Save Needed)
File Edit View Plugins Help
cadence
Update Needed
Table View
Tree View
Target: Instance

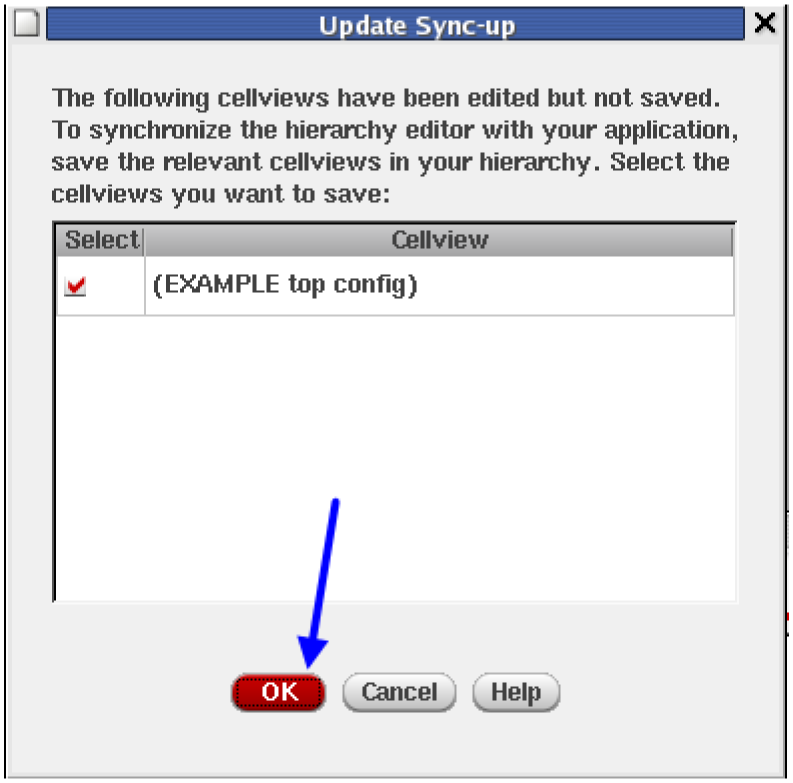
- The window banner will also say "save needed" and there will be a text message in the menu bar for "update needed"


Virtuoso® Hierarchy Editor: (EXAMPLE top config) (Save Needed)
File Edit View Plugins Help
cadence
Update Needed
Table View
Tree View
Target: Instance

37) **Click the update icon in the Hierarchy Editor**


Virtuoso® Hierarchy Editor: (EXAMPLE top config) (Save Needed)
File Edit View Plugins Help
cadence
Update Needed
Table View
Tree View
Target: Instance

38) **A window will open prompting to save the updated config** (see below). **Click OK**



The following cellviews have been edited but not saved. To synchronize the hierarchy editor with your application, save the relevant cellviews in your hierarchy. Select the cellviews you want to save:

Select	Cellview
✓	(EXAMPLE top config)

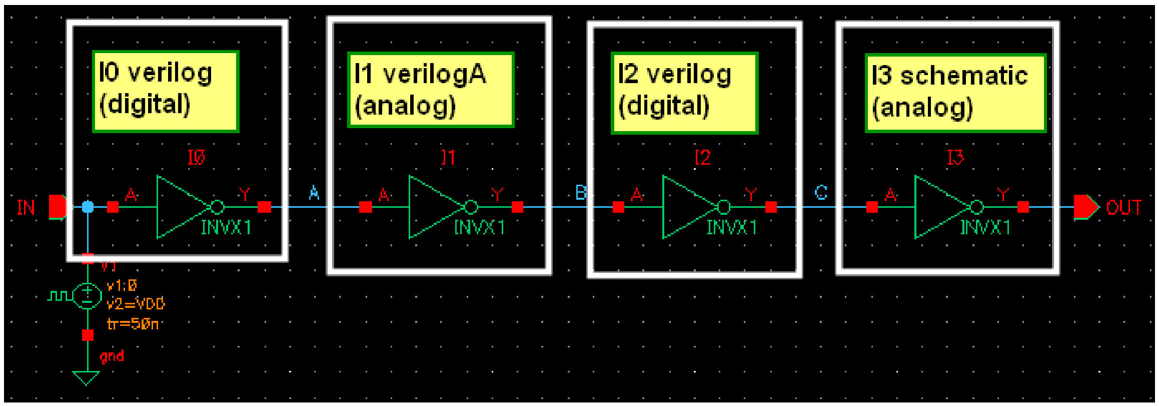
This change to config view (Hierarchy Editor) tells the simulation to use the veriloga (analog) representation for instance I1 instead verilog (digital).

The schematic order is

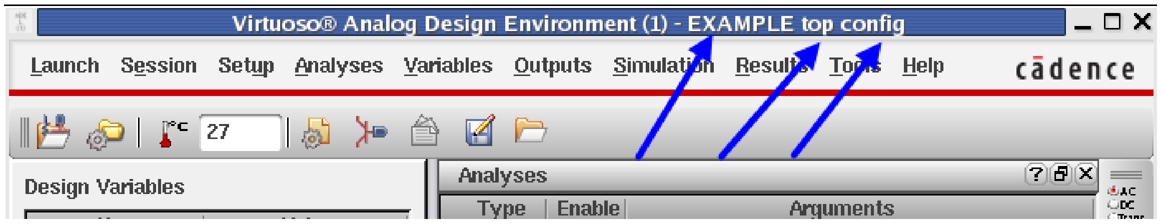
I0 → I1 → I2 → I3

Which will be (click on each instance in the hierarchy editor to see the corresponding schematic instance)

verilog (digital) → veriloga (analog) → verilog (digital) → schematic (analog)

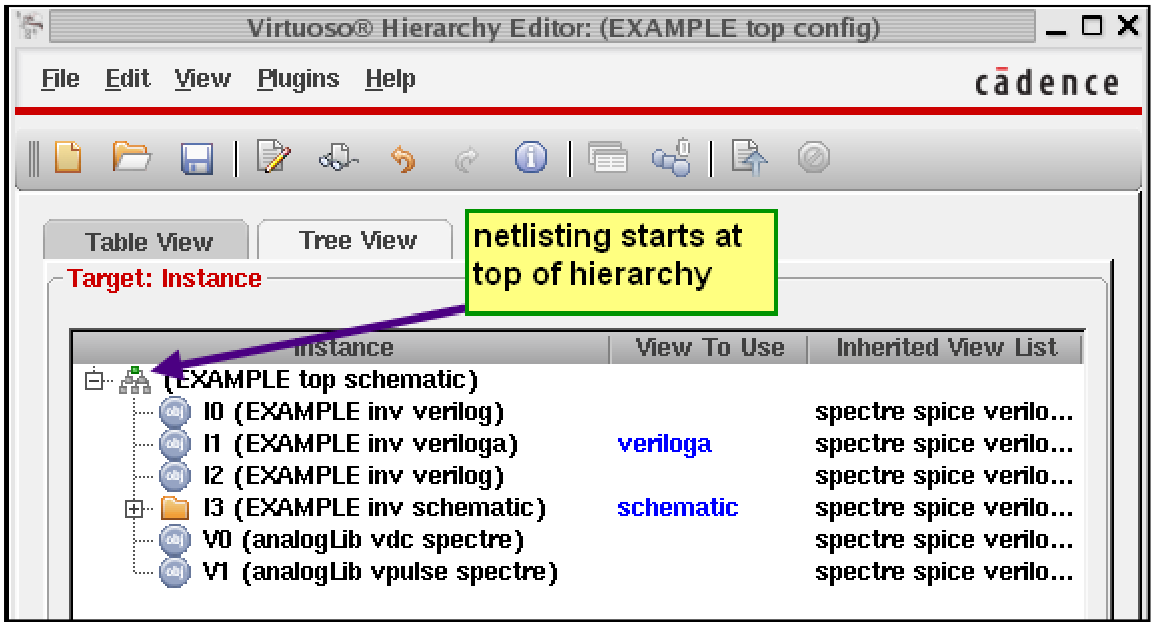


Recall, ADE is simulating the config view.


Virtuoso® Analog Design Environment (1) - EXAMPLE top config
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
27
Design Variables
Analyses
Type Enable Arguments

The netlister reads the config view to determine what view to use for each block when creating the netlist. Modifying the config view allows swapping in and out different representations (verilog, schematic, veriloga, parasitic extracted, etc.) for the design blocks without changing the schematic.

The netlister starts are the top of the tree in the config view and uses the views specified in the config

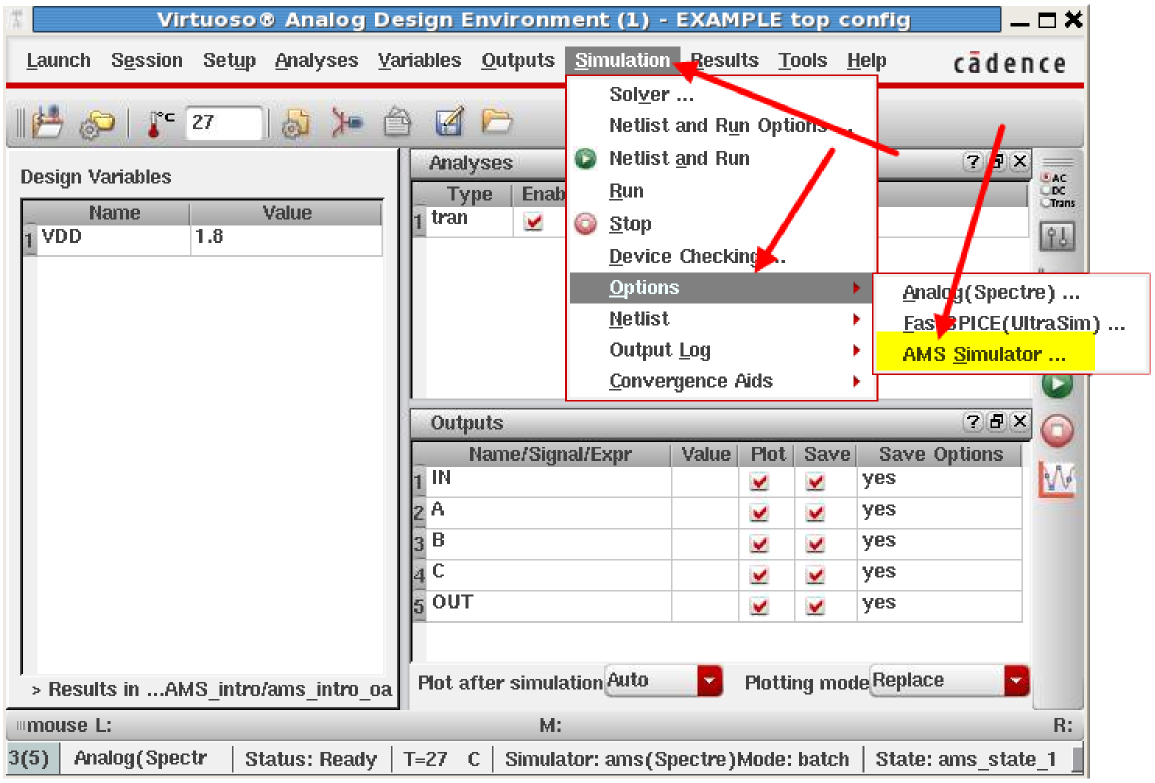

Virtuoso® Hierarchy Editor: (EXAMPLE top config)
File Edit View Plugins Help
cādence
Table View
Tree View
netlisting starts at top of hierarchy
Target: Instance

Instance	View To Use	Inherited View List
(EXAMPLE top schematic)		
I0 (EXAMPLE inv verilog)		spectre spice verilo...
I1 (EXAMPLE inv veriloga)	veriloga	spectre spice verilo...
I2 (EXAMPLE inv verilog)		spectre spice verilo...
I3 (EXAMPLE inv schematic)	schematic	spectre spice verilo...
V0 (analogLib vdc spectre)		spectre spice verilo...
V1 (analogLib vpulse spectre)		spectre spice verilo...

The ***inherited view*** (last column on the right side in the above screen capture) is the default set by the ***view list***. In this case I0, gets the default with is verilog.

For instance I1, the ***view_to_use*** (2nd column in the tree view above) overrides the default verilog and tells the netlister to use the **veriloga** representation instead. Likewise, instance I3 has an override set to use the schematic view.

39) In ADE, pick **Simulation → Options → AMS Simulator**



	Name	Value
1	VDD	1.8

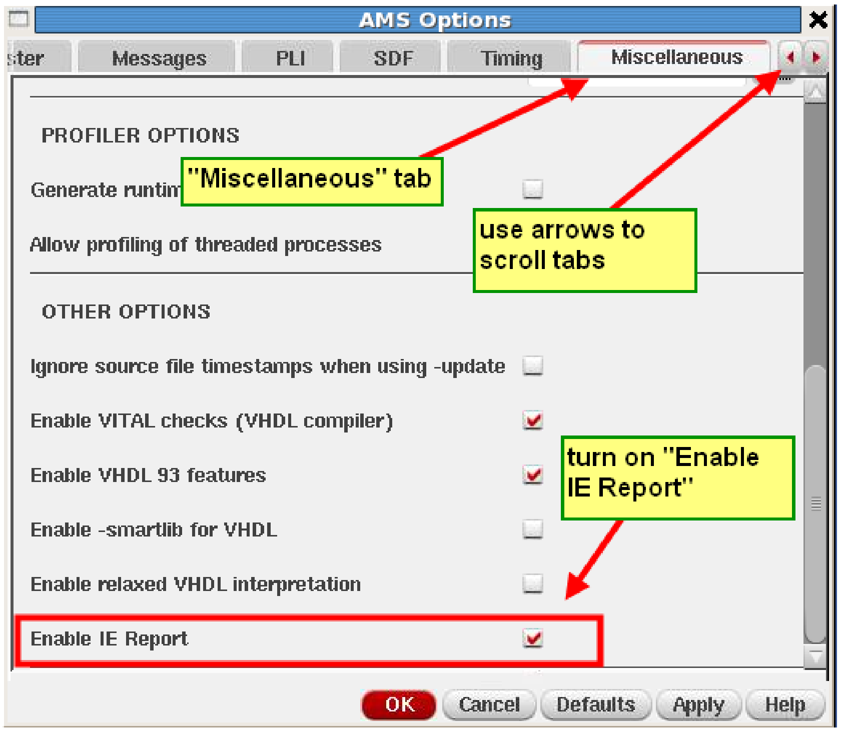
	Type	Enab
1	tran	✓

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	IN		✓	✓	yes
2	A		✓	✓	yes
3	B		✓	✓	yes
4	C		✓	✓	yes
5	OUT		✓	✓	yes

> Results in ...AMS_intro/ams_intro_oa

3(5) | Analog(Spectr | Status: Ready | T=27 C | Simulator: ams(Spectre)Mode: batch | State: ams_state_1

40) Pick the **Miscellaneous** tab and **enable IE Report** at the <u>bottom</u> as shown below



PROFILER OPTIONS

Generate runtim	☐
Allow profiling of threaded processes	

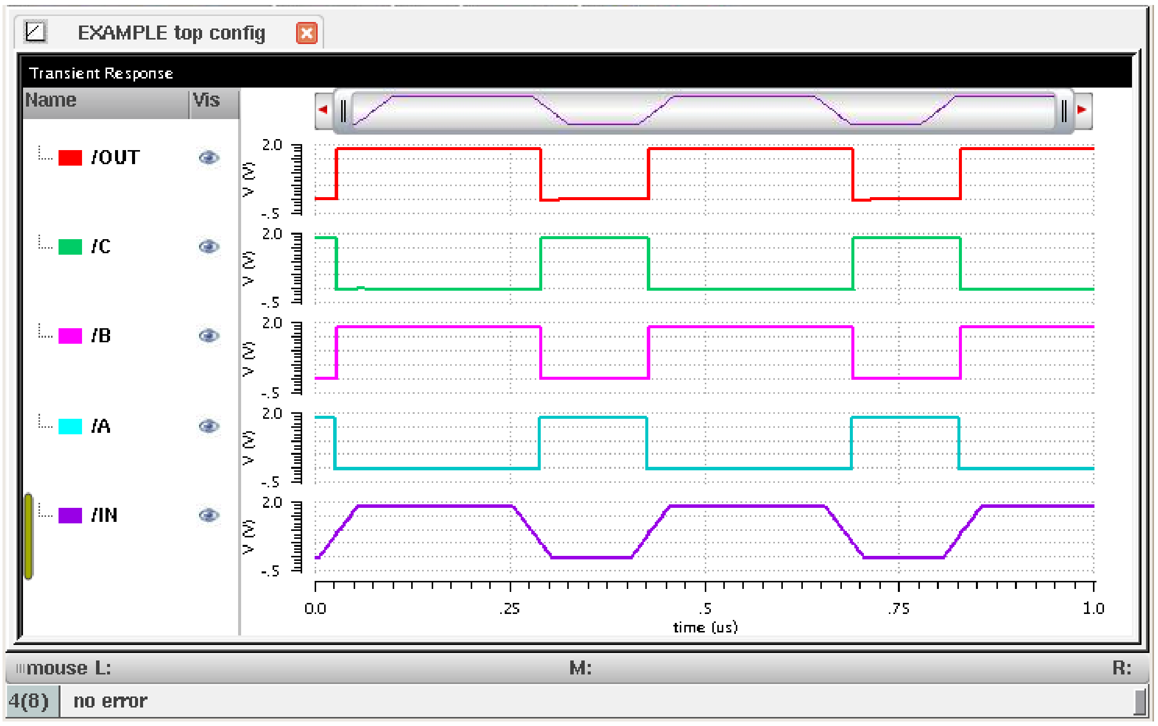
OTHER OPTIONS

Ignore source file timestamps when using -update	☐
Enable VITAL checks (VHDL compiler)	✓
Enable VHDL 93 features	✓
Enable -smartlib for VHDL	☐
Enable relaxed VHDL interpretation	☐
Enable IE Report	✓

41) **Click OK on the AMS Options form**

42) **Re-run the simulation**.

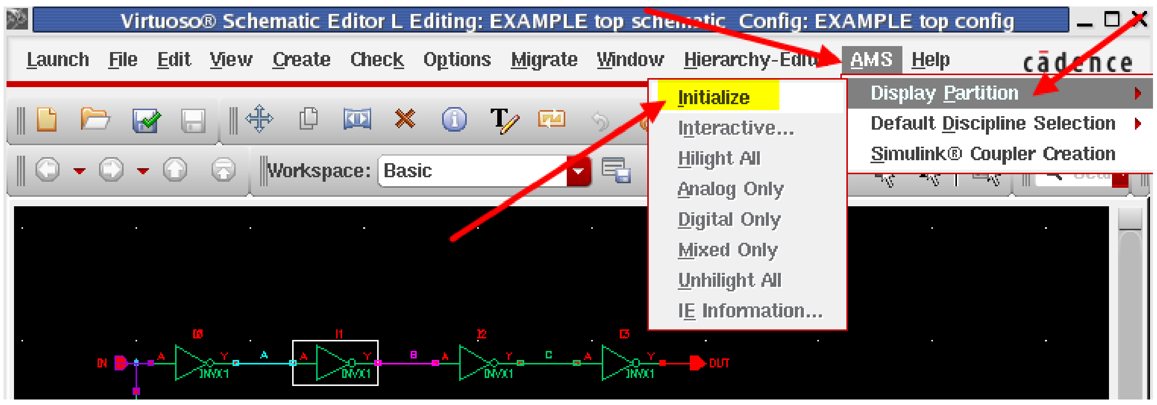
43) When **ViVA** opens, **change to strip mode**.



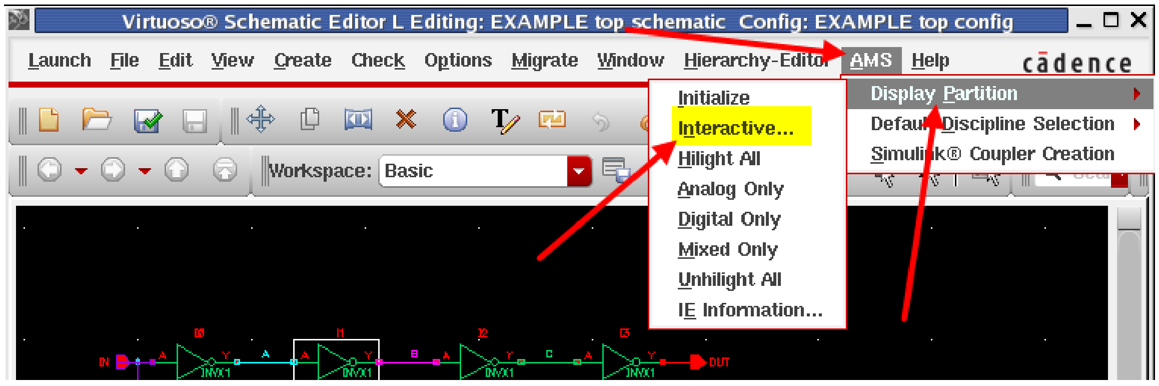
Note all the signals appear as analog in the waveform window.

44) **Close ViVA** but **leave the log files open.**

45) From the schematic pick **AMS → Display Partition → Initialize**

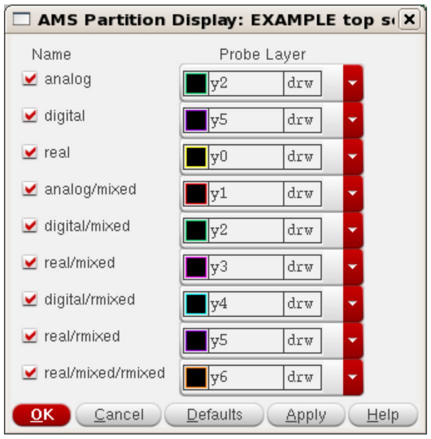

Virtuoso® Schematic Editor L Editing: EXAMPLE top schematic Config: EXAMPLE top config
Launch File Edit View Create Check Options Migrate Window Hierarchy-Editor AMS Help
cadence
Display Partition
Default Discipline Selection
Simulink® Coupler Creation
Initialize
Interactive...
Hilight All
Analog Only
Digital Only
Mixed Only
Unhilight All
IE Information...
Workspace: Basic

46) From the schematic window, pick **AMS → Display Partition → Interactive**


Virtuoso® Schematic Editor L Editing: EXAMPLE top schematic Config: EXAMPLE top config
Launch File Edit View Create Check Options Migrate Window Hierarchy-Editor AMS Help
cadence
Display Partition
Default Discipline Selection
Simulink® Coupler Creation
Initialize
Interactive...
Hilight All
Analog Only
Digital Only
Mixed Only
Unhilight All
IE Information...
Workspace: Basic

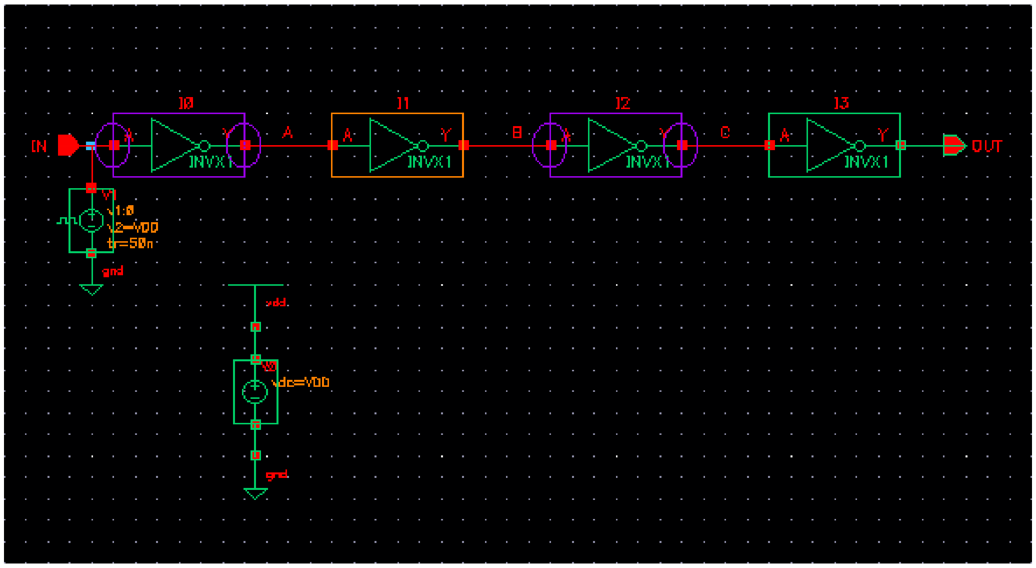
This will open the display partition dialog

47) **Click Apply** in the display partition dialog


AMS Partition Display: EXAMPLE top s

Name	Probe Layer	
✓ analog	y2	drw
✓ digital	y5	drw
✓ real	y0	drw
✓ analog/mixed	y1	drw
✓ digital/mixed	y2	drw
✓ real/mixed	y3	drw
✓ digital/rmixed	y4	drw
✓ real/rmixed	y5	drw
✓ real/mixed/rmixed	y6	drw

This will annotate the partitioning information (analog/digital) on the schematic following the color scheme set by the probe layer as shown below.



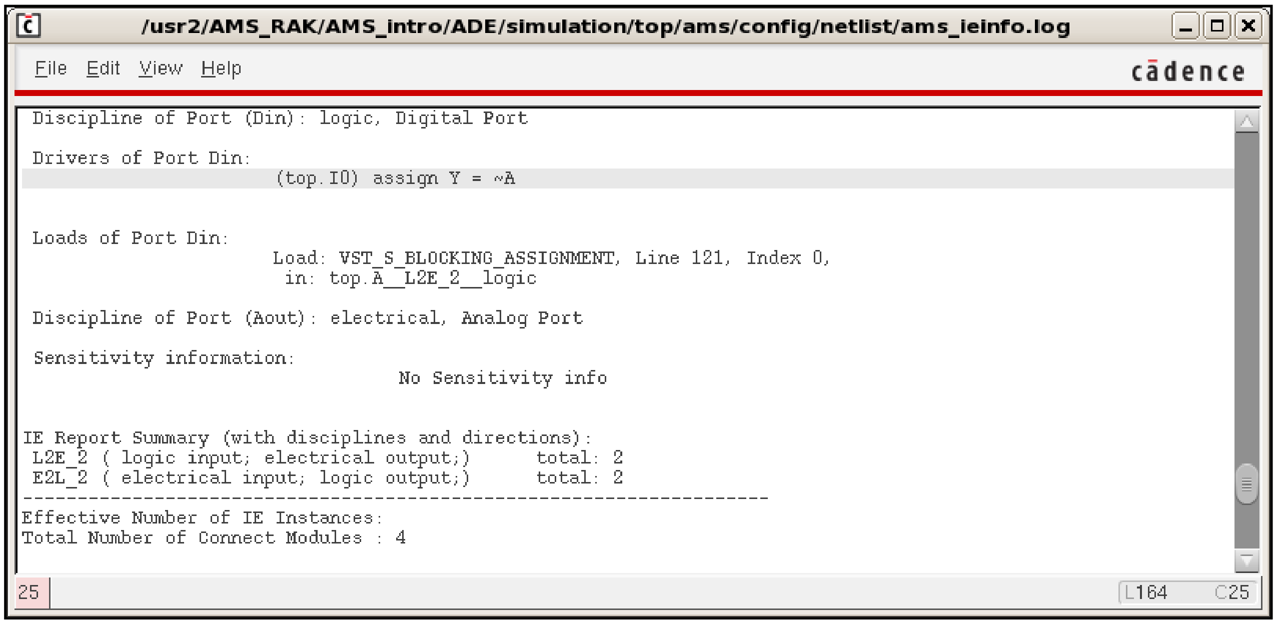
The highlight colors are set by the y0-y9 layers in the PDK. For this example, green is analog and purple is digital.

The circles represent where connect modules have been inserted by the simulation.

The connect modules are inserted on the digital ports which is why all the nets show up as analog in the waveform window.

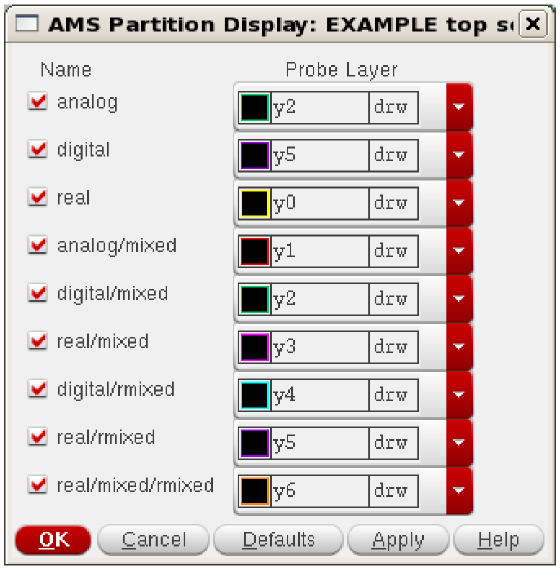
It can also be seen from the AMS partition display that there are 4 connect modules (sometimes called "Interface Elements"). Again, these are 4 circles on the inverter ports in the screen capture above.

48) **Compare this to the IE report in the ams_ieinfo.log file**. This file will be behind the irun.log file. The IE report appears at the bottom of the file. Alternatively, you can use File → Find and search for “IE Report Summary”


/usr2/AMS_RAK/AMS_intro/ADE/simulation/top/ams/config/netlist/ams_ieinfo.log
File Edit View Help
cādence
25
L164 C25

```
 Discipline of Port (Din): logic, Digital Port

 Drivers of Port Din:
                       (top.I0) assign Y = ~A


 Loads of Port Din:
                       Load: VST_S_BLOCKING_ASSIGNMENT, Line 121, Index 0,
                        in: top.A__L2E_2__logic

 Discipline of Port (Aout): electrical, Analog Port

 Sensitivity information:
                                  No Sensitivity info


IE Report Summary (with disciplines and directions):
 L2E_2 ( logic input; electrical output;)      total: 2
 E2L_2 ( electrical input; logic output;)      total: 2
--------------------------------------------------------------------
Effective Number of IE Instances:
Total Number of Connect Modules : 4
```

In addition to analog and digital, the AMS partitioning display also shows "analog/mixed" and "digital/mixed" and “real/mixed” variations.



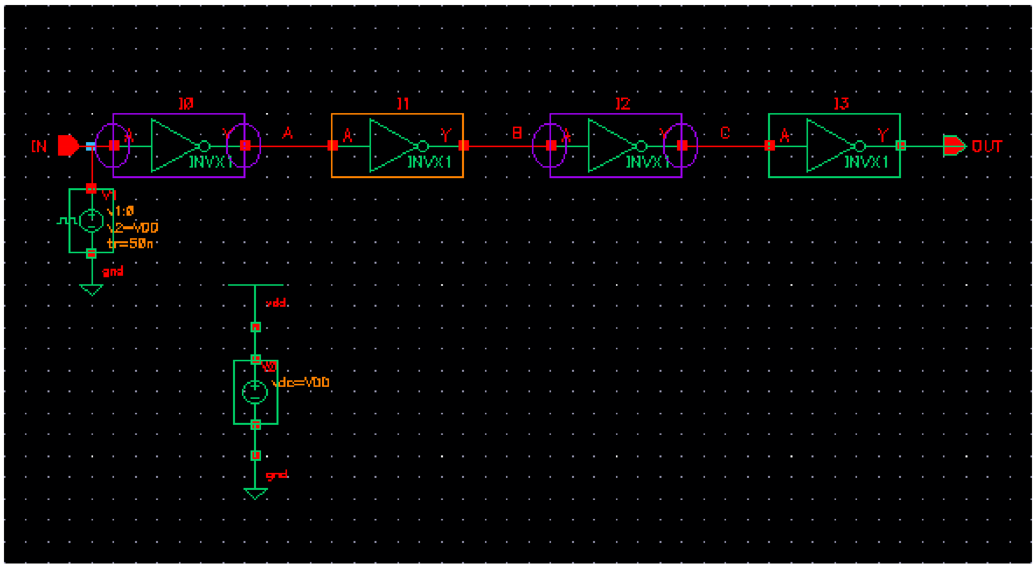
AMS Partition Display: EXAMPLE top s

Name	Probe Layer	
✓ analog	y2	drw
✓ digital	y5	drw
✓ real	y0	drw
✓ analog/mixed	y1	drw
✓ digital/mixed	y2	drw
✓ real/mixed	y3	drw
✓ digital/rmixed	y4	drw
✓ real/rmixed	y5	drw
✓ real/mixed/rmixed	y6	drw

- **Digital/Mixed** means digital at the current level of the hierarchy (from where you opened the Partition Display form) and mixed somewhere else in the hierarchy.

- **Analog/Mixed** means analog at the current level and mixed somewhere else in the hierarchy.

In this example, nets A, B and C are analog/mixed (red) which means they are analog on the current schematic and are mixed (analog and digital) down inside the schematic hierarchy.

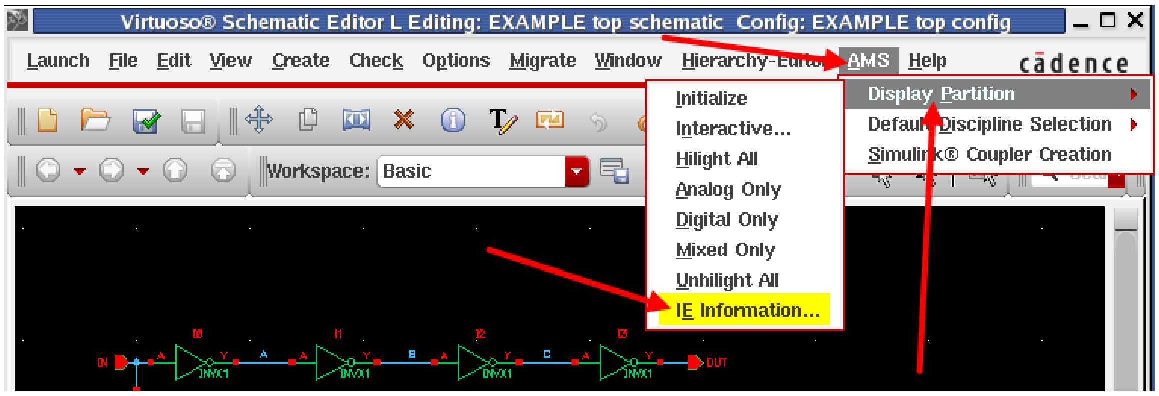


For an example, take net B. Instance I1 drives net B which is an analog inverter so the signal is analog on the top level schematic. However descending into instance I1 (verilog) net B is digital, and therefore is represented as analog/mixed in the AMS partitioning display.

Analog/mixed nets appear as analog in the waveform window since the signal is analog where the probe is placed. This is why the ViVA plot window shows all the waveforms as analog even though the simulation has both analog and digital. Nets `IN`, `A`, `B`, and `C` are analog/mixed (red) and `OUT` is analog (green)

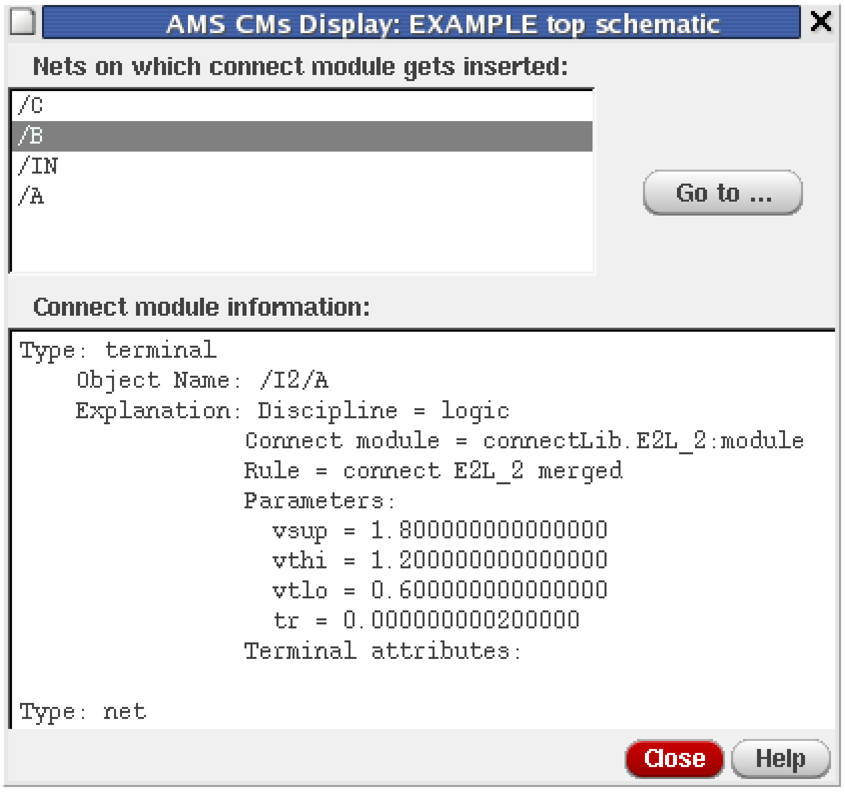
49) **Cancel the display partition dialog**

50) Pick **AMS → Display Partition → IE Information**


Virtuoso® Schematic Editor L Editing: EXAMPLE top schematic Config: EXAMPLE top config
Launch File Edit View Create Check Options Migrate Window Hierarchy-Editor AMS Help
cādence
Workspace: Basic
Initialize
Interactive...
Hilight All
Analog Only
Digital Only
Mixed Only
Unhilight All
IE Information...
Display Partition
Default Discipline Selection
Simulink® Coupler Creation
IN
A
B
C
OUT
INVX1

This will open a window that shows the details of the connect modules used in the simulation.

51) **Click on net B in the IE Info window**



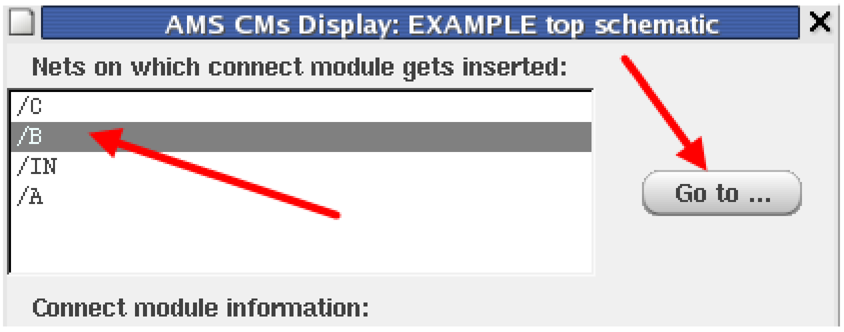
Nets on which connect module gets inserted:

```
/C
/B
/IN
/A
```

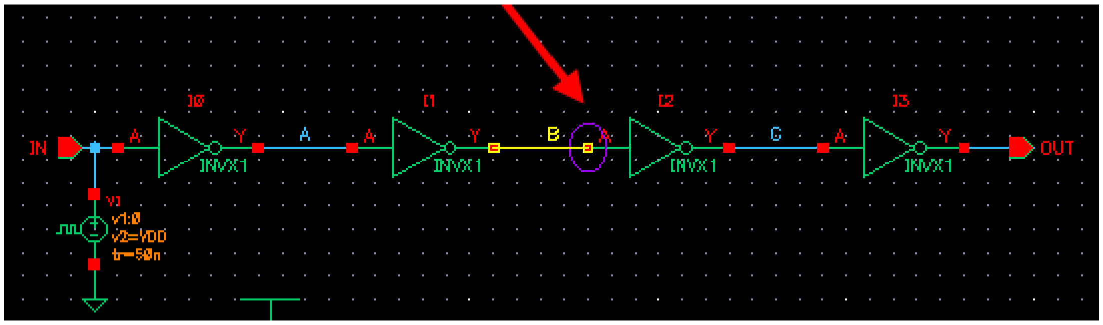
Connect module information:

```
Type: terminal
    Object Name: /I2/A
    Explanation: Discipline = logic
                 Connect module = connectLib.E2L_2:module
                 Rule = connect E2L_2 merged
                 Parameters:
                   vsup = 1.800000000000000
                   vthi = 1.200000000000000
                   vtlo = 0.600000000000000
                   tr = 0.000000000200000
                 Terminal attributes:

Type: net
```

52) **Click the "Go to" button in the IE info window**


AMS CMs Display: EXAMPLE top schematic
Nets on which connect module gets inserted:
/C
/B
/IN
/A
Go to ...
Connect module information:

Clicking the "go to" button highlights the schematic where the connect module is located as shown in the screen capture below.



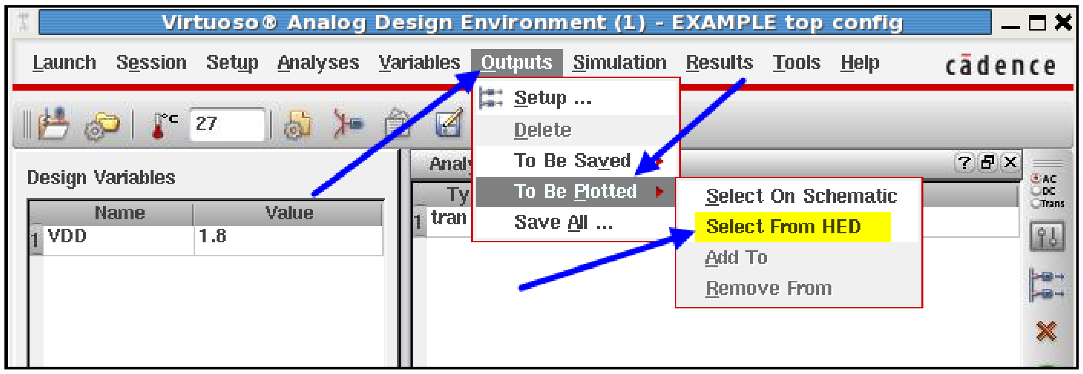
53) **Close the IE info window**

54) **Close the ams_ieinfo window and irun log window**

From the AMS partitioning display on the schematic, we saw that all the nets in the schematic for this configuration (meaning the config view) are all analog and analog/mixed. This is why all the waveforms show up as analog in the waveform viewer.

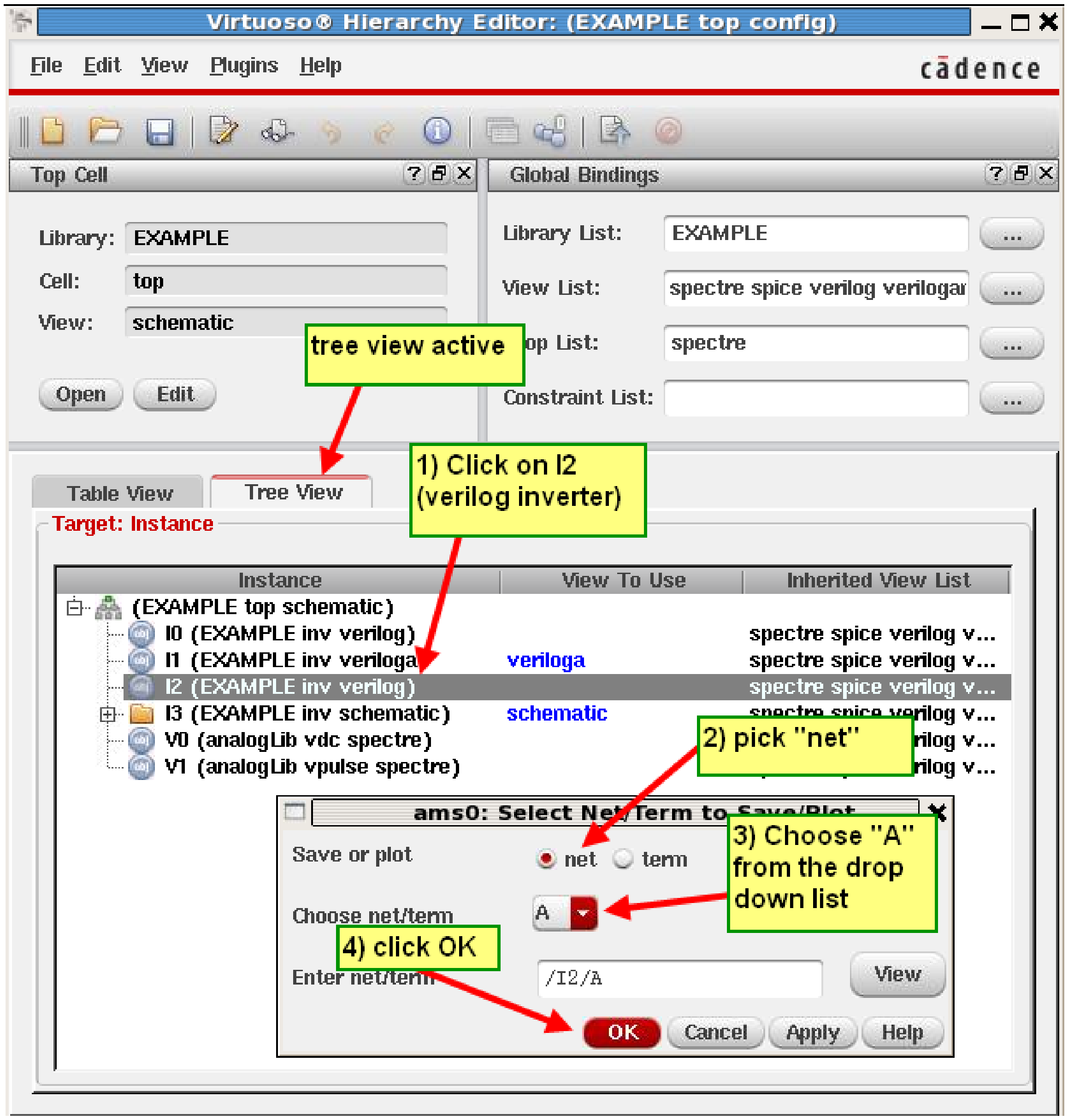
Now, let's look at how to probe a digital net down inside one of the digital inverters from ADE.

55) Go back to **ADE** and pick **outputs → to be plotted → select from HED**



	Name	Value
1	VDD	1.8

56) **In the Hierarchy Editor, click on I2** which is configured to be a verilog inverter as shown below


Virtuoso® Hierarchy Editor: (EXAMPLE top config)
File Edit View Plugins Help
cādence
Top Cell
Global Bindings
Open
Edit
tree view active
1) Click on I2 (verilog inverter)
Table View
Tree View
Target: Instance
2) pick "net"
ams0: Select Net/Term to Save/Plot
3) Choose "A" from the drop down list
4) click OK
OK
Cancel
Apply
Help
View

Library: EXAMPLE
Cell: top
View: schematic

Library List: EXAMPLE
View List: spectre spice verilog verilogar
Stop List: spectre
Constraint List:

Instance	View To Use	Inherited View List
(EXAMPLE top schematic)		
I0 (EXAMPLE inv verilog)		spectre spice verilog v...
I1 (EXAMPLE inv verilog)	veriloga	spectre spice verilog v...
I2 (EXAMPLE inv verilog)		spectre spice verilog v...
I3 (EXAMPLE inv schematic)	schematic	spectre spice verilog v...
V0 (analogLib vdc spectre)		spectre spice verilog v...
V1 (analogLib vpulse spectre)		spectre spice verilog v...

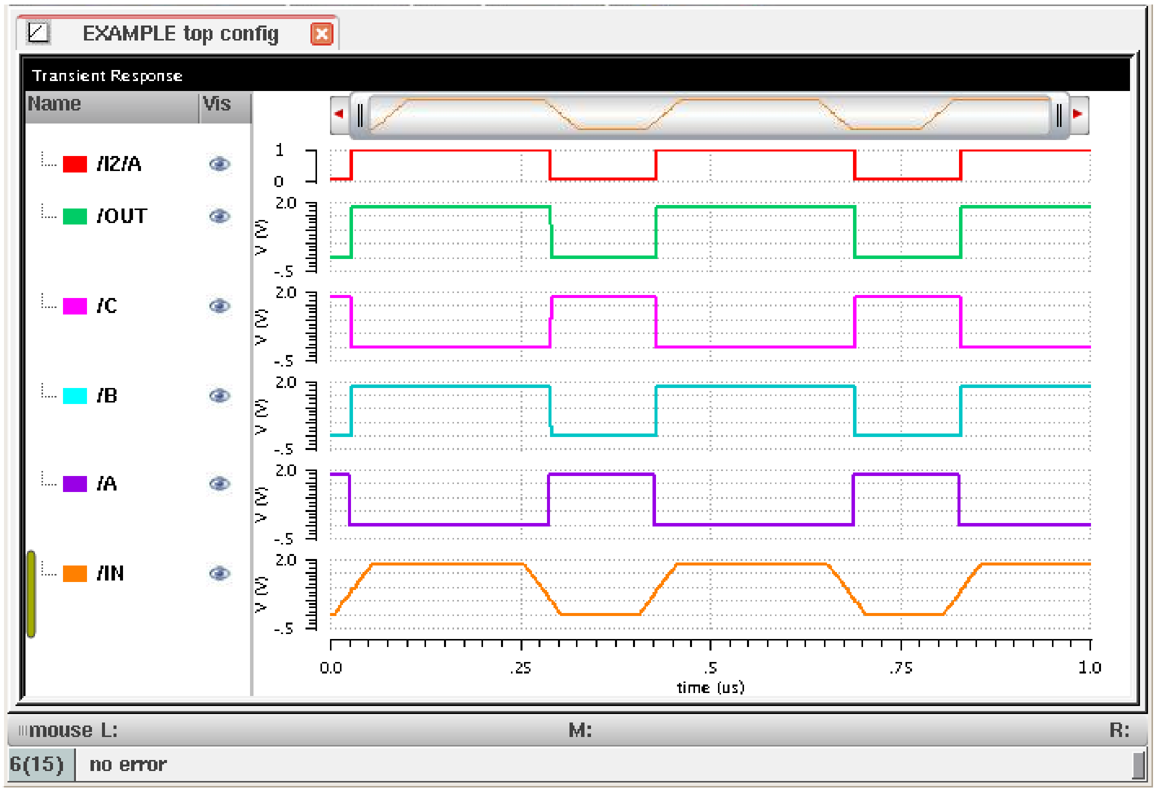
Save or plot [✓ net] term
Choose net/term A
Enter net/term /I2/A

57) After clicking on I2, a dialog will prompt for which net to probe. **Choose net A and click OK as shown above.**

58) **Re-run the simulation.** If prompted to save outputs that will be plotted, click yes.

When the simulation completes, ViVA will open with the digital waveform at the top as shown below. In this case, we have plotted `I2.A` which is the digital equivalent of net "B". In other words, the probes are placed on both sides of the interface element (connect module) which is on the input of inverter I2.

This technique of placing probes on both sides of a connect module can be useful for debugging when the simulation shows problems with signals moving between analog and digital blocks.



59) **Close ViVA and irun log window and ams_ieinfo log window**. Leave the schematic and ADE open for part II.

II. Resolving setup errors in ADE

For this section, the same design is used and you can skip to step #3 below if the schematic and ADE are still open from part 1.

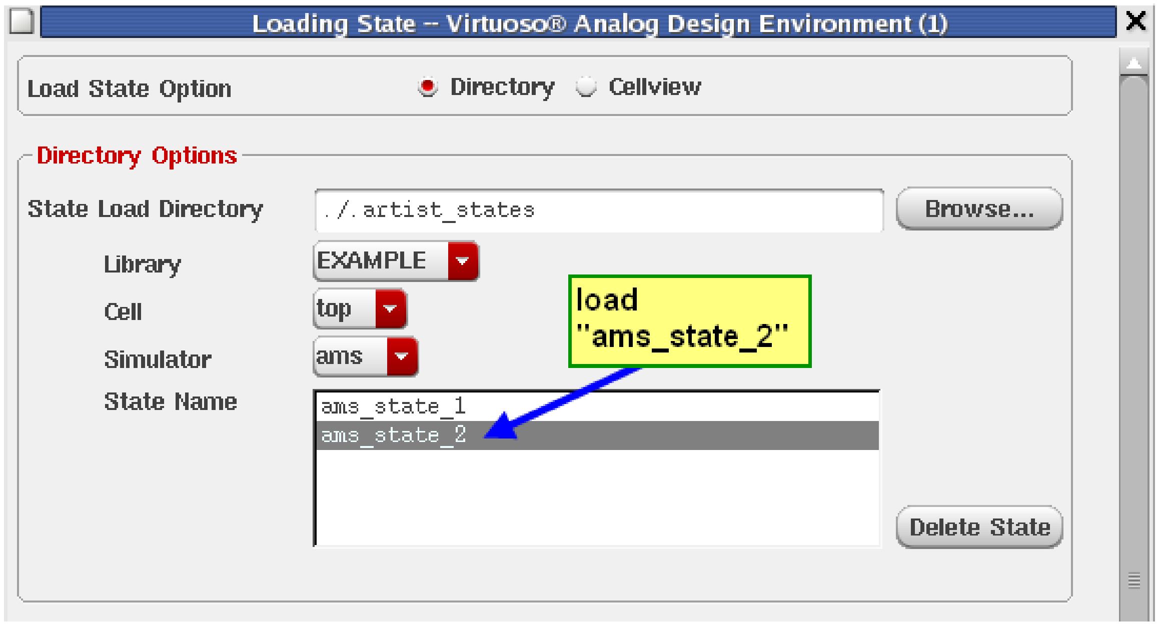
1) **If needed start virtuoso and open the design** (steps 1-6 in part 1)

   Library = EXAMPLE
   Cell = top
   View = config

   - When prompted, open both the config view and schematic.

2) **If needed, start ADE and set the simulator to be AMS** (steps 9-13 in part 1)

3) From ADE, pick **Session → Load State** and load **state = ams_state_2**



Loading State -- Virtuoso® Analog Design Environment (1)

Load State Option: [✓ Directory] Cellview

Directory Options

State Load Directory: ./.artist_states

Library: EXAMPLE

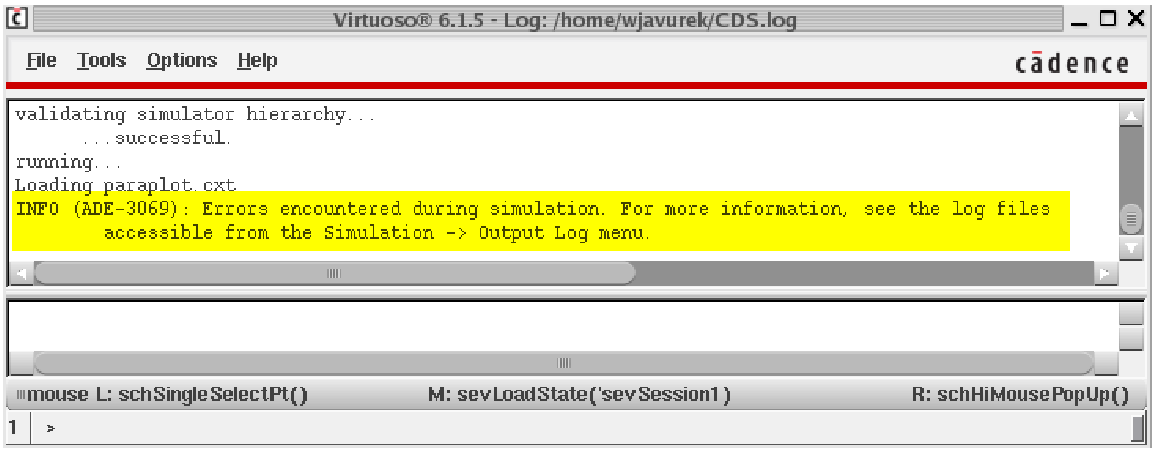
Cell: top

Simulator: ams

State Name: ams_state_1, [✓ ams_state_2]

4) **Run the simulation** (**simulation → netlist and run** or click the green Netlist and Run icon on lower right corner of the ADE window)

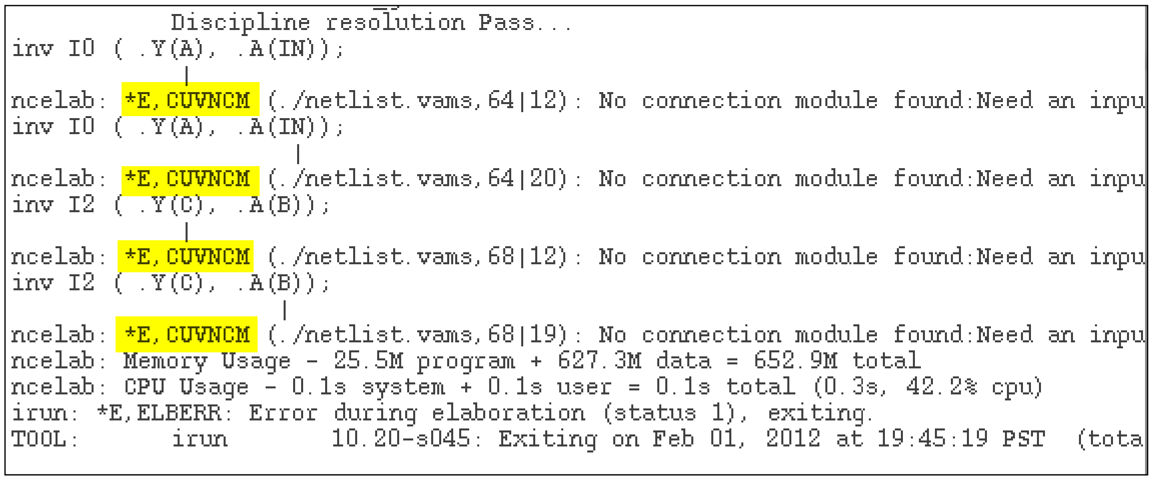
The CIW will show a message that the simulation failed.


Virtuoso® 6.1.5 - Log: /home/wjavurek/CDS.log
File Tools Options Help
cādence

```
validating simulator hierarchy...
        ...successful.
running...
Loading paraplot.cxt
INFO (ADE-3069): Errors encountered during simulation. For more information, see the log files
           accessible from the Simulation -> Output Log menu.
```

mouse L: schSingleSelectPt()
M: sevLoadState('sevSession1)
R: schHiMousePopUp()
1 >

The end of the simulation log shows elaboration errors



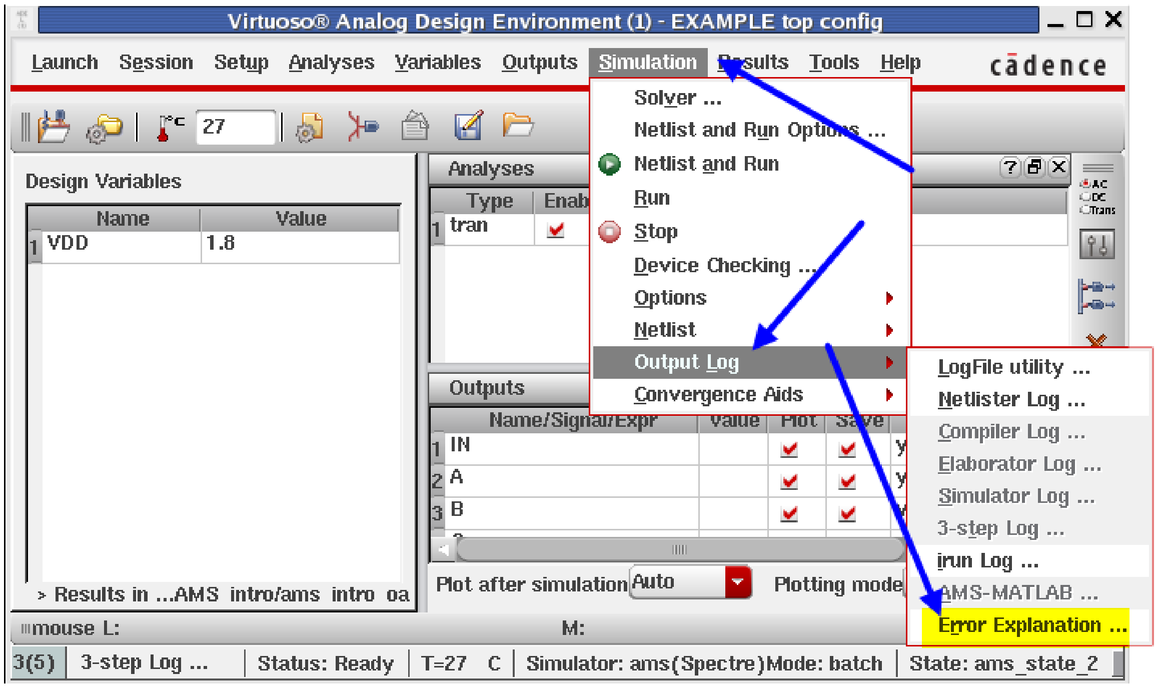
```
            Discipline resolution Pass...
inv I0 ( .Y(A), .A(IN));
             |
ncelab: *E,CUVNCM (./netlist.vams,64|12): No connection module found:Need an inpu
inv I0 ( .Y(A), .A(IN));
                     |
ncelab: *E,CUVNCM (./netlist.vams,64|20): No connection module found:Need an inpu
inv I2 ( .Y(C), .A(B));
             |
ncelab: *E,CUVNCM (./netlist.vams,68|12): No connection module found:Need an inpu
inv I2 ( .Y(C), .A(B));
                    |
ncelab: *E,CUVNCM (./netlist.vams,68|19): No connection module found:Need an inpu
ncelab: Memory Usage - 25.5M program + 627.3M data = 652.9M total
ncelab: CPU Usage - 0.1s system + 0.1s user = 0.1s total (0.3s, 42.2% cpu)
irun: *E,ELBERR: Error during elaboration (status 1), exiting.
TOOL:       irun        10.20-s045: Exiting on Feb 01, 2012 at 19:45:19 PST  (tota
```

The error and warning messages in the log are from irun and in the context of the AMS simulation flow discussed earlier (compile, elaborate, simulate). The terminology used in the error messages is a little different to what we're used to seeing with an analog simulator, such as Spectre.

ADE provides an ***Error Explanation Tool*** to help with understanding AMS error messages and to locate the source of the error.

5) **Close the simulation log window**

6) From ADE, pick **Simulation → Output Log → Error Explanation**



Name	Value
VDD	1.8

Type	Enab
tran	✓

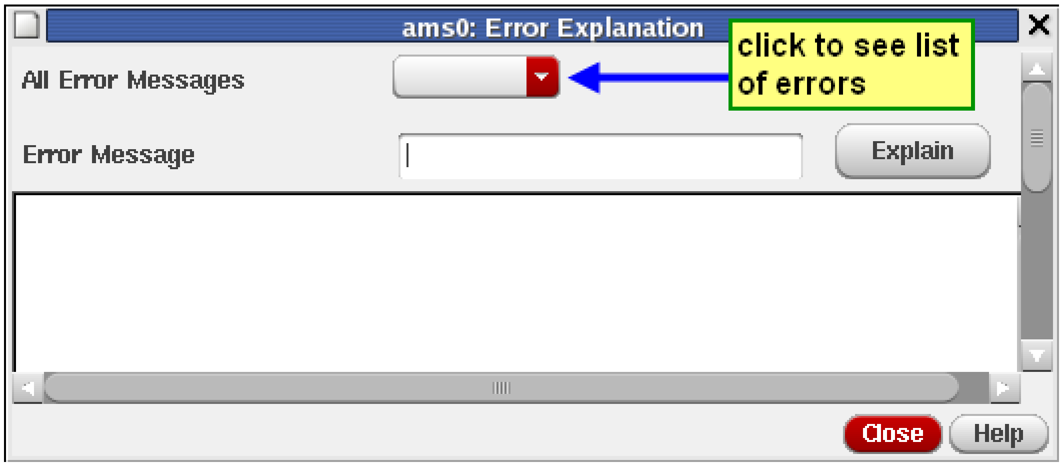
Name/Signal/Expr	Value	Plot	Save
IN		✓	✓
A		✓	✓
B		✓	✓

> Results in ...AMS intro/ams intro oa

mouse L: M:

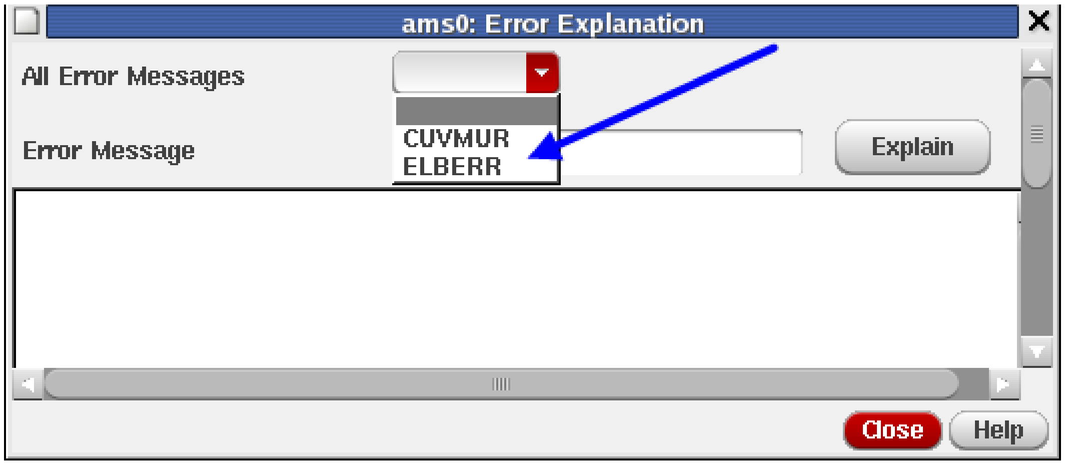
3(5) 3-step Log ... Status: Ready T=27 C Simulator: ams(Spectre) Mode: batch State: ams_state_2

7) The ***Error Explanation*** window will open

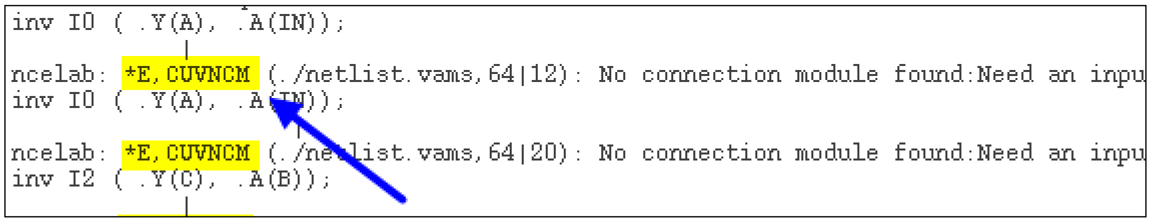


Error messages are available from a drop down list.

8) **Click the drop down list to expand the error messages**.

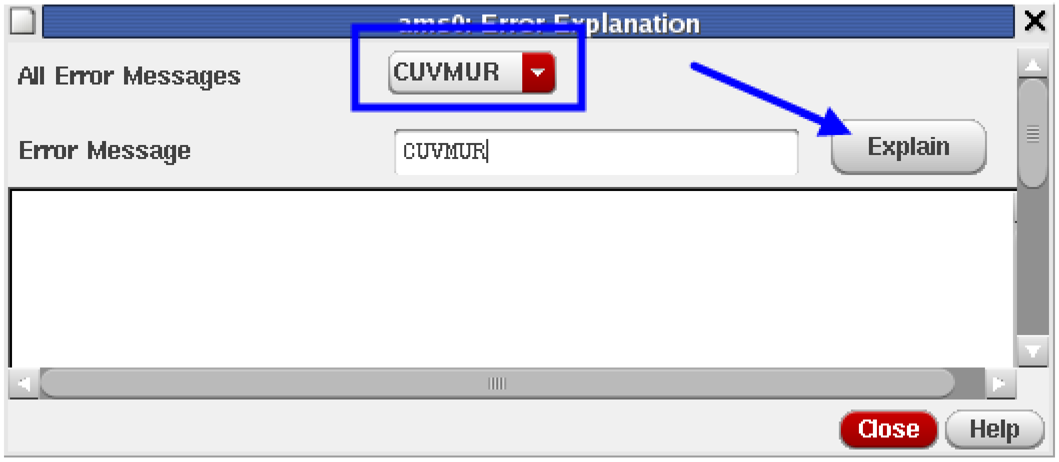


ELBERR is a generic elaboration error and will appear any time elaboration fails. The problem which causes the elaboration to fail is the CUVUMR error.



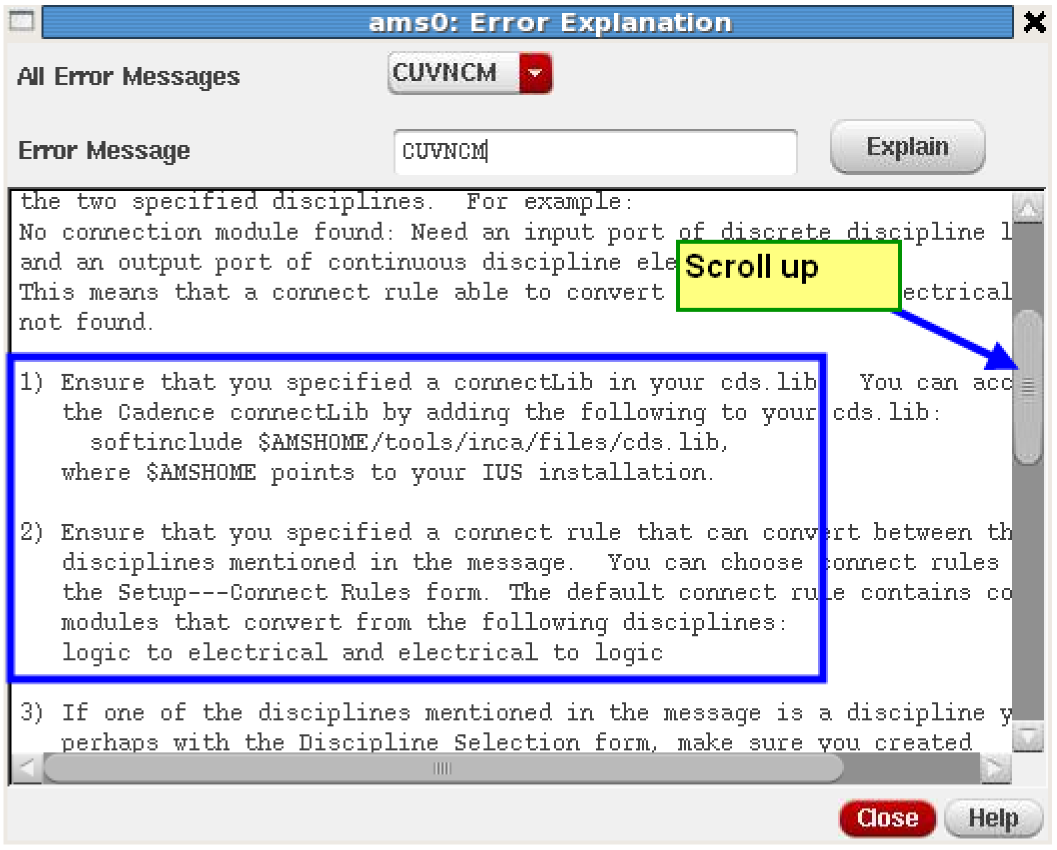
```
inv I0 ( .Y(A), .A(IN));

ncelab: *E,CUVNCM (./netlist.vams,64|12): No connection module found:Need an inpu
inv I0 ( .Y(A), .A(IN));

ncelab: *E,CUVNCM (./netlist.vams,64|20): No connection module found:Need an inpu
inv I2 ( .Y(C), .A(B));
```

9) **Pick the** CUVMUR **error code and click the Explain button**



Possible causes for this error will appear in the window.

10) **Scroll up** to the first item in the list

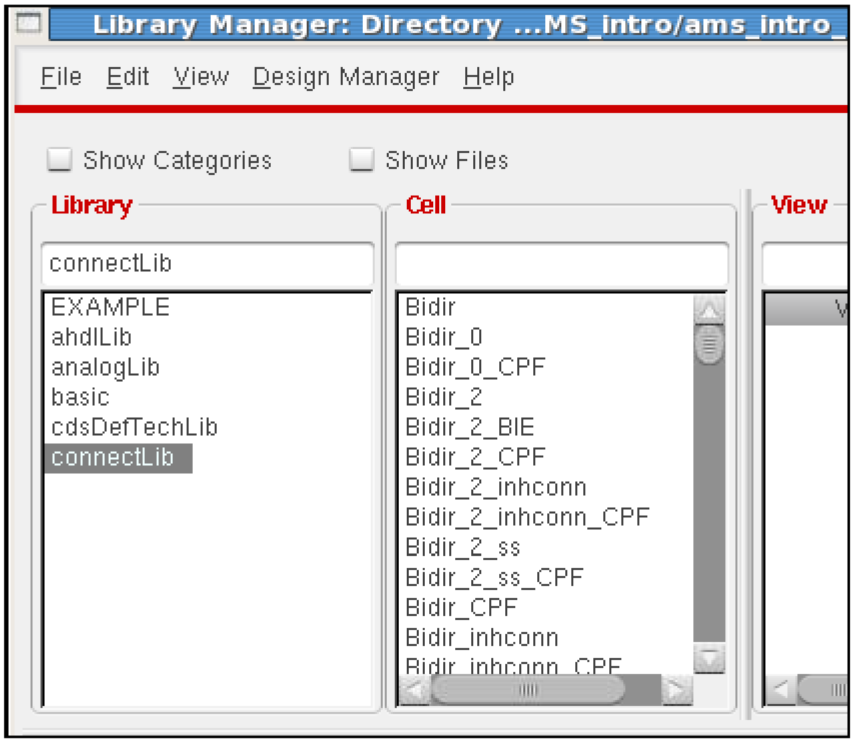


```
the two specified disciplines.  For example:
No connection module found: Need an input port of discrete discipline l
and an output port of continuous discipline ele
This means that a connect rule able to convert                    ectrical
not found.

1) Ensure that you specified a connectLib in your cds.lib.  You can acc
   the Cadence connectLib by adding the following to your cds.lib:
     softinclude $AMSHOME/tools/inca/files/cds.lib,
   where $AMSHOME points to your IUS installation.

2) Ensure that you specified a connect rule that can convert between th
   disciplines mentioned in the message.  You can choose connect rules
   the Setup---Connect Rules form. The default connect rule contains co
   modules that convert from the following disciplines:
   logic to electrical and electrical to logic

3) If one of the disciplines mentioned in the message is a discipline y
   perhaps with the Discipline Selection form, make sure you created
```

The Error Explanation tool provides possible causes for the AMS simulation error(s) that have occurred (scroll up and down to see the list).

The first possibility listed is that the connectLib is not defined in the cds.lib.

11) **Open the Library Manager and verify that the connectLib is visible**

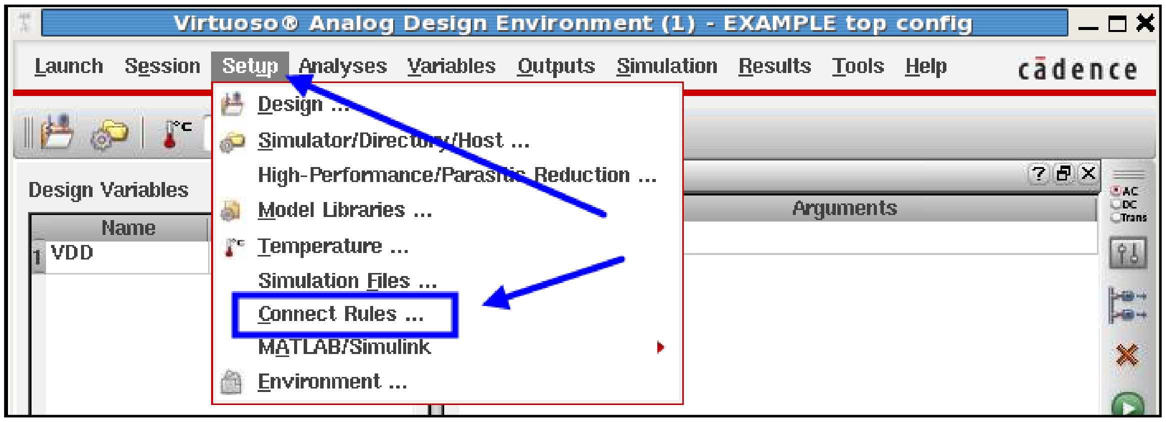


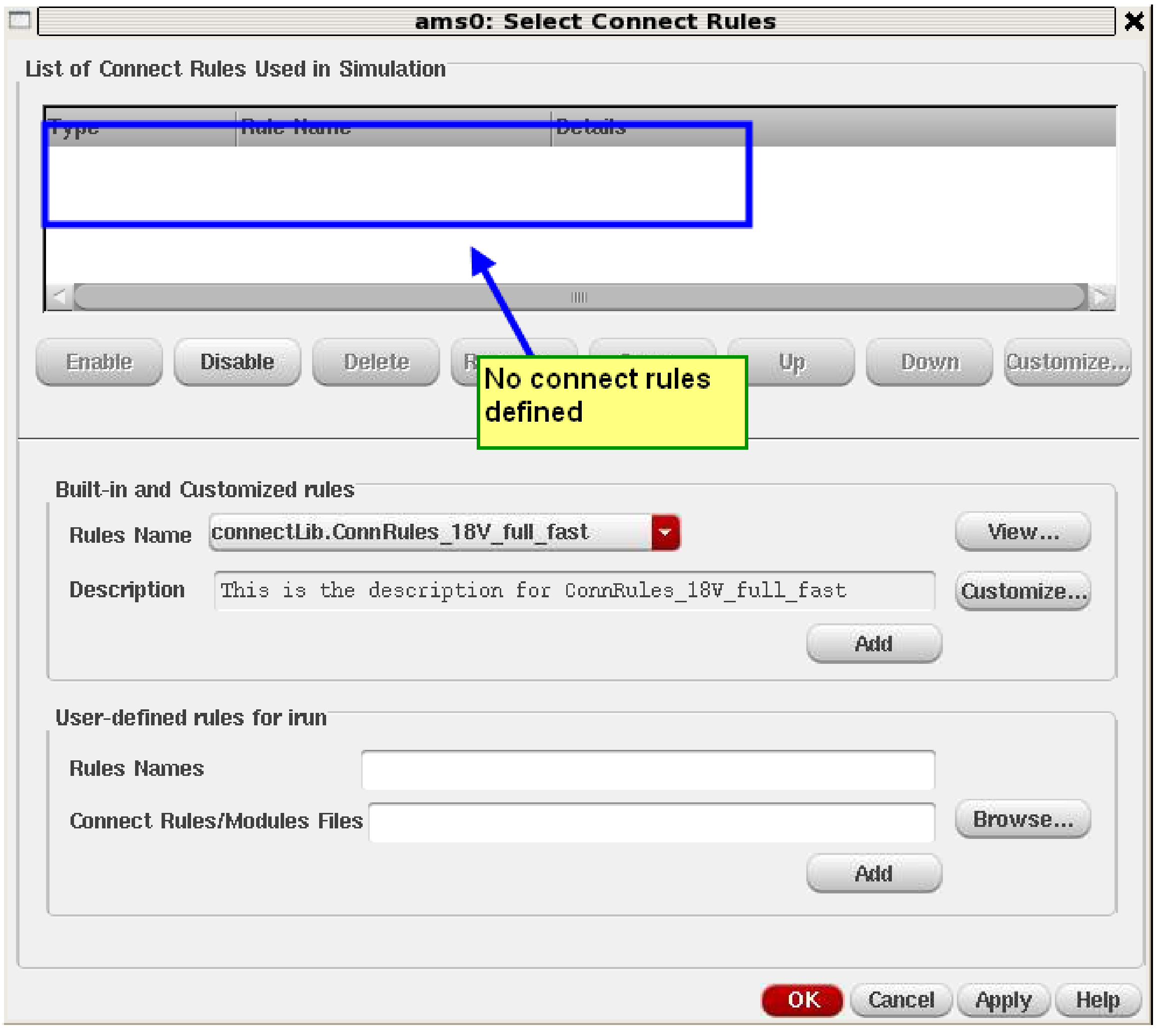
The `connectLib` is visible in the Library Manager.

12) **Item #2 in the Error Explanation is to check that the Connect Rules are defined in the simulation**

```
2) Ensure that you specified a connect rule that can convert between th
   disciplines mentioned in the message.  You can choose connect rules
   the Setup---Connect Rules form. The default connect rule contains co
   modules that convert from the following disciplines:
   logic to electrical and electrical to logic
```

13) From ADE, pick **Setup → Connect Rules**


Virtuoso® Analog Design Environment (1) - EXAMPLE top config
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
Design ...
Simulator/Directory/Host ...
High-Performance/Parasitic Reduction ...
Model Libraries ...
Temperature ...
Simulation Files ...
Connect Rules ...
MATLAB/Simulink
Environment ...
Design Variables
Name
1 VDD
Arguments
AC
DC
Trans



Rules Name: connectLib.ConnRules_18V_full_fast

Description: This is the description for ConnRules_18V_full_fast

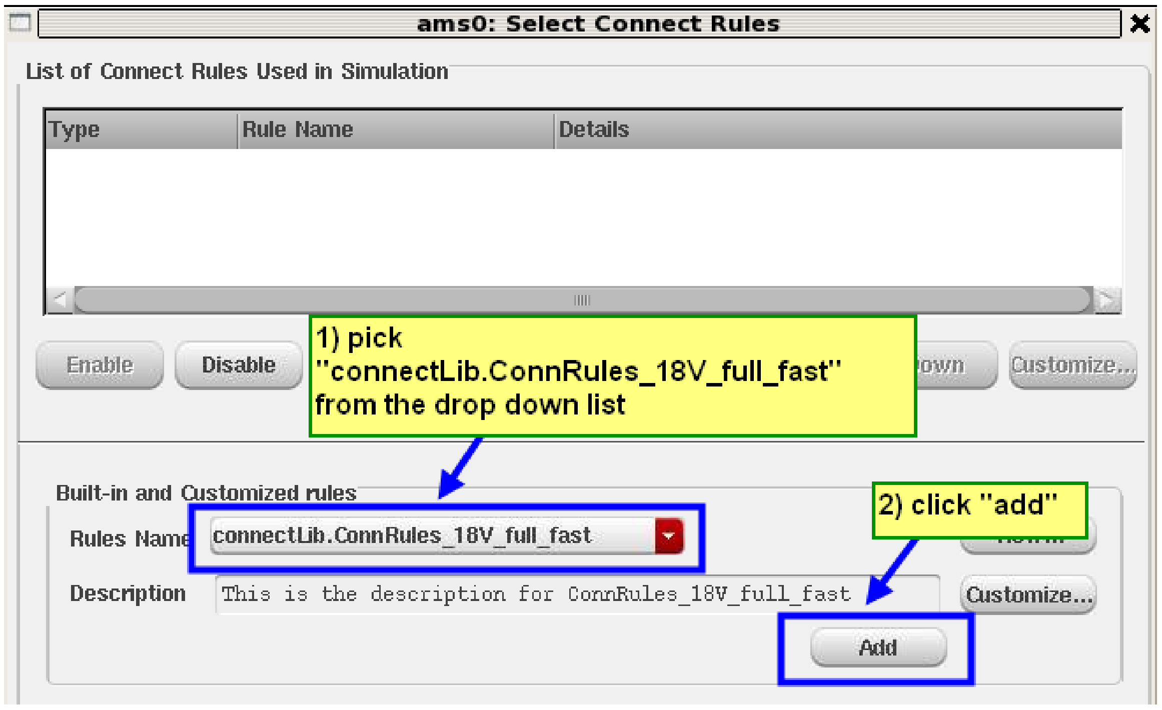
Rules Names:

Connect Rules/Modules Files:

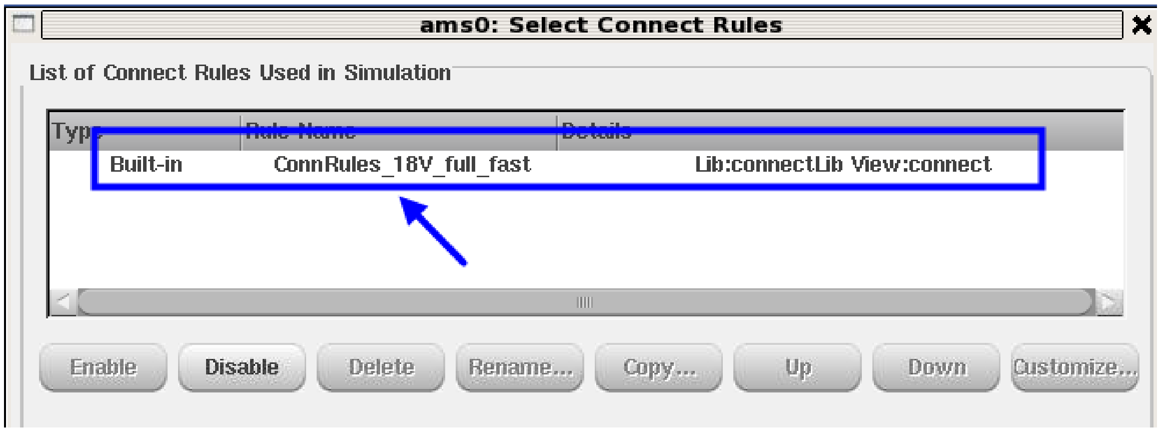
The ADE connect rule setup is empty. There are no connect rules defined in the simulation which is the likely source of the problem.

This simulation uses 1.8V for a logic power supply, therefore we need to setup the 1.8V full fast connect rules.

14) **Pick connectLib.ConnRules_18V_full_fast from the drop down list, and click the "add" button as shown below**


ams0: Select Connect Rules
List of Connect Rules Used in Simulation
Type
Rule Name
Details
Enable
Disable
Down
Customize...
1) pick "connectLib.ConnRules_18V_full_fast" from the drop down list
Built-in and Customized rules
2) click "add"
Rules Name
connectLib.ConnRules_18V_full_fast
Description
This is the description for ConnRules_18V_full_fast
Customize...
Add

This will add the 1.8V full fast rules to the simulation


ams0: Select Connect Rules
List of Connect Rules Used in Simulation
Enable
Disable
Delete
Rename...
Copy...
Up
Down
Customize...

Type	Rule Name	Details
Built-in	ConnRules_18V_full_fast	Lib:connectLib View:connect

15) **Click OK** at the bottom of the Connect Rule setup form.

16) **Close the Error Explanation window.**

17) **Run the simulation again**. This time it completes.

18) **Close ADE, ViVA and the simulation log**

- If prompted to save the ADE simulation state, **pick No**

19) **Close the schematic and the Hierarchy Editor.** Leave the Cadence design framework open for part III.

III. Setting up the config view from scratch

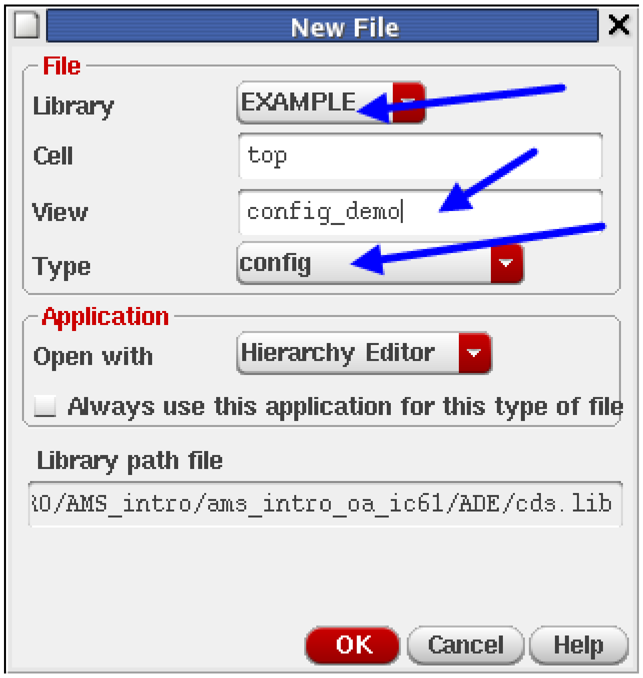
The objective of this section is to setup a config view from scratch that can be used with AMS designer.

1) **From the Library Manager, create a new cell view for this test case**

- From the library manager, pick **File → New → Cell View**

2) **In the new cell view dialog, pick**

- library name: **EXAMPLE**
- cell name: **top**
- view name: **config_demo**
- type: **config**



New File

File

Library: EXAMPLE

Cell: top

View: config_demo

Type: config

Application

Open with: Hierarchy Editor

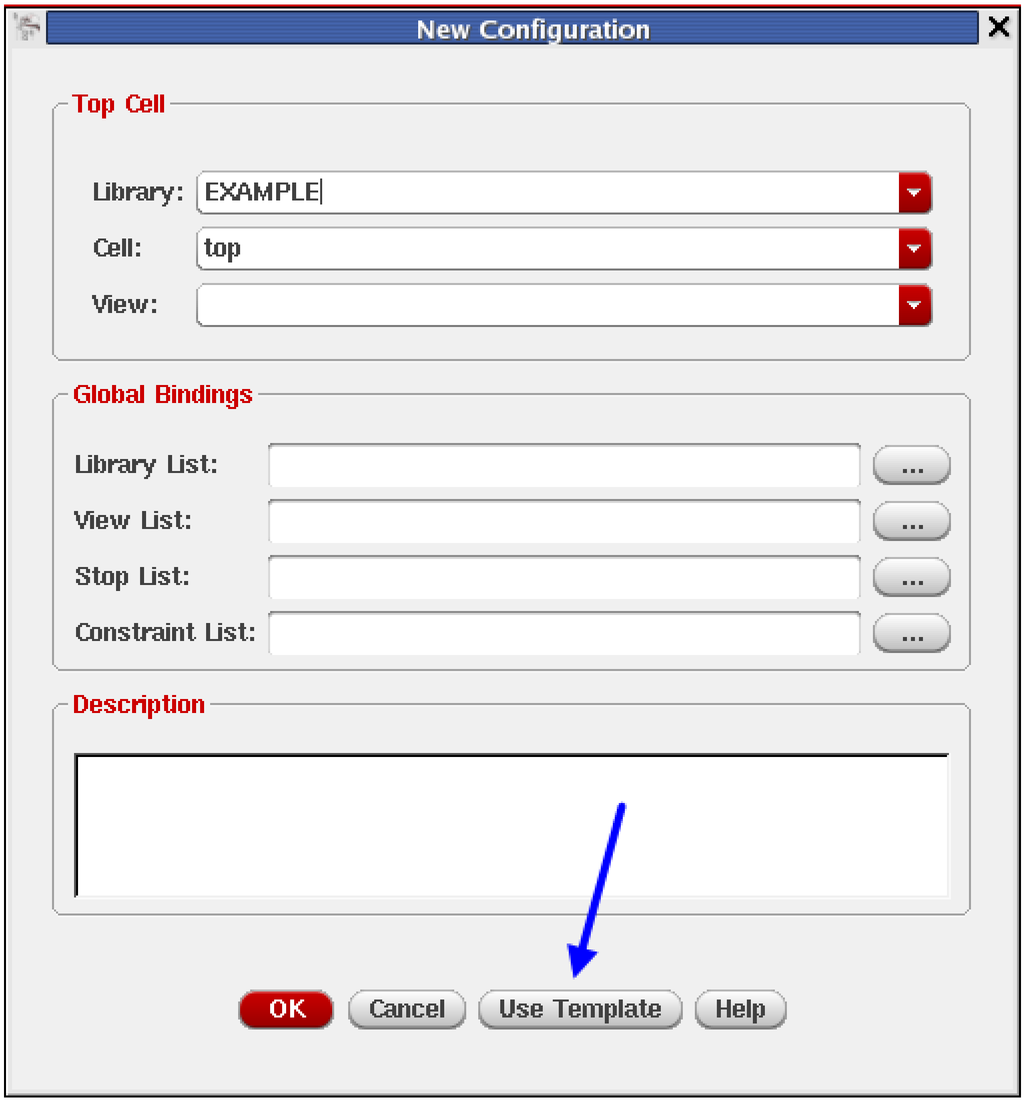
Always use this application for this type of file

Library path file

```
:0/AMS_intro/ams_intro_oa_ic61/ADE/cds.lib
```

3) **Click OK** on the new cell view window

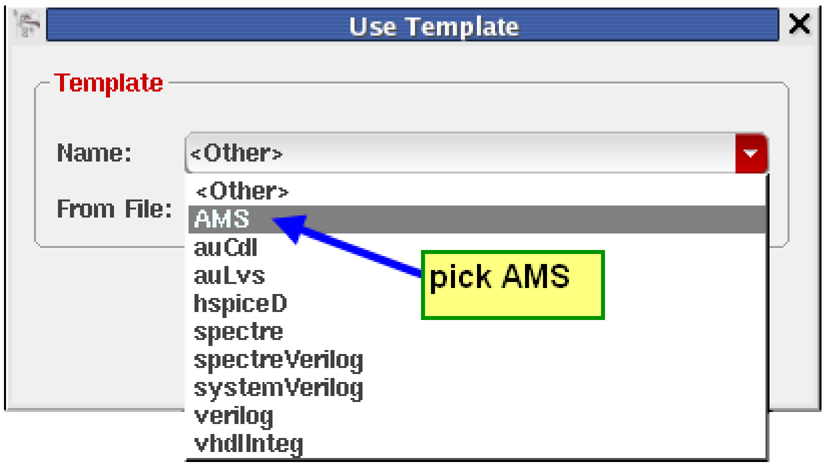
This will open the new configuration setup window



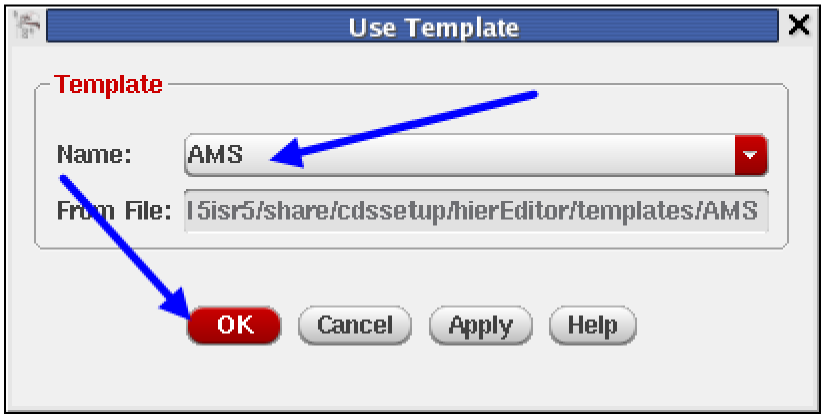
Library: EXAMPLE
Cell: top
View:
Library List:
View List:
Stop List:
Constraint List:

4) Click **Use Template** in the new configuration window as shown above.

5) In the template setup expand the drop down list for "name" and **pick the AMS for the template**.



6) **Click OK** on the template setup window

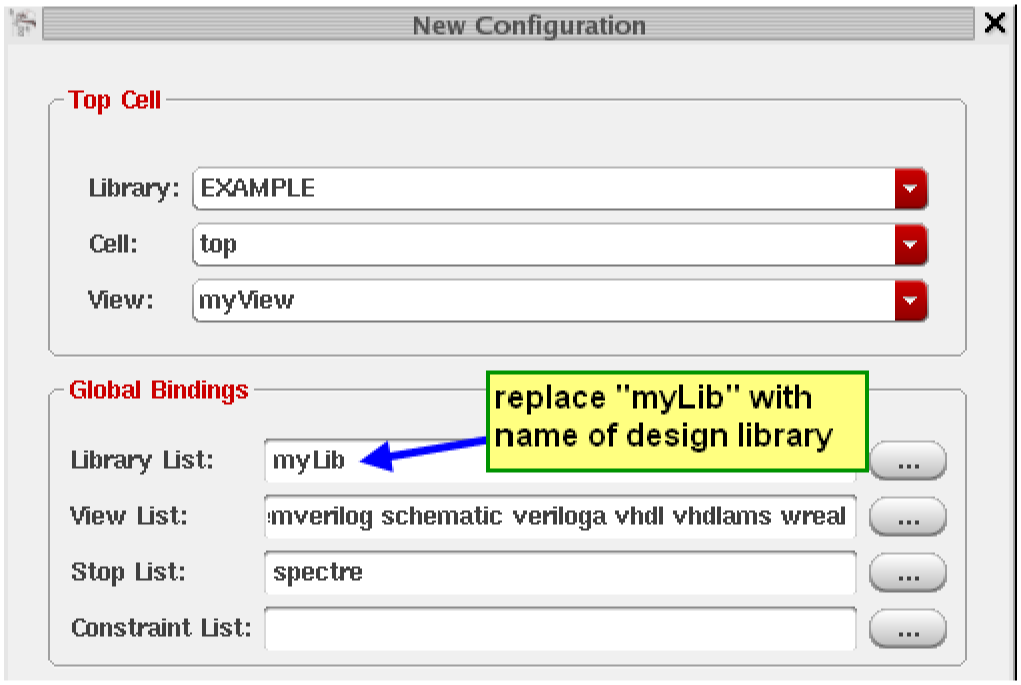


Name: AMS

From File: l5isr5/share/cdssetup/hierEditor/templates/AMS

Using the template sets the default **View List** and **Stop List** for running AMS Designer.

7) **Replace "mylib" with "EXAMPLE" in the Global Bindings section** since that is the design library which the config view will be used.



New Configuration

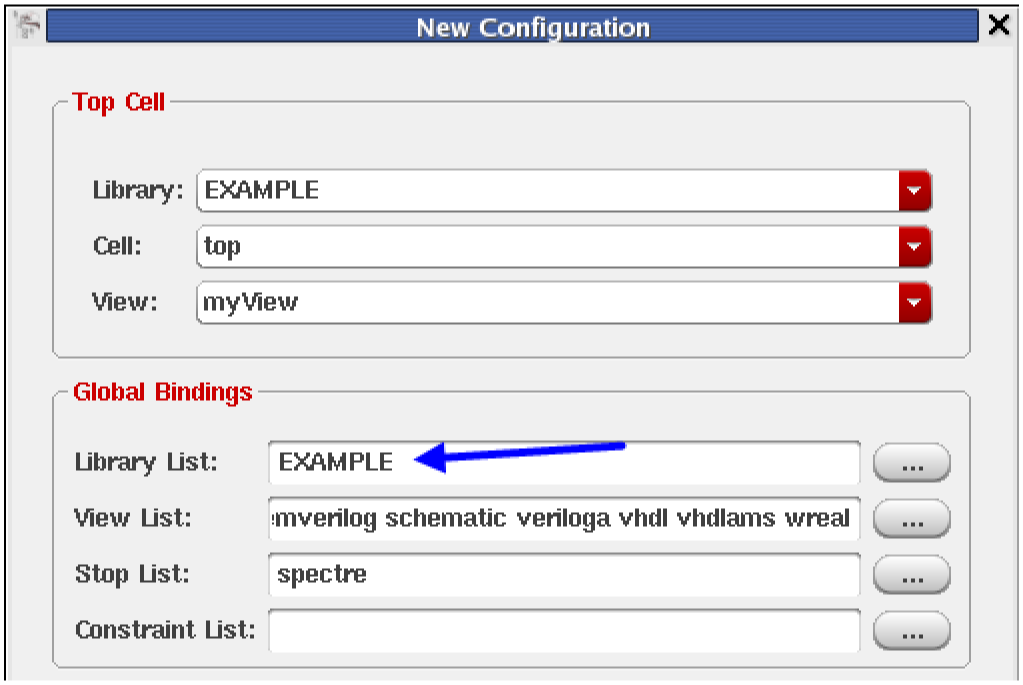
Top Cell

Library: EXAMPLE
Cell: top
View: myView

Global Bindings

Library List: myLib
View List: ımverilog schematic verilog vhdl vhdlams wreal
Stop List: spectre
Constraint List:

replace "myLib" with name of design library



New Configuration

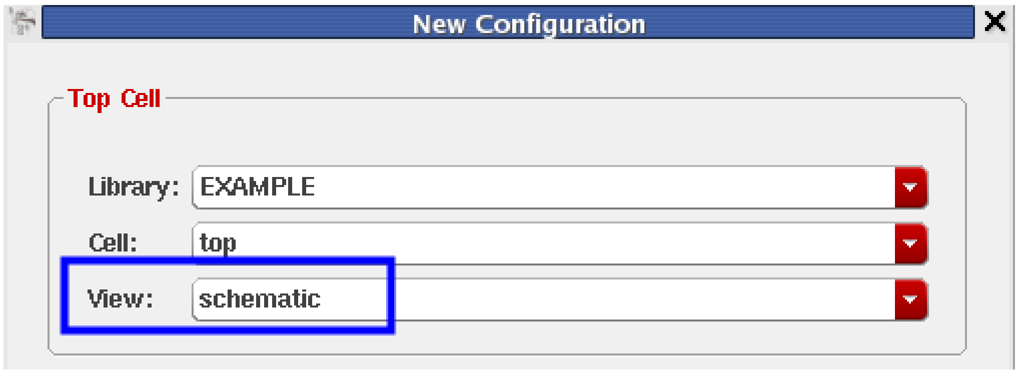
Top Cell

Library: EXAMPLE
Cell: top
View: myView

Global Bindings

Library List: EXAMPLE
View List: ımverilog schematic verilog vhdl vhdlams wreal
Stop List: spectre
Constraint List:

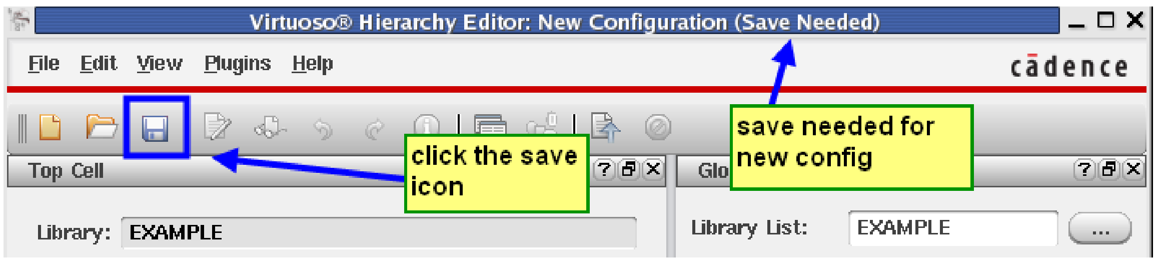
8) Set the **Top Cell** view to be **schematic**. In other words, replace **myView** with **schematic**. This can be done by typing in "schematic" or choosing schematic from the drop down list.



Library: EXAMPLE
Cell: top
View: schematic

9) **Click OK** to close the new config view setup window.

10) **Click the save icon** or pick **File → Save (Needed)** to save the new config


Virtuoso® Hierarchy Editor: New Configuration (Save Needed)
File Edit View Plugins Help
cadence
click the save icon
save needed for new config
Top Cell
Glo
Library: EXAMPLE
Library List: EXAMPLE

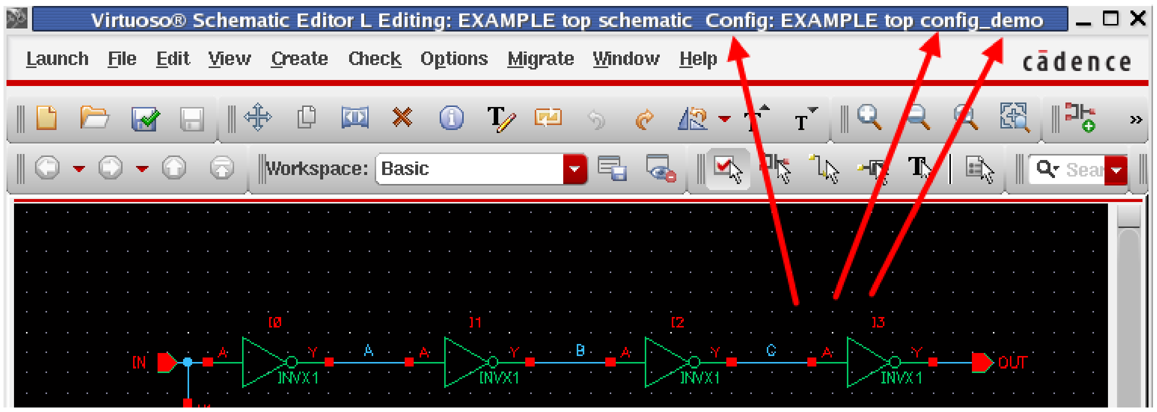
11) This completes the setup for the config view.

12) **Close the Hierarchy Editor**

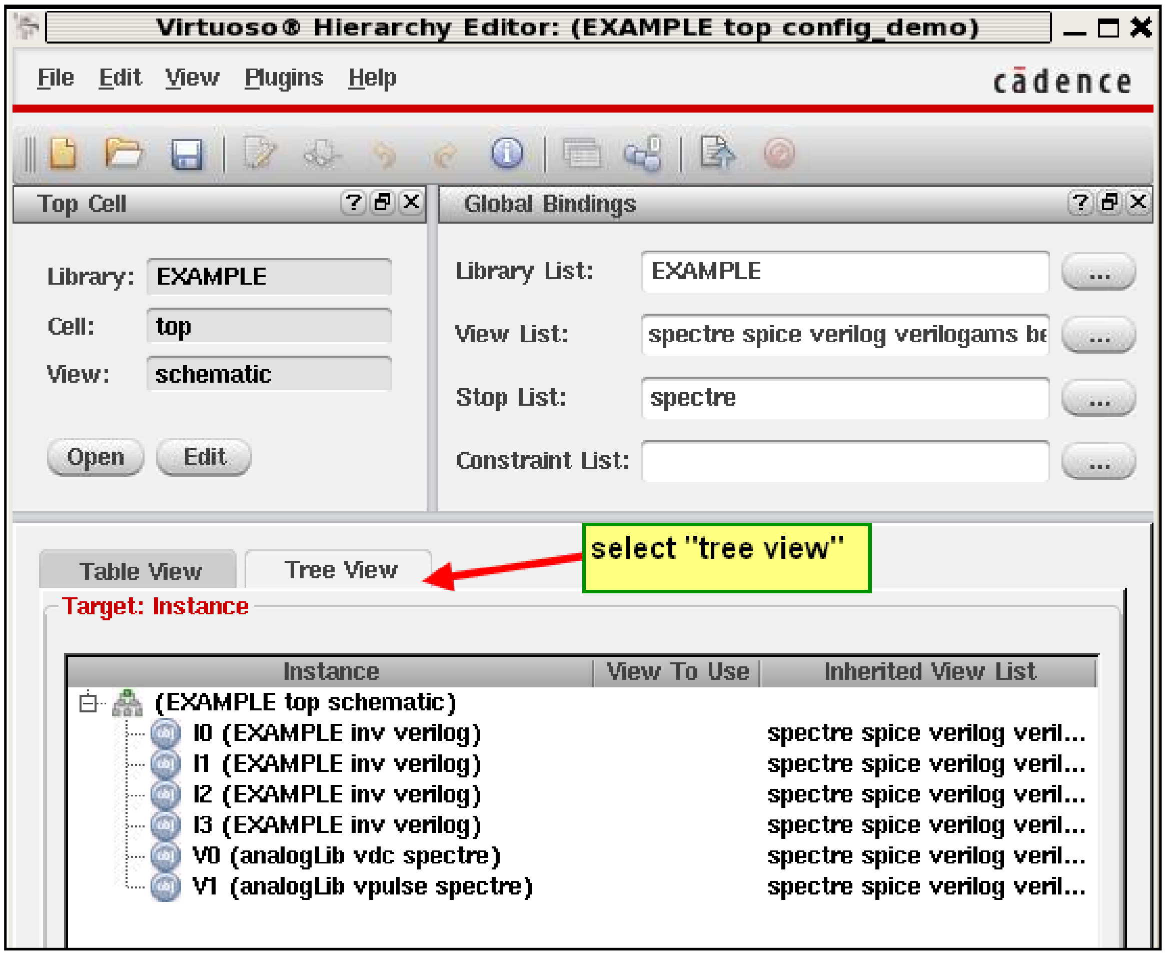
13) **From the library manager open, double click on config_demo,** to open the schematic with the new config_demo configuration.

14) **Choose both the schematic and config when prompted**

15) After opening the schematic and config view, **verify that the schematic is associated with the config_demo.**


Virtuoso® Schematic Editor L Editing: EXAMPLE top schematic Config: EXAMPLE top config_demo
Launch File Edit View Create Check Options Migrate Window Help
cadence
Workspace: Basic
I0
I1
I2
I3
IN
A
Y
INVX1
A
B
C
OUT

16) **In the Hierarchy Editor, pick the tree view tab**



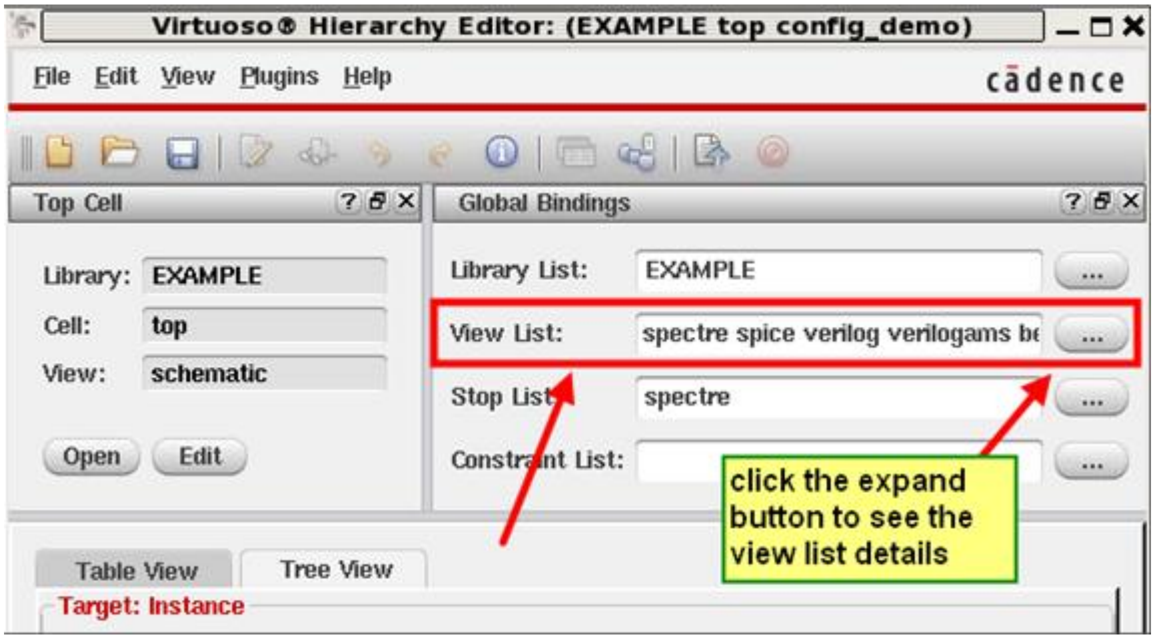
Library: EXAMPLE
Cell: top
View: schematic

Library List: EXAMPLE
View List: spectre spice verilog verilogams be
Stop List: spectre
Constraint List:

Instance	View To Use	Inherited View List
(EXAMPLE top schematic)		
I0 (EXAMPLE inv verilog)		spectre spice verilog veril...
I1 (EXAMPLE inv verilog)		spectre spice verilog veril...
I2 (EXAMPLE inv verilog)		spectre spice verilog veril...
I3 (EXAMPLE inv verilog)		spectre spice verilog veril...
V0 (analogLib vdc spectre)		spectre spice verilog veril...
V1 (analogLib vpulse spectre)		spectre spice verilog veril...

The **AMS template** used to create the config view sets the default configuration by defining the ***view list***.

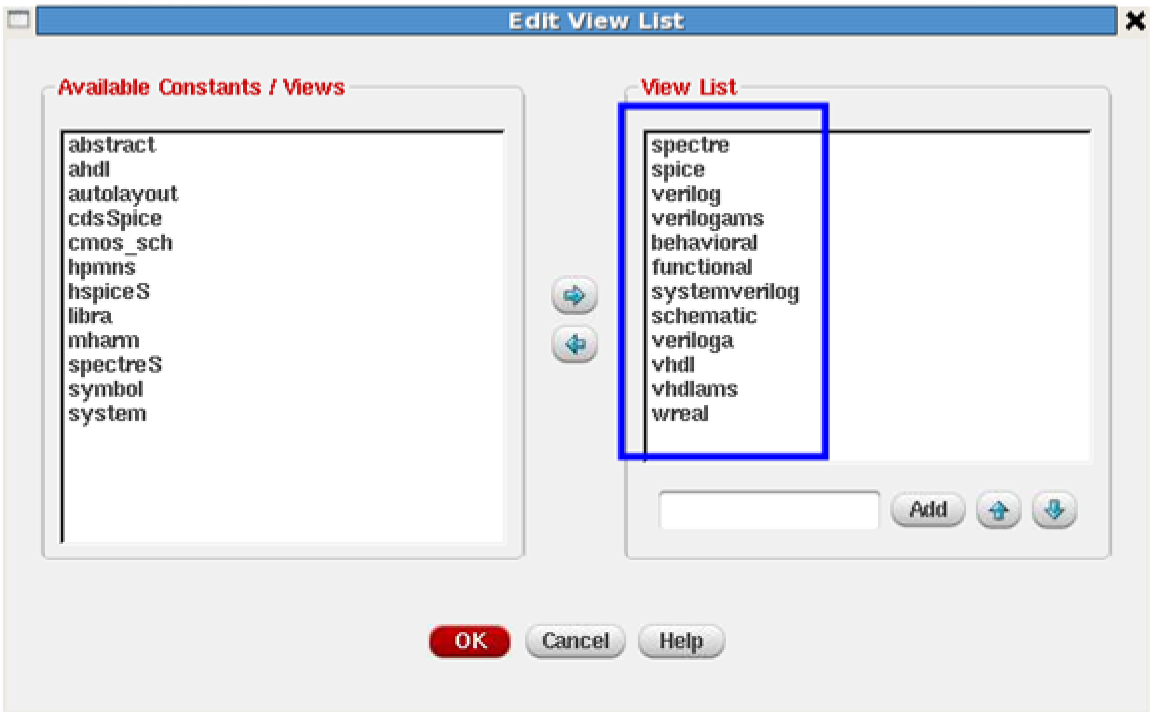
The ***view list*** determines the order of precedence for choosing the cell views which represent each block in the design.


Virtuoso® Hierarchy Editor: (EXAMPLE top config_demo)
File Edit View Plugins Help
cādence
Top Cell
Global Bindings
Open
Edit
Table View
Tree View
Target: Instance
click the expand button to see the view list details

Library: EXAMPLE
Cell: top
View: schematic

Library List: EXAMPLE
View List: spectre spice verilog verilogams be
Stop List: spectre
Constraint List:

17) **Click the "expand button" to the right of the view list to expand the details** as shown above



Available Constants / Views	View List
abstract	spectre
ahdl	spice
autolayout	verilog
cdsSpice	verilogams
cmos_sch	behavioral
hpmns	functional
hspiceS	systemverilog
libra	schematic
mharm	veriloga
spectreS	vhdl
symbol	vhdlams
system	wreal

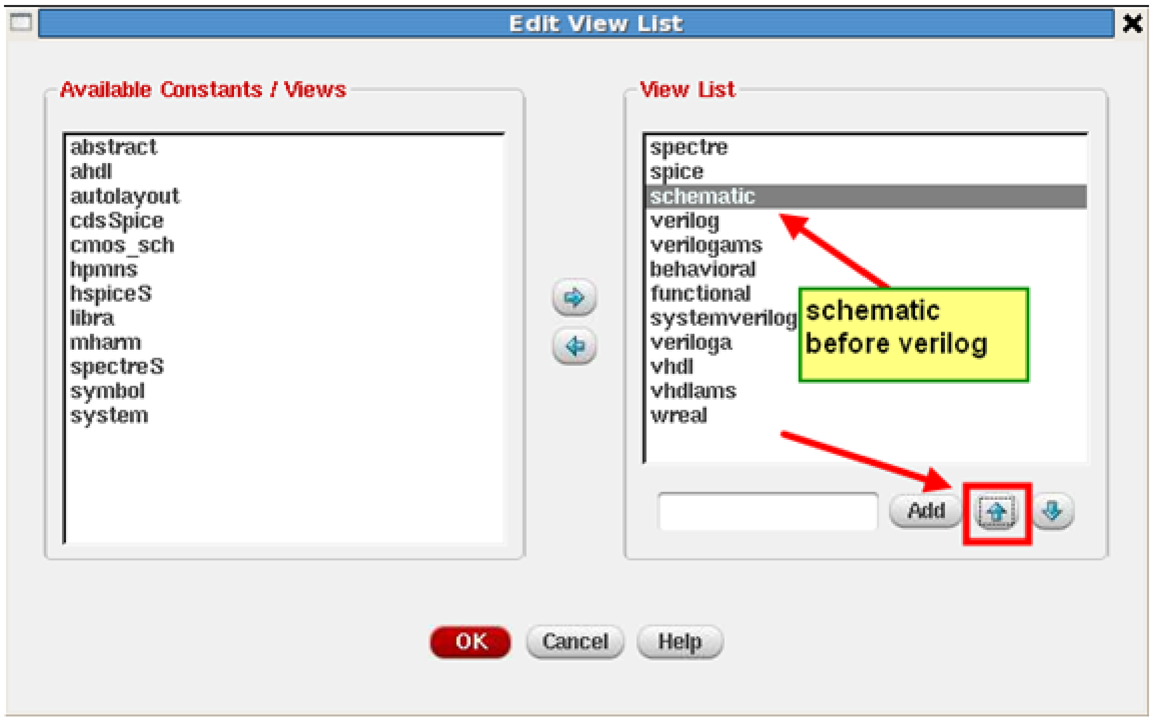
The Hierarchy Editor looks for representations of each design block in the order specified by the view list. **The AMS template places behavioral views <u>before</u> schematic views as shown above**.

Since **verilog** is in the view list before **schematic,** any cell that has both a schematic and verilog view will default to verilog.

In this case, all the blocks in the design have verilog views, and instances I0-I4 will be represented as digital (verilog) in the config.

18) **Change the order of the view** list so that schematic is before **verilog as shown below**.

- Click on "schematic"
- Use the up arrow to change the position of schematic in the list so that it is before “verilog” as shown below.
- Click OK

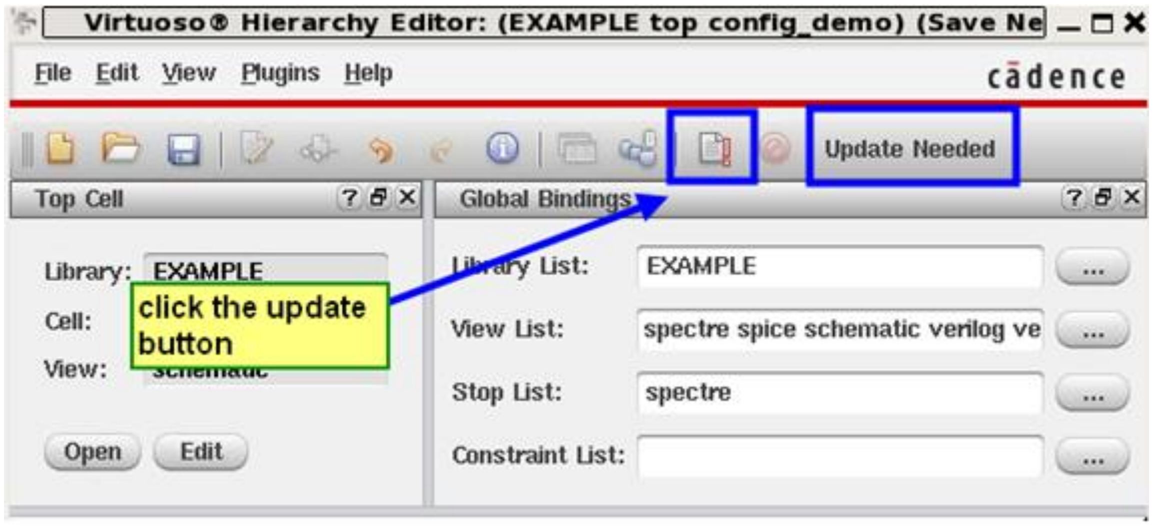


Available Constants / Views:
abstract
ahdl
autolayout
cdsSpice
cmos_sch
hpmns
hspiceS
libra
mharm
spectreS
symbol
system

View List:
spectre
spice
schematic
verilog
verilogams
behavioral
functional
systemverilog
veriloga
vhdl
vhdlams
wreal

19) **Click OK after moving schematic before verilog to update the view list** in the Hierarchy Editor.

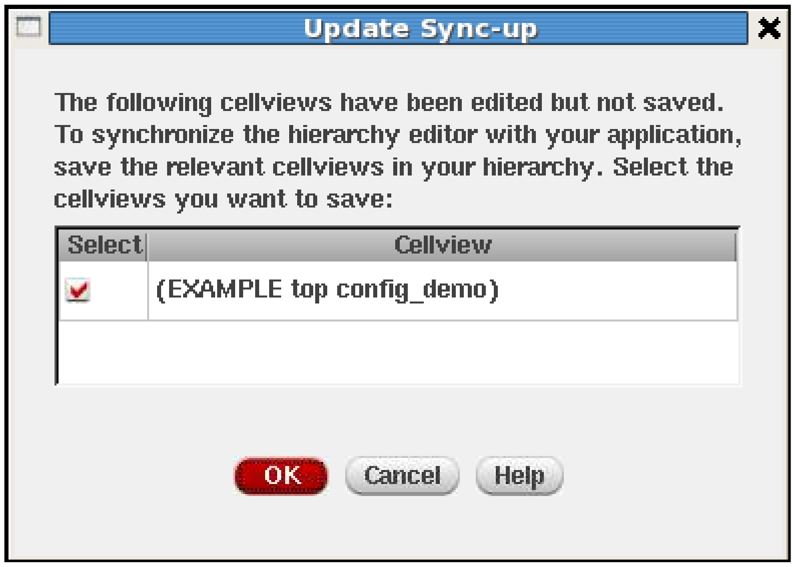
20) To apply the changes to the config, **click the update button.**


Virtuoso® Hierarchy Editor: (EXAMPLE top config_demo) (Save Ne
File Edit View Plugins Help
cādence
Update Needed
Top Cell
Global Bindings
click the update button

Library: EXAMPLE
Cell:
View: schematic

Library List: EXAMPLE
View List: spectre spice schematic verilog ve
Stop List: spectre
Constraint List:

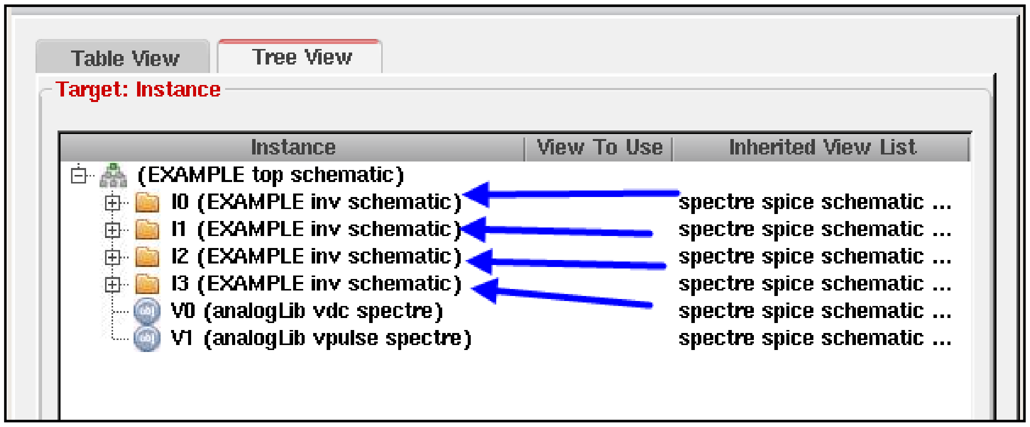
21) **Click OK** on the update confirmation dialog.



The following cellviews have been edited but not saved. To synchronize the hierarchy editor with your application, save the relevant cellviews in your hierarchy. Select the cellviews you want to save:

Select	Cellview
✓	(EXAMPLE top config_demo)

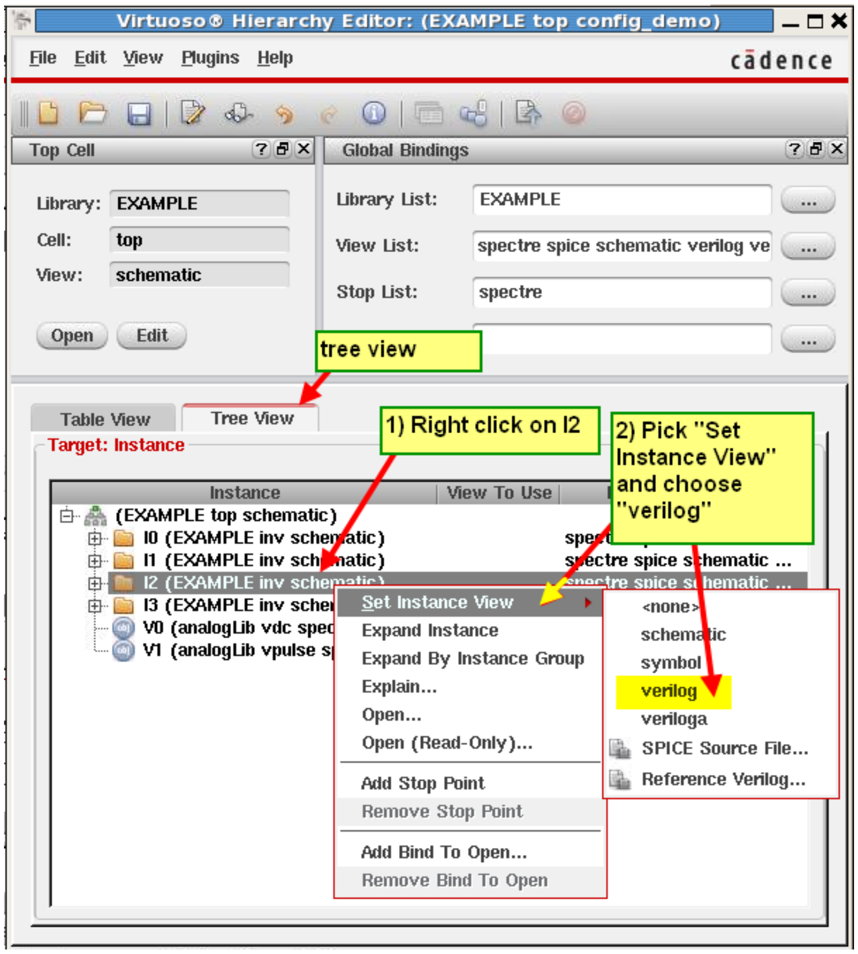
Now I0-I4 will default to schematic views since the inverters have schematic views and "schematic" is in the view list before verilog.



Instance	View To Use	Inherited View List
(EXAMPLE top schematic)		
I0 (EXAMPLE inv schematic)		spectre spice schematic ...
I1 (EXAMPLE inv schematic)		spectre spice schematic ...
I2 (EXAMPLE inv schematic)		spectre spice schematic ...
I3 (EXAMPLE inv schematic)		spectre spice schematic ...
V0 (analogLib vdc spectre)		spectre spice schematic ...
V1 (analogLib vpulse spectre)		spectre spice schematic ...

From this point, individual blocks can be configured for different representations as was done in part 1.

22) For example, **set I2 to verilog** by right clicking on I2 and pick viewing to use as verilog. **Be sure to click the update icon after making the change.**



Library: EXAMPLE
Cell: top
View: schematic

Library List: EXAMPLE
View List: spectre spice schematic verilog ve
Stop List: spectre

Instance	View To Use	
(EXAMPLE top schematic)		
I0 (EXAMPLE inv schematic)		spe
I1 (EXAMPLE inv schematic)		spectre spice schematic ...
I2 (EXAMPLE inv schematic)		spectre spice schematic ...
I3 (EXAMPLE inv sche		
V0 (analogLib vdc spec		
V1 (analogLib vpulse s		

At this point, the steps outlined in Section 1 can be used to setup ADE and run the AMS simulation.

- From the schematic, pick **Launch → ADE L**
- From ADE, pick **Session → Load State**, choose `ams_state1`
- Run the simulation

Please note to duplicate the simulation results from Section 1 (page 32), it will be necessary to also change `I0` and `I1` to Verilog in the config view. The reason that 3 instances, `I0`, `I1` and `I2`, need to be changed to Verilog is that we had changed the view list in the Hierarchy Editor to have schematic before Verilog whereas the simulation in part 1 uses the default view list which gives priority to Verilog views.

23) Close the Cadence Design Framework (CIW: **File → Exit**)

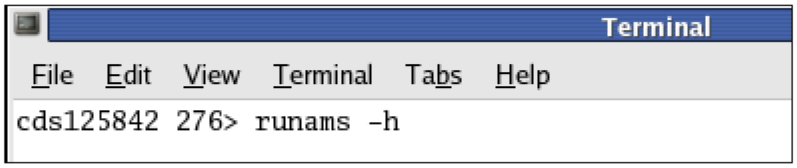
IV. Running from the command line with runams

runams was introduced in 2009 to provide support for netlisting and running an AMS simulation from the command line with the OSS based flow. runams is essentially a command line equivalent to picking “simulate” or "netlist and simulate" from the ADE GUI when the simulation mode set to OSS/irun. An important feature of runams is that runams can generate a netlist from the command line.

One use model is to use the ADE to setup and debug the simulation then move to command line simulation with runams.

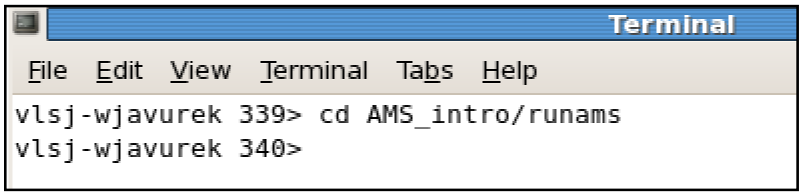
- Once the simulation is running inside ADE, it is very easy to move to a command line flow with runams.
- runams reads the config view to generate the netlist. This makes it convenient to change the representation of the design blocks by creating different config views with the Hierarchy Editor. runams will read the config view and create the appropriate netlist
- runams can use the ADE saved state to setup the command line simulation

To access the online help for runams, use `runams -h`.



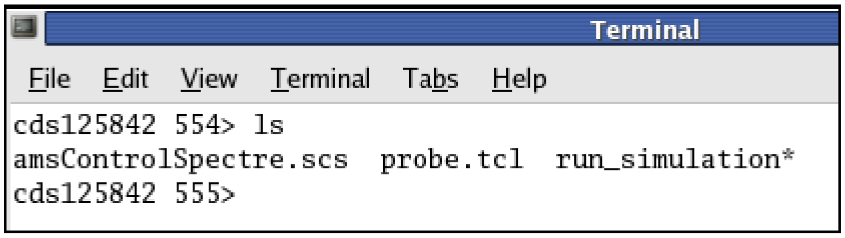
```
cds125842 276> runams -h
```

1) Change directory to AMS_intro/runams



```
vlsj-wjavurek 339> cd AMS_intro/runams
vlsj-wjavurek 340>
```

In this directory, you will see 3 files



```
cds125842 554> ls
amsControlSpectre.scs  probe.tcl  run_simulation*
cds125842 555>
```

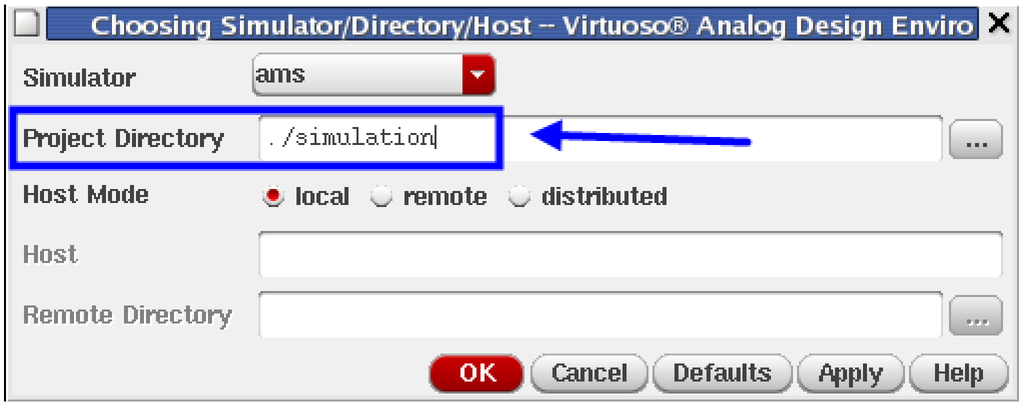
a) `amsControlSpectre.scs` has the analog simulation options for Spectre.

b) `probe.tcl` defines the signals to be saved in the simulation. This file was copied from the ADE simulation run directory (see below).

c) `run_simulation` is the script that runs the simulation

ADE Simulation Run Directory

The ADE simulation run directory is the directory where ADE creates the netlist and runs the simulation. This can be found by choosing **setup → simulator, directory, host** from ADE and looking for the **project directory.**


Choosing Simulator/Directory/Host -- Virtuoso® Analog Design Enviro
...
OK
Cancel
Defaults
Apply
Help

Simulator: ams

Project Directory: ./simulation

Host Mode: [✓ local] remote distributed

Host:

Remote Directory:

The netlist files are found in
`<project_directory>/<cell_name>/<simulator>/<view>/netlist`

For this example, the netlist directory will be
`../ADE/simulation/top/ams/config/netlist`

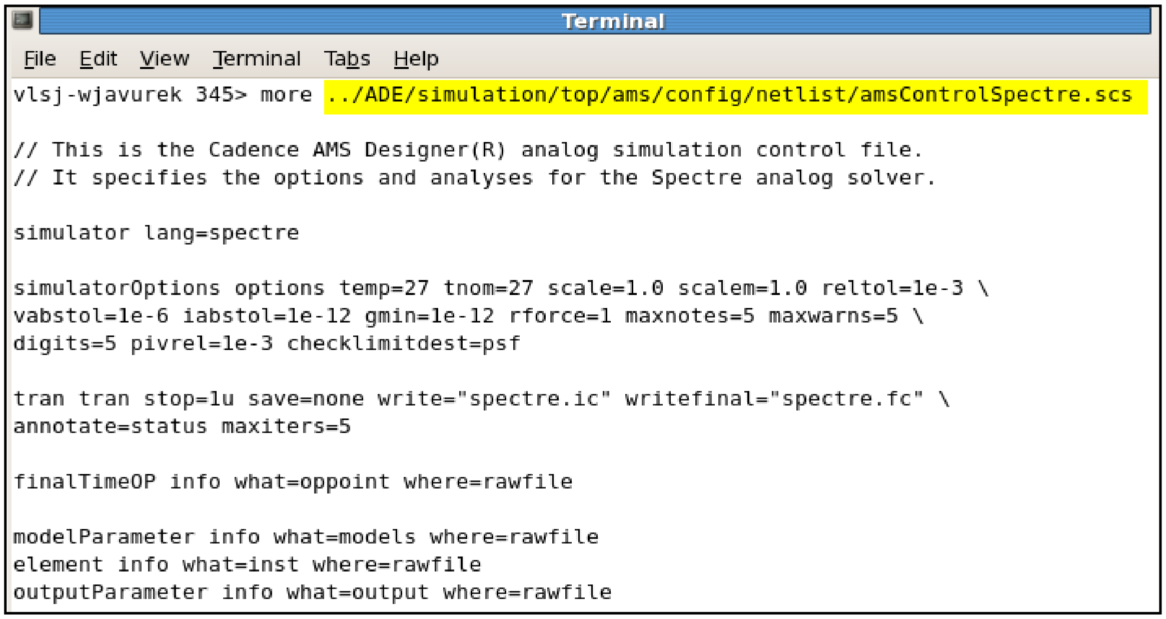
amsControlSpectre.scs

In this file, any options required for the analog portion of the simulation are set. This file can be copied from the ADE simulation run directory or setup from scratch.

The netlist directory for the ADE simulation in part 1 can be found in the following location:

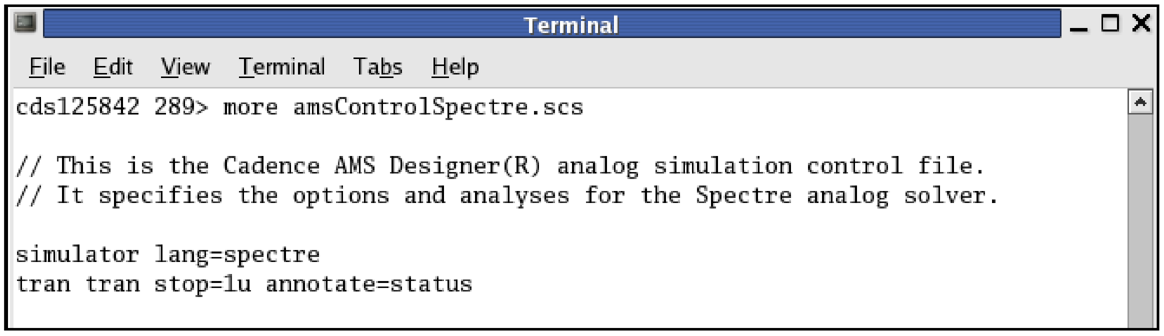
```
../ADE/simulation/top/ams/config/netlist
```

Copying the `amsControlSpectre.scs` file from the ADE simulation run directory will have all the default simulation options and makes a nice template.



```
vlsj-wjavurek 345> more ../ADE/simulation/top/ams/config/netlist/amsControlSpectre.scs

// This is the Cadence AMS Designer(R) analog simulation control file.
// It specifies the options and analyses for the Spectre analog solver.

simulator lang=spectre

simulatorOptions options temp=27 tnom=27 scale=1.0 scalem=1.0 reltol=1e-3 \
vabstol=1e-6 iabstol=1e-12 gmin=1e-12 rforce=1 maxnotes=5 maxwarns=5 \
digits=5 pivrel=1e-3 checklimitdest=psf

tran tran stop=1u save=none write="spectre.ic" writefinal="spectre.fc" \
annotate=status maxiters=5

finalTimeOP info what=oppoint where=rawfile

modelParameter info what=models where=rawfile
element info what=inst where=rawfile
outputParameter info what=output where=rawfile
```

Alternatively, the `amsControlSpectre.scs` file could be created manually. The minimum requirement is to specify the transient simulation stop time

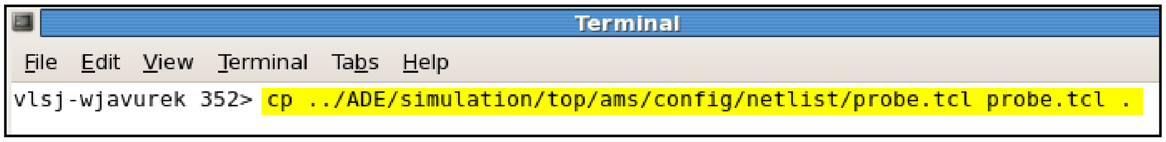


```
cds125842 289> more amsControlSpectre.scs

// This is the Cadence AMS Designer(R) analog simulation control file.
// It specifies the options and analyses for the Spectre analog solver.

simulator lang=spectre
tran tran stop=1u annotate=status
```

2) **Use "more" or a text editor to view** the `amsControlSpectre.scs` file

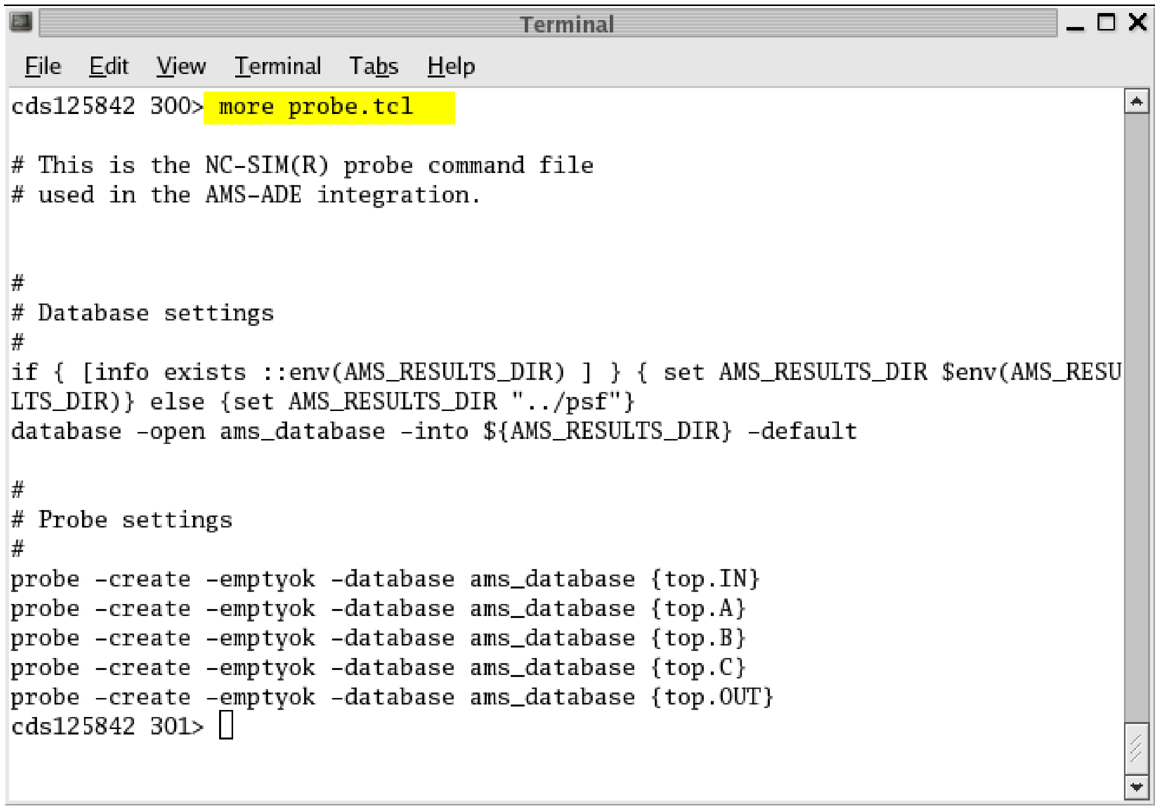
probe.tcl

This file defines the signals to be saved in the AMS simulation. The syntax is ***tcl*** and will be familiar to someone who has used *ncverilog*. The easiest thing to do is to copy this file from the ADE run directory



```
vlsj-wjavurek 352> cp ../ADE/simulation/top/ams/config/netlist/probe.tcl probe.tcl .
```

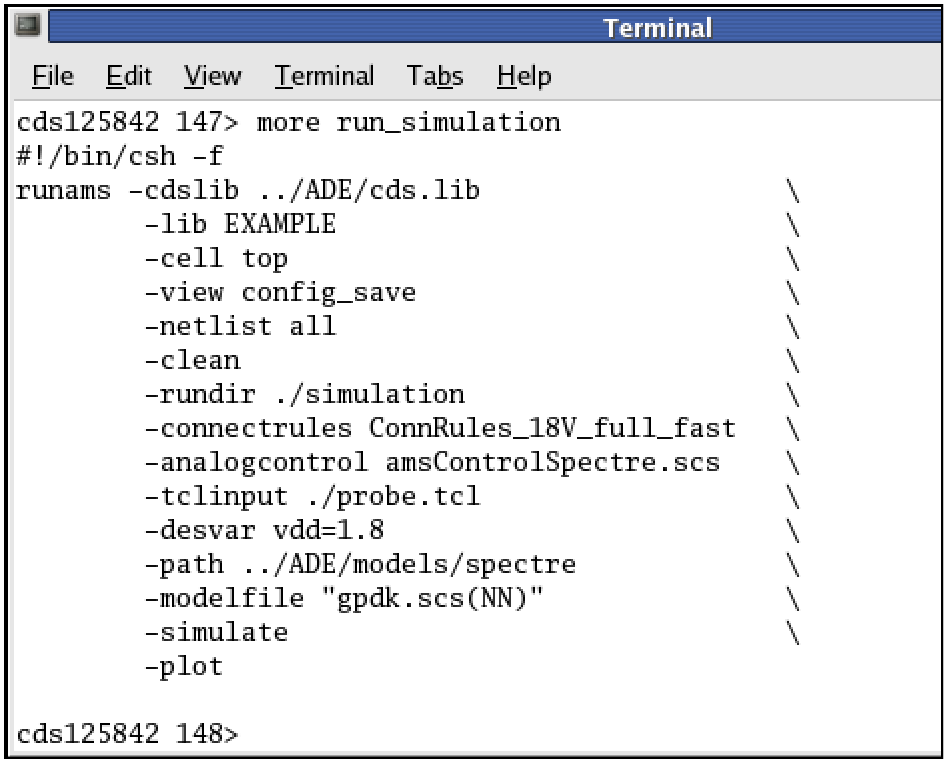
3) **Use "more" or a text editor to view the** probe.tcl **file**



```
cds125842 300> more probe.tcl

# This is the NC-SIM(R) probe command file
# used in the AMS-ADE integration.


#
# Database settings
#
if { [info exists ::env(AMS_RESULTS_DIR) ] } { set AMS_RESULTS_DIR $env(AMS_RESU
LTS_DIR)} else {set AMS_RESULTS_DIR "../psf"}
database -open ams_database -into ${AMS_RESULTS_DIR} -default

#
# Probe settings
#
probe -create -emptyok -database ams_database {top.IN}
probe -create -emptyok -database ams_database {top.A}
probe -create -emptyok -database ams_database {top.B}
probe -create -emptyok -database ams_database {top.C}
probe -create -emptyok -database ams_database {top.OUT}
cds125842 301>
```

run_simulation

This is the script which calls ***runams*** to run the simulation and was created from scratch using the documentation provided with the runams online help (runams –h)



```
cds125842 147> more run_simulation
#!/bin/csh -f
runams -cdslib ../ADE/cds.lib                        \
        -lib EXAMPLE                                 \
        -cell top                                    \
        -view config_save                            \
        -netlist all                                 \
        -clean                                       \
        -rundir ./simulation                         \
        -connectrules ConnRules_18V_full_fast        \
        -analogcontrol amsControlSpectre.scs         \
        -tclinput ./probe.tcl                        \
        -desvar vdd=1.8                              \
        -path ../ADE/models/spectre                  \
        -modelfile "gpdk.scs(NN)"                    \
        -simulate                                    \
        -plot

cds125842 148>
```

4) **Use "more" or a text editor to view the** `run_simulation` **script**

Note that the `runams` command references the `cds.lib` from the ADE simulation in part1 and specifies the library, cell and view to simulation. The `-netlist` option tells `runams` to generate a new netlist when the simulation is run.

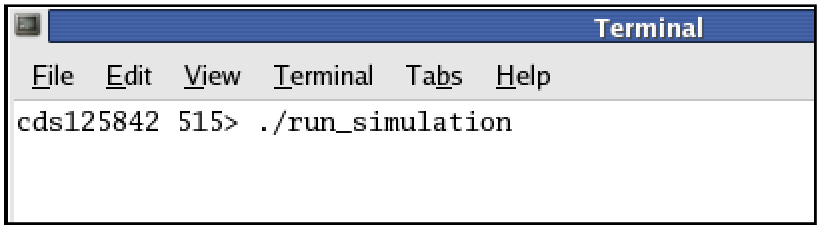
The command line options for `runams` are shown below

-cdslib	points to the cds.lib file so that runams can find the design data
-lib, -cell, -view	defines the lib/cell/view to be simulated
-netlist	tells runams to netlist the full design (same as netlist all in the ADE GUI)
-clean	removes old pak and snapshot files
-rundir	defines the directory where runams will store the netlist and simulation results
-connectrules	sets the connect rules to be used in the simulation
-analogcontrol	control file for the analog solver (described above)

-tclinput	loads in any tcl commands for irun, in this case defines the probes
-desvar	sets any design variables needed
-pat	sets the include path for the model deck
-modelfile	defines the models and section, syntax is model_file(section), double quotes are needed because of the parentheses
-simulate	tell runams to actually run the simulation. Leave this option out if you only want a netlist
-plot	starts the waveform viewer after the simulation finishes (optional)

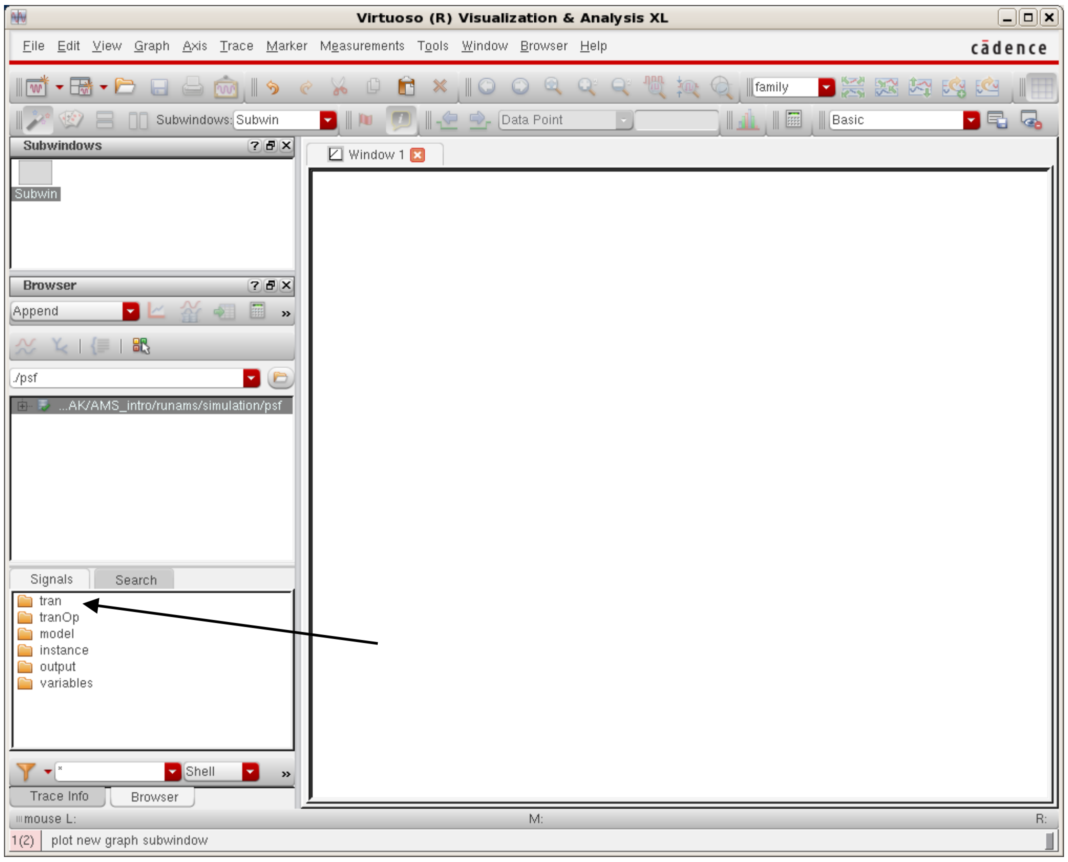
Again, the easiest way to get the syntax for runams is to use "runams –h"

5) **Run the simulation with the** `run_simulation` **script**



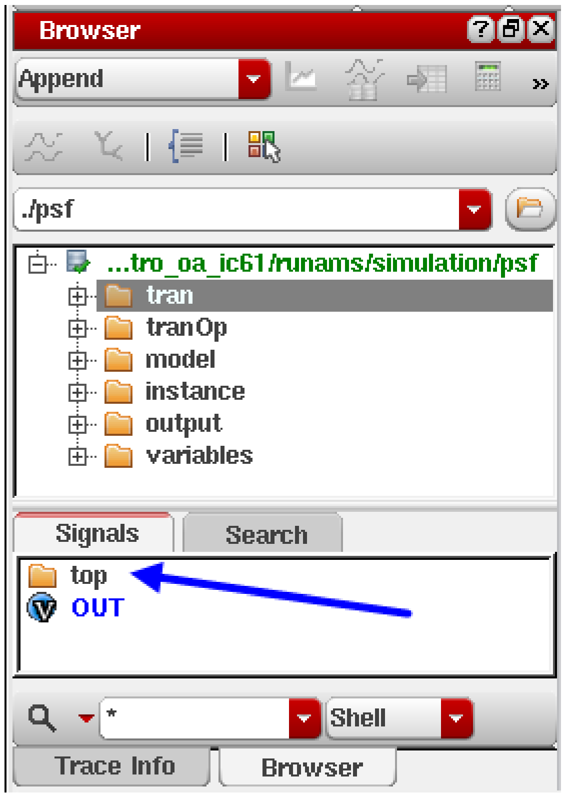
```
cds125842 515> ./run_simulation
```

When the simulation finishes, the standalone version of **ViVA** will open.



6) In the results browser **double click on the tran folder** (shown above)

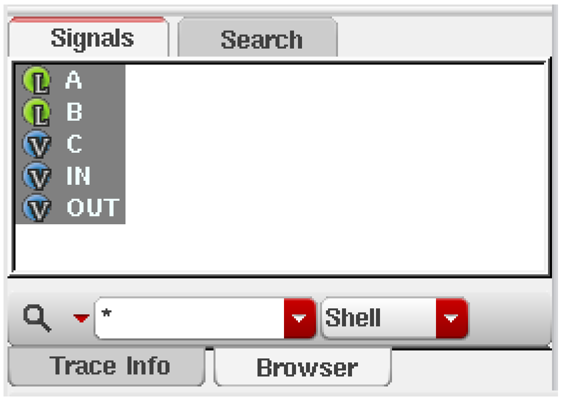
7) **Double click on top** (shown below)



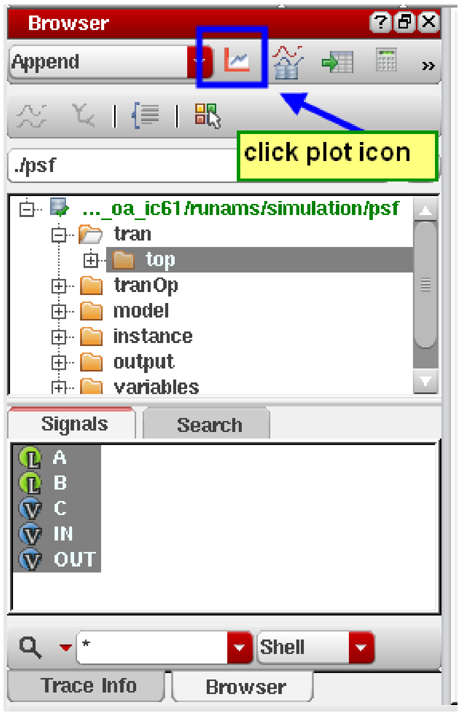
This will point the results browser at the top level signals

8) **Plot all the signals in the top folder**

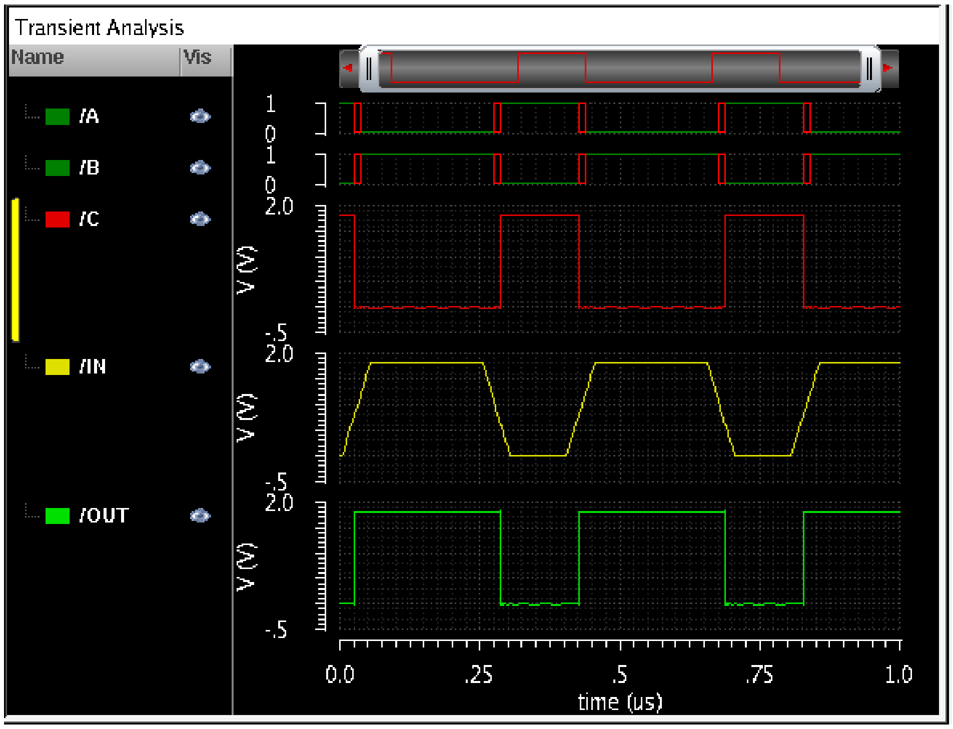
- **Click on the first** signal in the list, **hold down the shift key** and **click on the last**



9) **Click the plot icon** at the top



10) Pick strip mode (**Graph → Split Current Strip**)



11) **Close ViVA** by choosing **File → Exit** from the CIW

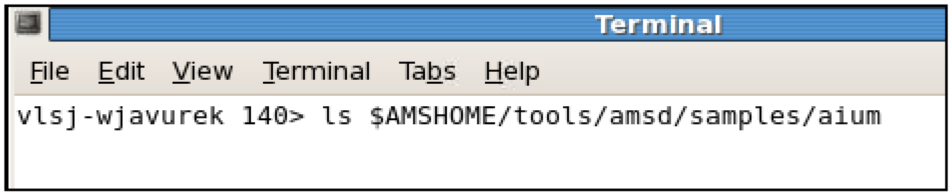
V. Netlist Driven Flow with *irun*

In this section, we will walk through the process of migrating the ADE based simulation to a command line flow with *irun* based on netlists.

For the netlist driven flow, the Spectre (or Spice) netlists and Verilog netlists are saved to text files and the AMS simulation is build up manually. How the simulation is built up depends on what kind of source files (schematics, netlists and so on) are in the simulation. What is contained in this document is a brief example showing one way to go about setting up a netlist based AMS simulation with irun.

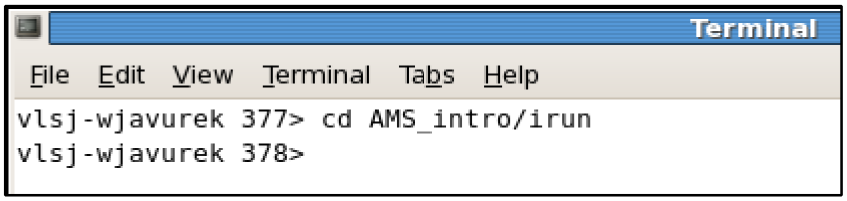
There are more detailed examples on using ***irun*** for command line based AMS simulation in the Cadence Incisive software installation.

`<INCISIV_INSTALL_DIR>/tools/amsd/samples/aium`



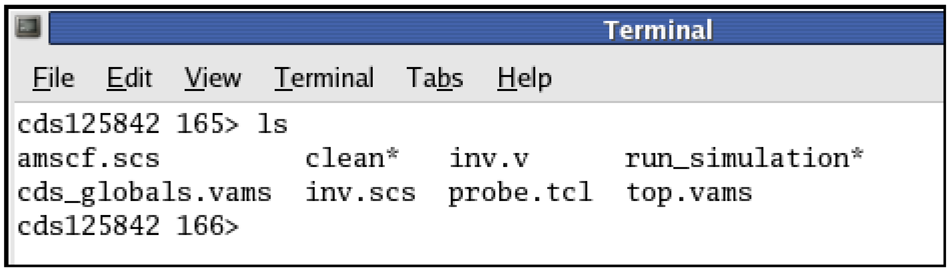
```
vlsj-wjavurek 140> ls $AMSHOME/tools/amsd/samples/aium
```

1) **Change directory to** `AMS_into/irun`



```
vlsj-wjavurek 377> cd AMS_intro/irun
vlsj-wjavurek 378>
```

In this directory, you will see several files



```
cds125842 165> ls
amscf.scs          clean*   inv.v      run_simulation*
cds_globals.vams   inv.scs  probe.tcl  top.vams
cds125842 166>
```

Simulation files used for irun

- ***top.vams*** – top level test bench
- ***inv.scs*** – spectre netlist for inverter
- ***inv.v*** – verilog netlist for inverter
- ***amscf.scs*** – analog control file
- ***cds_globals.vams*** – defines the global signals and design variables
- ***probe.tcl*** – defines the signals to save
- ***clean*** – script to clean the simulation run directory (removes results, logs and intermediate files)
- ***run_simulation*** – script to run the simulation

Let's go through the files one-by-one.

top.vams

This is the top level verilogams test bench for the simulation.

```
// verilogams top level test bench
// Library - EXAMPLE, Cell - top, View - schematic

`include "disciplines.vams"
`timescale 1ns / 1ns

module top (IN, OUT);

output  OUT;
input  IN;

vsource #(.type("pulse"), .val0(0), .val1(cds_globals.VDD),
    .period(400n), .delay(5n), .rise(50n), .fall(50n), .width(200n))
    V1 (IN, cds_globals.\gnd! );

vsource #(.dc(cds_globals.VDD), .type("dc")) V0 (cds_globals.\vdd! ,
    cds_globals.\gnd! );

inv I3 ( .A(C), .Y(OUT));
inv I2 ( .A(B), .Y(C));
inv I1 ( .A(A), .Y(B));
inv I0 ( .A(IN), .Y(A));

endmodule
```

2) **Use "more" or a text editor to view the** `top.vams` **file.**

inv.v

This is the verilog netlist for the inverter and is the same verilog code as in the verilog cell view in the Library Manager.

```
//Verilog HDL for "EXAMPLE", "inv" "verilog"
module inv ( Y, A );
  output Y;
  input A;
  assign Y = ~A;
endmodule
```

3) **Use "more" or a text editor to view the** `inv.v` **file.**

inv.scs

This is the spectre netlist for the inverter from ADE. To create the netlist, open the inverter schematic, launch ADE-L and pick **Simulation → Netlist → Recreate**. Then save the netlist to a file. Note that Virtuoso must be started from the "ADE" directory used in part 1 to generate the inverter netlist.

This file needs some modification so that the schematic representations can be mixed with verilog representations of the inverter. The Spectre netlist from ADE is as follows

```
// Generated for: spectre
// Generated on: Nov 11 09:29:16 2014
// Design library name: EXAMPLE
// Design cell name: inv
// Design view name: schematic
simulator lang=spectre
global 0 vdd!


// Library name: EXAMPLE
// Cell name: inv
// View name: schematic
M1 (Y A 0 0) nmos1 w=1u l=180.00n
M0 (Y A vdd! vdd!) pmos1 w=2u l=180.00n
simulatorOptions options reltol=1e-3 vabstol=1e-6 iabstol=1e-12 temp=27 \
    tnom=27 scalem=1.0 scale=1.0 gmin=1e-12 rforce=1 maxnotes=5
maxwarns=5 \
    digits=5 cols=80 pivrel=1e-3 sensfile="../psf/sens.output" \
    checklimitdest=psf
modelParameter info what=models where=rawfile
element info what=inst where=rawfile
outputParameter info what=output where=rawfile
designParamVals info what=parameters where=rawfile
primitives info what=primitives where=rawfile
subckts info what=subckts  where=rawfile
saveOptions options save=allpub
```

The part needed for the AMS simulation is the netlist for the inverter (**bold green** text above) wrapped in a subcircuit definition as shown below

```
// Library name: EXAMPLE
// Cell name: inv
// View name: schematic

subckt inv A Y
    M1 (Y A 0 0) nmos1 w=1u l=180.00n
    M0 (Y A vdd! vdd!) pmos1 w=2u l=180.00n
ends inv
// End of subcircuit definition.
```

4) **Use “more” or a text editor to view the** `inv.scs` **file.**

Notice the order of the port definitions in the Spectre subcircuit (**bold green** above) are opposite of the verilog netlist. This is okay for this example since the top level netlist (top.vams) uses explicit connections as shown below.

```
// snip from top.vams
inv I3 ( .A(C), .Y(OUT));
inv I2 ( .A(B), .Y(C));
inv I1 ( .A(A), .Y(B));
inv I0 ( .A(IN), .Y(A));
```

If the connections are made by port order (implicit connections), make sure the input and output ports are in the same order on the blocks which you want to swap Verilog and Spice/Spectre netlists.

cds_globals.vams

This one was copied from the ADE simulation done in part 1 and contains the global signal definitions for `vdd!` and `gnd!` along with the `VDD` design variable.

```
`include "disciplines.vams"
// `include "userDisciplines.vams"

module cds_globals;

// Global Signals
   electrical \gnd! ;
   ground \gnd! ;
   wire \vdd! ;

// Design Variables
   dynamicparam real VDD = 1.8;

endmodule
```

The only change here is to comment out the *userDisciplines.vams* line (see **bold green** above). User disciplines are one way of handling multiple power supplies which are not used in this case.

If you want to keep the userDisciplines.vams line, copy the userDisiciplines.vams file from the ADE simulation run directory.

5) **Use "more" or a text editor to view the** `cds_globals.vams` **file.**

Notice in the `cds_globals.vams` is where the `VDD` design variable is defined.

amscf.scs

This is the "ams control file" and has the setup for the AMS simulation. This is where the simulation is configured.

- Define global signals. The first signal in the list is the reference node (node 0)
- Load in the model deck
- Include any spice/spectre netlists that are needed. Nested includes are supported
- The ***amsd block*** configures what is represented as spice (spectre) or verilog and defines any custom connect rule parameters.
- Define the analysis for AMS, in this case, transient

```
//
****************************************************************
// AMS simulation control file for irun netlist based flow
//
****************************************************************
simulator lang=spectre

// global signals.
global 0 vdd!

// model deck
include "../ADE/models/spectre/gpdk.scs" section=NN

// ams configuration options
amsd{

// set the connect rule power supply
  ie vsup=1.8

// mix analog and digital
  config inst="top.I0 top.I1 top.I2" use=hdl
  portmap subckt=inv
  config inst=top.I3 use=spice

}

// run a little bit beyond 1uS so that simvision
// will be active after 1u of transient simulation
tran tran stop=1.1u annotate=status
```

The amsd block is the key element to setting up the ams simulation.

- Define the supply voltage for the connect rules. In this case 1.8V.

- Configure the representation of blocks as analog or digital.
- A portmap needs to be specified for the analog blocks (use=spice) so that AMS will know how to connect the ports.

The ***Virtuoso AMS Designer Simulator User Guide, Chapter 5: Using an amsd Block*** describes the options for the `amsd block` and how to use them.

6) **Use "more" or a text editor to view the `amscf.scs` file.**

probe.tcl

The tcl script to setup the simulation probes. The syntax is the same as that used for verilog simulation.

```
# define the database for the waveforms
database -open waves -into waves.shm -default

#   probe all
probe -create -database waves -all -depth all

#   probe specific signals
# probe -create -emptyok -database ams_database {top.IN}
# probe -create -emptyok -database ams_database {top.A}
# probe -create -emptyok -database ams_database {top.B}
# probe -create -emptyok -database ams_database {top.C}
# probe -create -emptyok -database ams_database {top.OUT}

# run simulation
run 1us
```

The probe.tcl can be copied from the ADE simulation run directory in part 1 for a starting point. Be sure to update the database definition (bold red text above).

The last line in the probe.tcl file runs the simulation up to 1us. The transient stop time in the analog control file is set to 1.1us so that the simulation will not be completed at 1us and the signals can be probed interactively within Simvision.

7) **Use "more" or a text editor to view the `probe.tcl` file.**

clean

Script to clean out the simulation run directory. Deletes the simulation results, log files and intermediate files. This file was created from scratch.

8) **Use "more" or a text editor to view the `clean` script.**

run_simulation

This is a script to run the AMS simulation with **irun**.

```
#!/bin/csh -f

irun                           \
     top.vams                  \
     cds_globals.vams          \
     inv.v                     \
     inv.scs                   \
     amscf.scs                 \
     -timescale 1ns/1ns        \
     -iereport                 \
     -access +rwc              \
     -input probe.tcl          \
     -gui
```

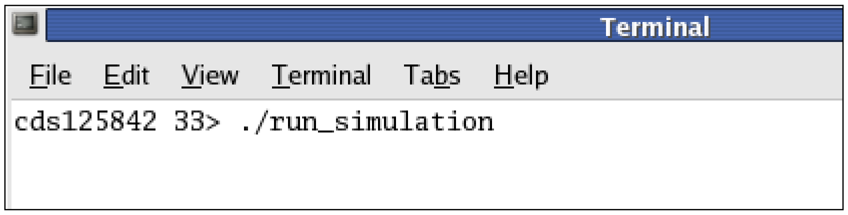
This script runs the AMS simulation

- Reads in the verilog and verilogAMS netlists. Wild cards are supported and you could use "*.v*" in place of the actual file names. Care needs to be taken with wild cards since the wild cards will pick up all matching files which may not be what you want. For example, you might have a simulation using counter.v and have a counter_old.v file in the run directory. The wild card *.v* will pick up both resulting in a duplicate definition for the counter and cause the simulation to fail.
- The amscf.scs is the analog control file for setting up the AMS simulation
- The timescale option sets the default timescale
- -iereport tells irun to report the connect modules (d2a/a2d interface elements) that are inserted during the elaboration phase
- -access provides read access to the simulation database for simvision (same use as with Verilog simulation.
- -input probe.tcl reads in the probe definitions
- The –gui option starts simvision.

Use **irun –helpall** for help with the command line options

9) **Use "more" or a text editor to view the `run_simulation` script.**

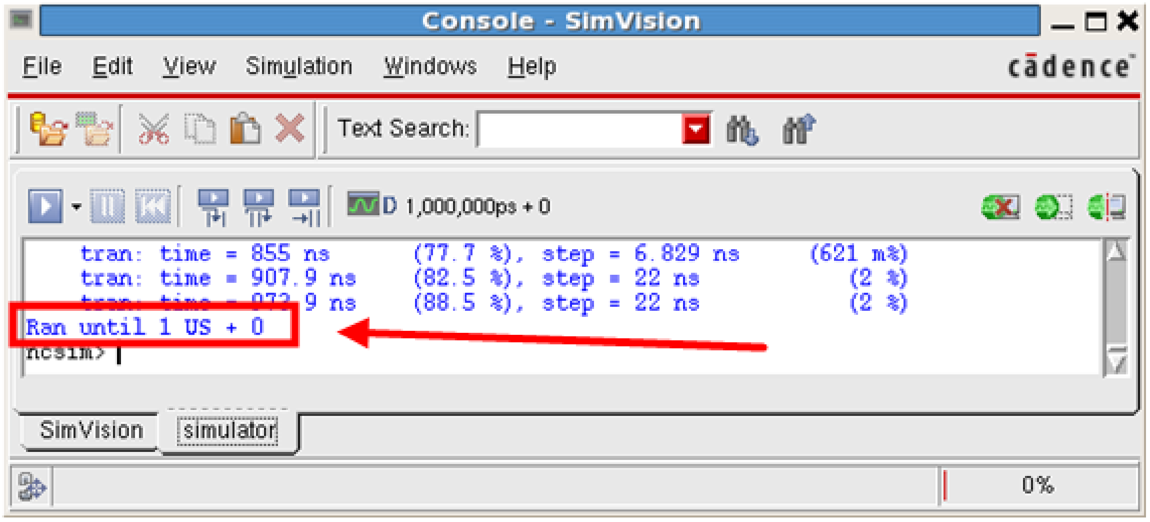
10) **Run the simulation using the `run_simulation` script**



```
cds125842 33> ./run_simulation
```

The script will start Simvision and the simulation will run out to 1us

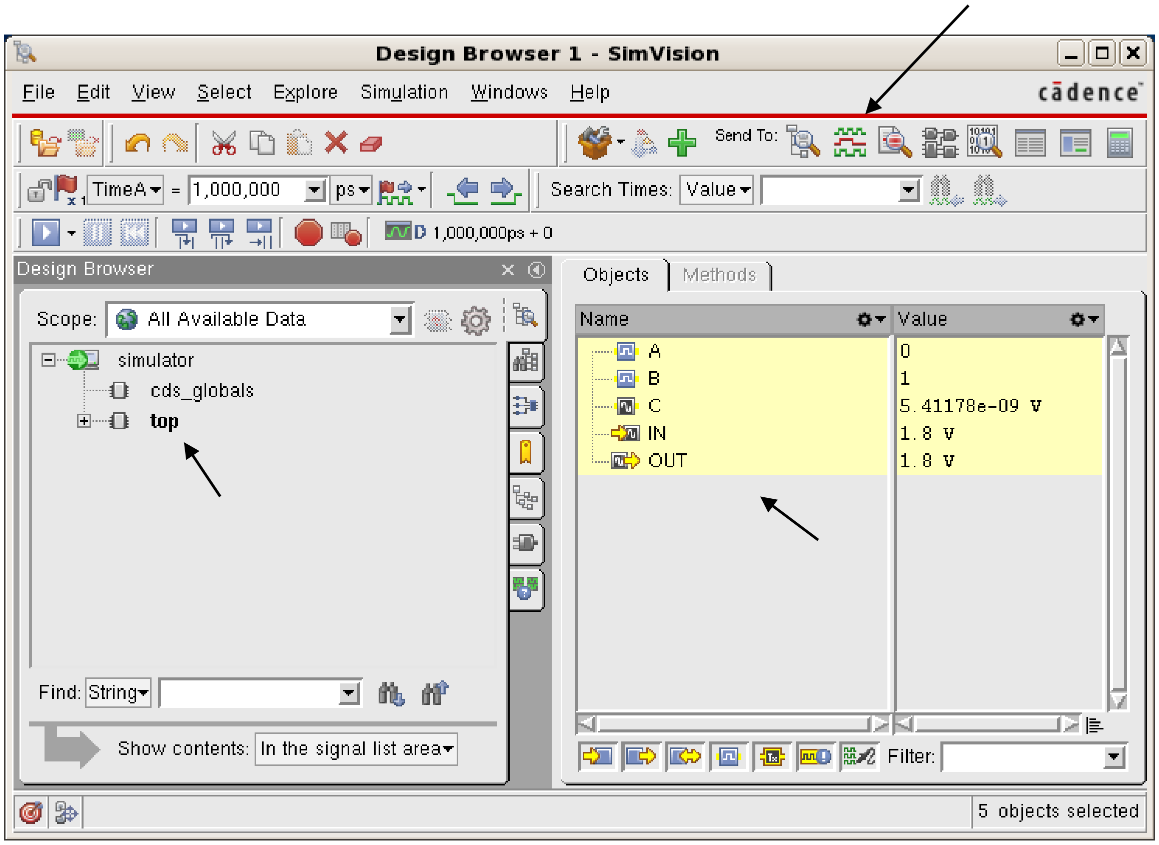
11) **Notice in the Simvision Console window that the simulation has completed to 1us which is the time specified in the `probe.tcl` file**



```
    tran: time = 855 ns       (77.7 %), step = 6.829 ns      (621 m%)
    tran: time = 907.9 ns     (82.5 %), step = 22 ns            (2 %)
    tran: time = 973.9 ns     (88.5 %), step = 22 ns            (2 %)
Ran until 1 US + 0
ncsim>
```

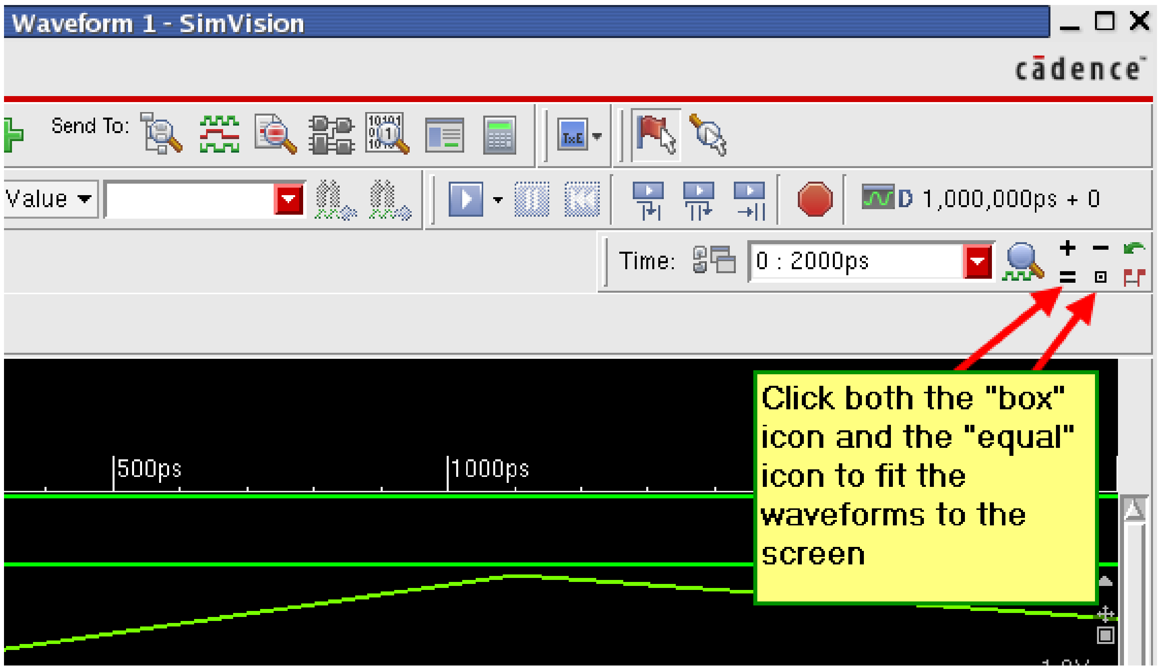
12) Go to the **Simvsion Design Browser** and do the following steps:

a) Click on “top” on Left pane
b) Click on “A” in Right pane. Hold shift key down and left click on “OUT”. All signals should be selected.
c) To plot, click on plot icon. Alternatively, since we are plotting all signals, you could have clicked on the plot icon after step a.



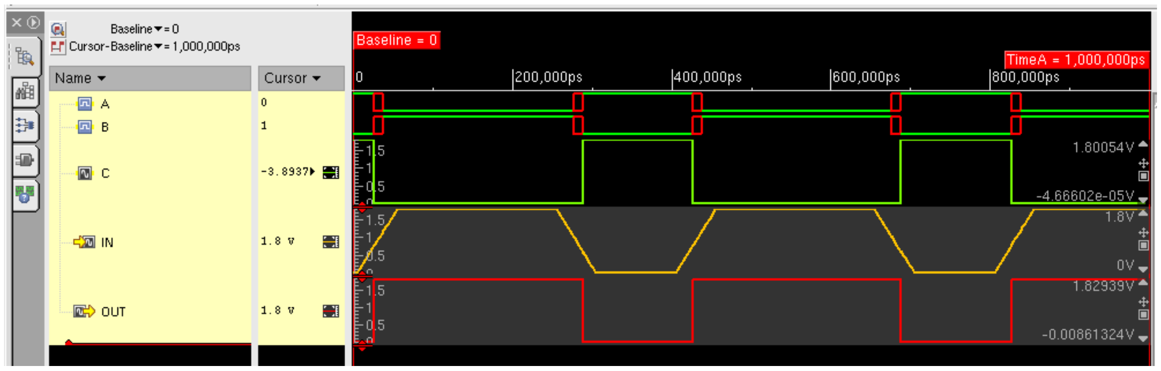
Name	Value
A	0
B	1
C	5.41178e-09 V
IN	1.8 V
OUT	1.8 V

The Simvision waveform window will open after clicking the plot button (see next page).



The **equal icon** fits the waveform horizontally and the **box icon** fits the waveform vertically.

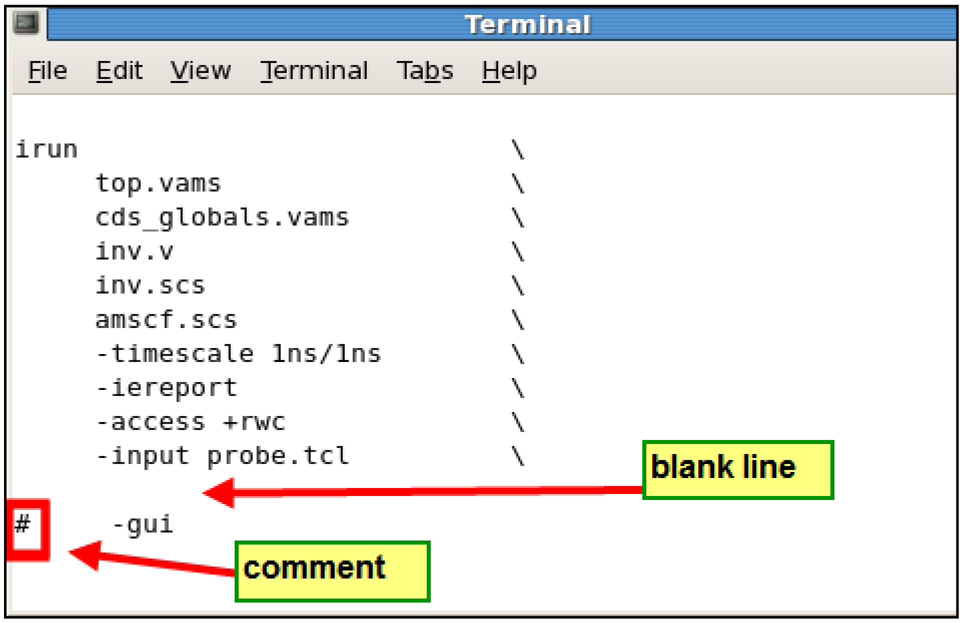
After fitting the waveforms to the window, the result is the same as in part 1 with ADE



13) **Close Simvision** by picking **File → Exit** from any of the Simvision windows. Choose "yes" when prompted to exit.

14) **Edit the** `run_simulation` **script and remove or comment out** `-gui` **option as shown in the screen capture below**

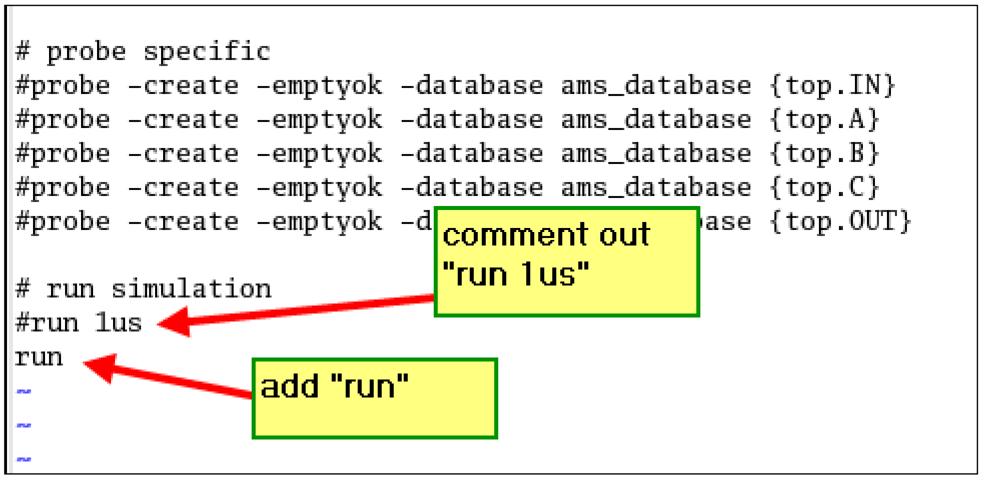
This change will switch the simulation from interactive mode with Simvision to batch mode.



```
irun                            \
     top.vams                   \
     cds_globals.vams           \
     inv.v                      \
     inv.scs                    \
     amscf.scs                  \
     -timescale 1ns/1ns         \
     -iereport                  \
     -access +rwc               \
     -input probe.tcl           \

#     -gui
```

If using the comment method, insert a blank line before the commented options as shown above so that the c-shell line continuation won't break the script.

15) **Edit the** `probe.tcl` **file and remove (or comment out) the 1us stop time as shown below.**



```
# probe specific
#probe -create -emptyok -database ams_database {top.IN}
#probe -create -emptyok -database ams_database {top.A}
#probe -create -emptyok -database ams_database {top.B}
#probe -create -emptyok -database ams_database {top.C}
#probe -create -emptyok -database ams_database {top.OUT}

# run simulation
#run 1us
run
~
~
~
```

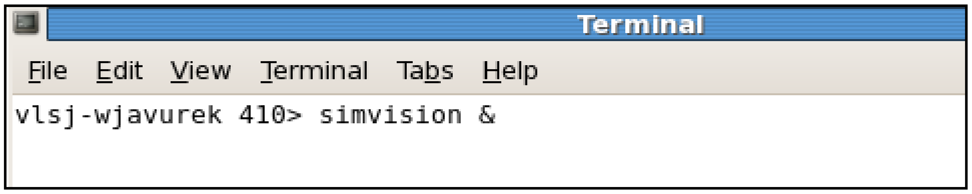
Removing the “1us” stop time in the `probe.tcl` tells AMS to run the simulation to the end which is the 1.1us stop time specified in the `amscf.scs` file.

16) **Run the simulation.with the** `run_simulation` **script**.

The simulation will run to completion and leave you back at the ncsim prompt. To get to the unix prompt, type exit

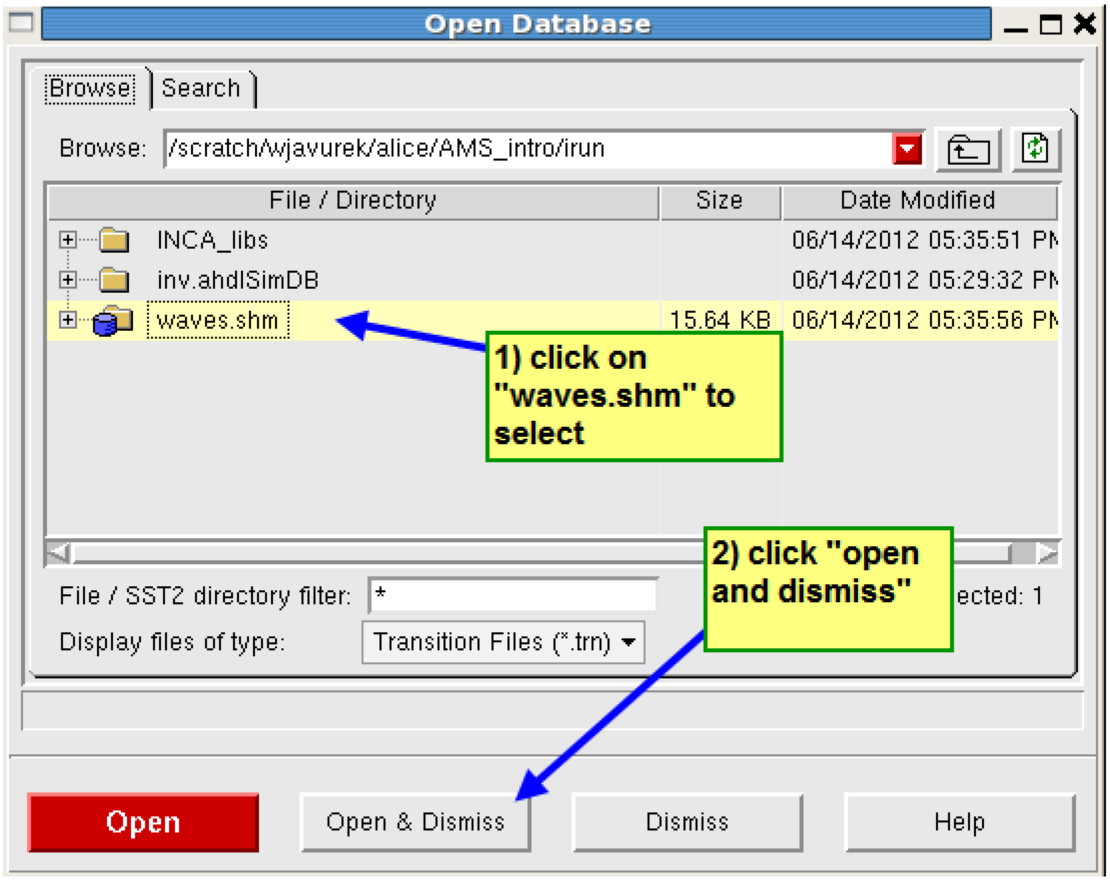
```
Simulation complete via transient analysis stoptime at time 1100 NS
Memory Usage - 17.6M program + 580.7M data = 598.3M total
CPU Usage - 0.1s system + 0.4s user = 0.5s total (45.5% cpu)
ncsim> exit
cds125842 46> █
```

17) **Simvision** or **ViVA** can be used to view the results. **Start Simvision from the terminal window**



```
vlsj-wjavurek 410> simvision &
```

18) Pick **File → Open Database** from Simvision and open the "waves.shm" database as shown below



Browse: /scratch/wjavurek/alice/AMS_intro/irun

File / Directory	Size	Date Modified
INCA_libs		06/14/2012 05:35:51 PM
inv.ahdlSimDB		06/14/2012 05:29:32 PM
waves.shm	15.64 KB	06/14/2012 05:35:56 PM

File / SST2 directory filter: *

Display files of type: Transition Files (*.trn)

ected: 1

19) **Plot the top level signals as described in step 12**

20) **Close Simvision** by picking **File → Exit** from any of the Simvision windows. Choose "yes" when prompted to exit.

This completes the overview of running AMS designer.

Document History

- V7.0, November 11, 2014
  - Update for IC 616 ISR8, UNL, and INCISIV 13.2

- v5.0, June 14, 2012, Bill Javurek
  - update for IC615 ISR 10 and Incisiv 11.1

- v1 – v4, Bill Javurek
  - development versions