cadence®

Cosimulation with SpectreRF and MATLAB*/Simulink*

Rapid Adoption Kit (RAK)

Product Version: IC 6.1.8 ISR4, SPECTRE 18.1 ISR7, MATLAB R2018a
July 2019

Copyright Statement

Contents

Purpose

SpectreRF-MATLAB / Simulink cosimulation stands for cosimulation using the Cadence Spectre simulator and the MathWorks MATLAB®/Simulink®.

This RAK describes how to setup and use a cosimulation link between the MATLAB/Simulink® system-level simulation environment and Cadence Spectre® circuit simulator RF analysis (SpectreRF).

Cosimulation helps to detect concept errors early, detect any design flaws before tape-out and to quickly correct issues and re-simulate.

With cosimulation, system designers can create low-level models of critical analog blocks and use these models one at a time to analyze the performance impact of individual blocks on the system-level simulation.

Audience

This document is intended for Analog and RF designers who are experienced with Virtuoso ADE, SpectreRF and MATLAB/Simulink.

Terms

ADE	Analog Design Environment
RF	Radio Frequency

Introduction

The high-level system concept is often specified in the early stages of design work. This process is well supported by both concept engineering and system-level simulation tools such as MATLAB/Simulink.

Cosimulation using SpectreRF and MATLAB/Simulink combines the best of system-level simulation with block-level analog and RF simulation. Simulink provides large libraries of DSP algorithms for generating complicated signals and post-processing, while SpectreRF supports transient and envelope analysis of common RF and communication circuits, such as mixers, oscillators, sample and holds, and switched capacitor filters at both the transistor and behavioral levels.

Running the RAK

This RAK has been developed and tested on a machine having OS RHEL7.6 version. MATLAB2018a requires minimum OS RHEL6.7 version or higher. SPECTRE18.1 requires OS RHEL6 or higher and IC6.1.8 requires OS RHEL6.5 or higher version. The estimated time to complete this tutorial is about three hours.

Setting up the Tutorial Database

1. Unzip and untar the attached database `RAK_SpectreRF_simulink_example_R2018a.tar.gz`

   ```
   Unix> tar zxvf RAK_SpectreRF_simulink_example_R2018a.tar.gz
   ```

2. Navigate to `SpectreRF_simulink_example` directory.

   ```
   Unix> cd SpectreRF_simulink_example
   ```

   This is the best place to start MATLAB, because the cosimulation modules load automatically here in accordance with the statements provided in the `startup.m` file.

3. Add the SpectreRF and MATLAB/Simulink engine install paths to the `MATLABPATH` environment variable. You can automate this step by adding the appropriate commands to your `.cshrc` file.

   First check for the ‘Spectre’ installation directory, by performing:

   ```
   Unix> which spectre
   ```

   It should return:

<SPECTRE_install_path>/tools/bin/spectre

For example:

/cadtools/serv1/cds/SPECTRE181ISR_lnx86/tools/bin/spectre

Now use the output of `cds_root spectre` to set the MATLABPATH environment variable as shown.

```
Unix> echo `cds_root spectre`
```

Should return something like: /cadtools/serv1/cds/SPECTRE181ISR_lnx86

```
Unix> setenv MATLABPATH `cds_root
spectre`/tools/spectre/matlab/64bit:$PATH

Unix> setenv MATLABPATH `cds_root
spectre`/tools/spectre/simulink:${MATLABPATH}
```

You may refer the already available **SETUP** file provided in this database, and perform:

```
Unix> source SETUP
```

before launching MATLAB.

The designs for the tutorials are provided in the tutorial database libraries. If the setup for the standalone simulations work correctly in MATLAB/Simulink and SpectreRF, then only three steps are required to set up the cosimulation:

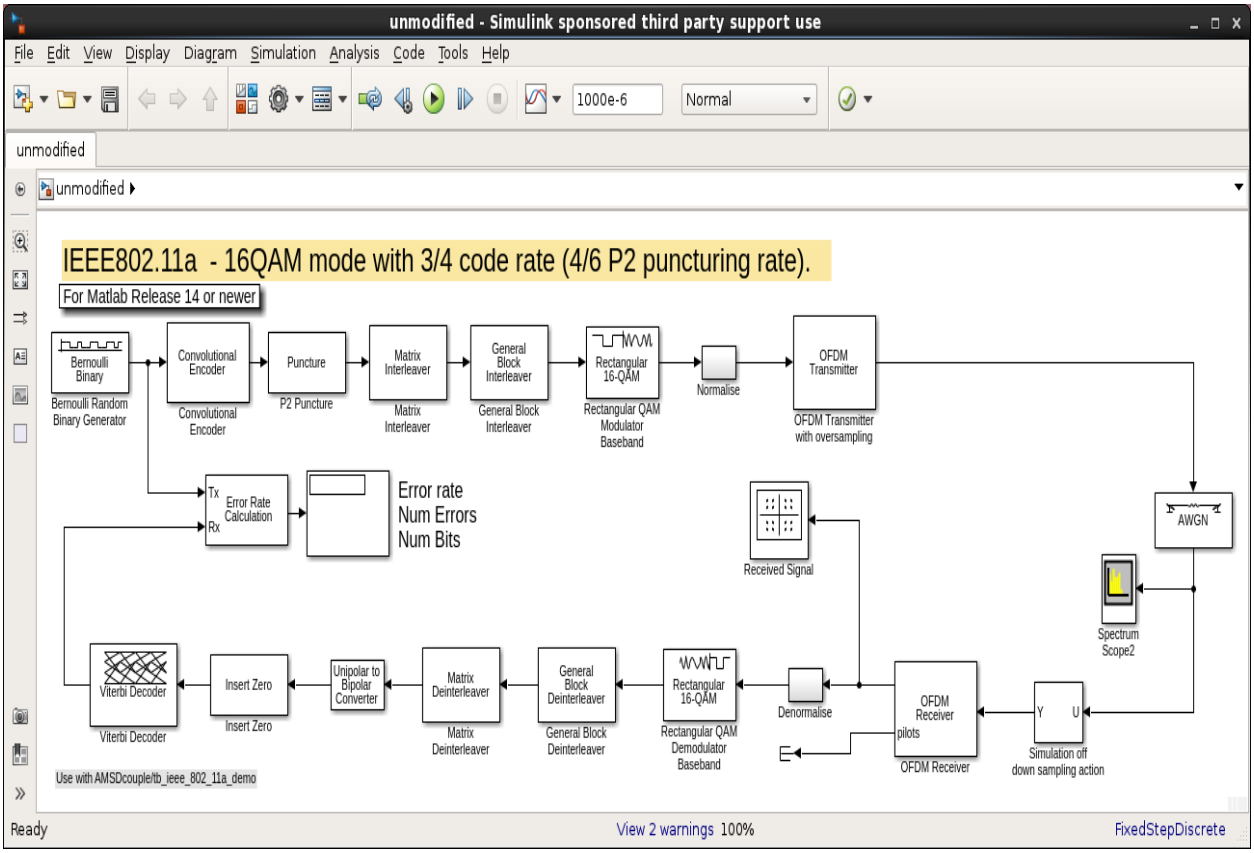
1. Insert and configure the SpectreRF engine in the Simulink schematic.
2. Setup the netlist to adopt the SpectreRF and MATLAB cosimulation.
3. Run the cosimulation.

Tutorial 1

Tutorial 1 describes cosimulation for the system-level module of a wireless LAN. Figure 1 shows the system-level schematic. The transmitter blocks encode and modulate binary random data and send the OFDM signal to a white Gaussian noise channel model.

The receiver blocks demodulate and decode the channel output. Finally, the received bits are compared with the original bit stream to compute the bit error rate.

Figure 1: End-to-End System-Level simulation with Simulink



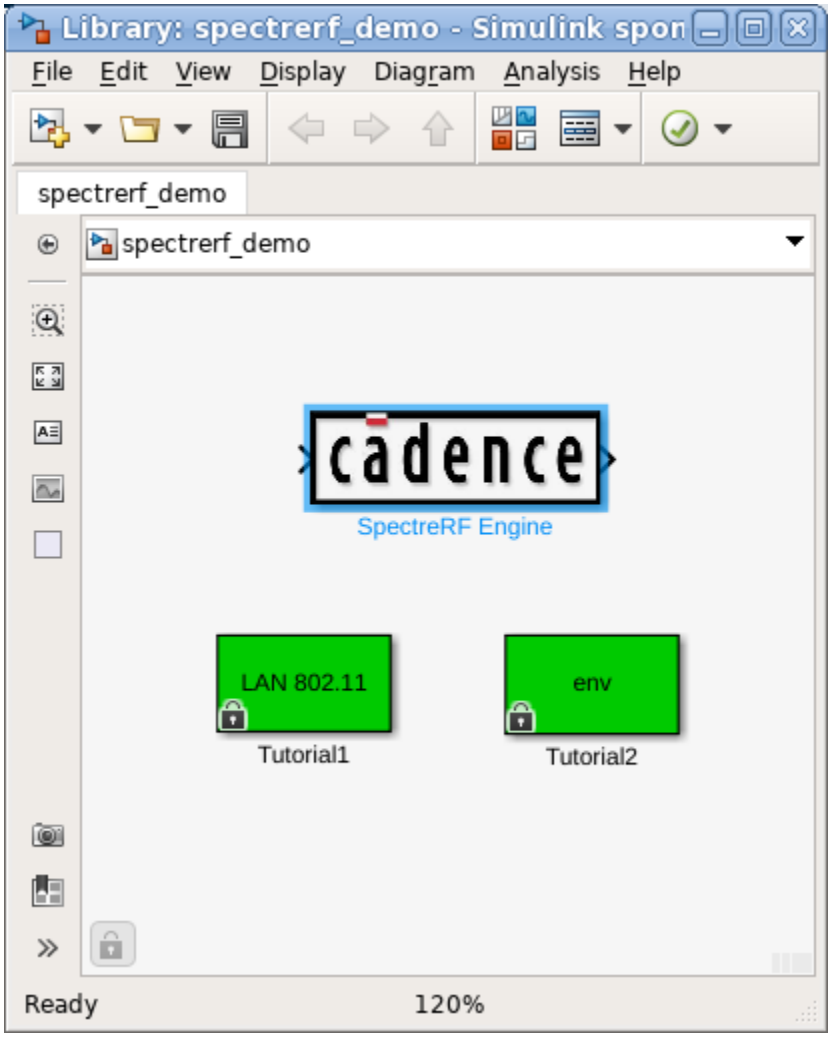
Validating the Simulink design

1. Start MATLAB, by typing the below command

   ```
   Unix> matlab&
   ```

   The `spectrerf_demo` Library opens, as shown in Figure 2.

Figure 2: spectrerf_demo Matlab Simulink Library for cosimulation

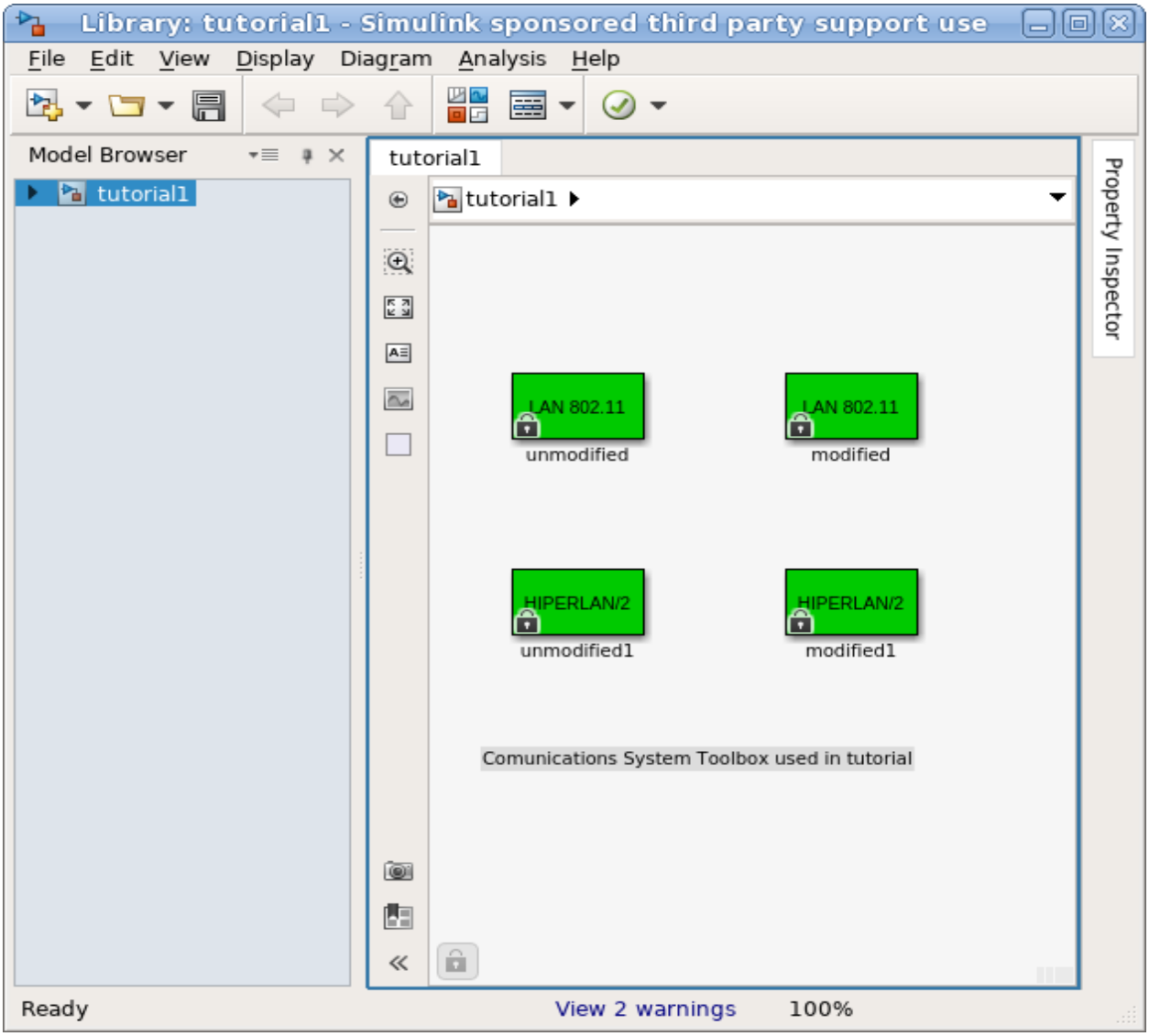


2. This library contains the SpectreRF engine block module for this tutorial (distinguished by the `Cadence` logo and labeled `SpectreRF Engine`). You can insert this '`SpectreRF Engine`' coupler module in any Simulink design by dragging and dropping it from this library.

3. Double-click the first green box labeled '`LAN 802.11 Tutorial1`' from the '`spectrerf_demo`' library (or click RMB and choose 'Open' option).

   The 'tutorial1' library opens, as shown in Figure 3, having the 'LAN 802.11' and 'HYPERLAN/2' blocks, each having an 'unmodified' as well as 'modified' block.

Figure 3: tutorial1 Library including the 'modified' and 'unmodified' models



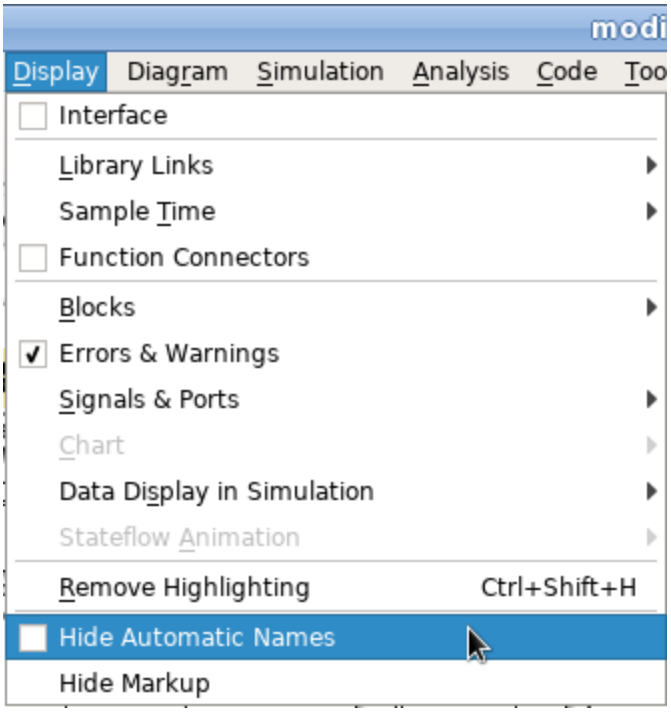
4. Double-click (or RMB and choose '`Open`' option) the '`unmodified`' block for the '`LAN 802.11`' first.

   The end-to-end design for a wireless LAN transmission system opens (as shown in Figure 1 above). This design contains the main DSP algorithms of a wireless LAN 802.11a physical layer for the transmitter and the receiver chains. It generates a standard-complaint 802.11a signal and implements the related post-processing algorithms.

   Note: Staring Matlab R2017b release, you might see that by default some of the block names are hidden in the Simulink model.

a. Click on the “Display” tab on the menu options at the top of the Simulink model editor.
b. Then uncheck the “Hide Automatic Names” option as shown below.

Figure 4: Display block names in Simulink model



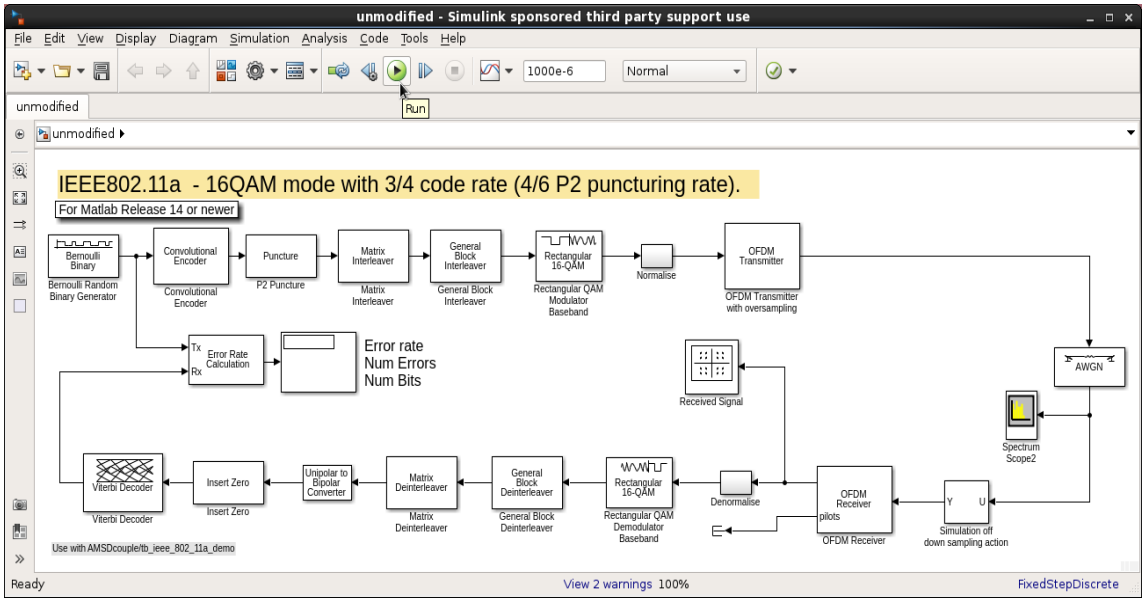
5. In the window displaying the LAN transmission system schematic of Figure 1, first ensure that all the schematic blocks are connected properly, and there are no missing `'Unresolved Link'` blocks.

   If there are any ‘Unresolved Link’ blocks, double-click on each of the problematic blocks and ensure its `'Block Parameters'` are set correctly.

   Navigate to each of the block properties tab and check if every option setting is as expected. If there are any changes, click ‘`Apply`’ and then ‘`OK`’ on the properties form.

   Save the schematic changes (if any) and click on the green `'Run'` button (or `Simulation -> Run` option) from the Simulink ‘unmodified’ block as shown in Figure 5.

Figure 5: Running standalone 'unmodified' Simulink testbench



A `spectrum scope` and a scatter diagram (also known as `Constellation Diagram`) comes up as shown in Figure 6 and Figure 7. The minor deviations in the scatter diagram originate from the noisy channel. Look at the `Bit Error Rate` (`BER`) display as shown in Figure 8.

Figure 6: Spectrum Scope of 'unmodified' testbench

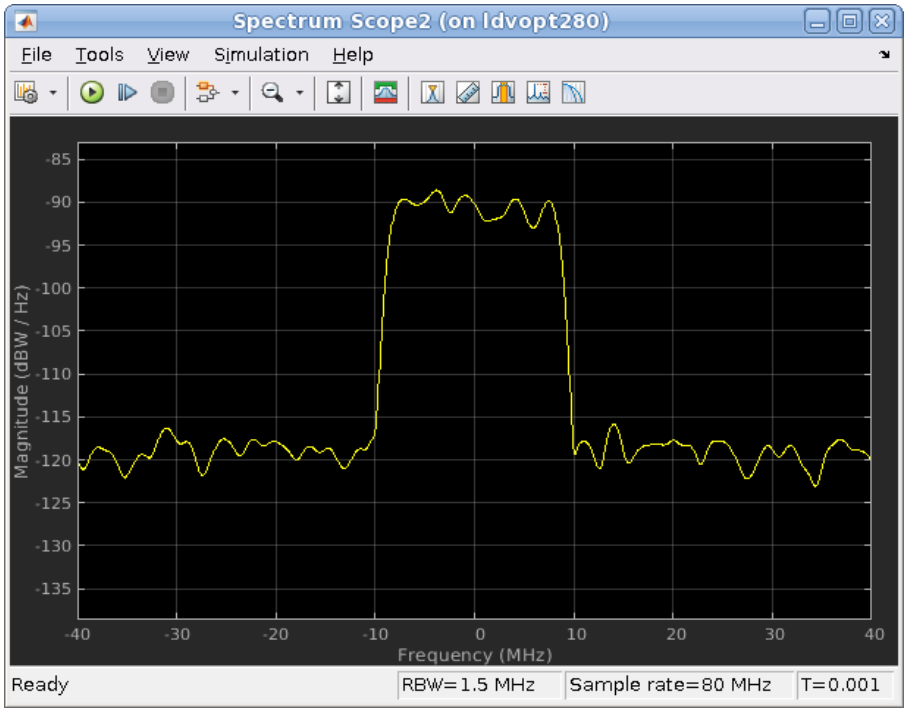


Figure 7: Scatter Plot of 'unmodified' testbench

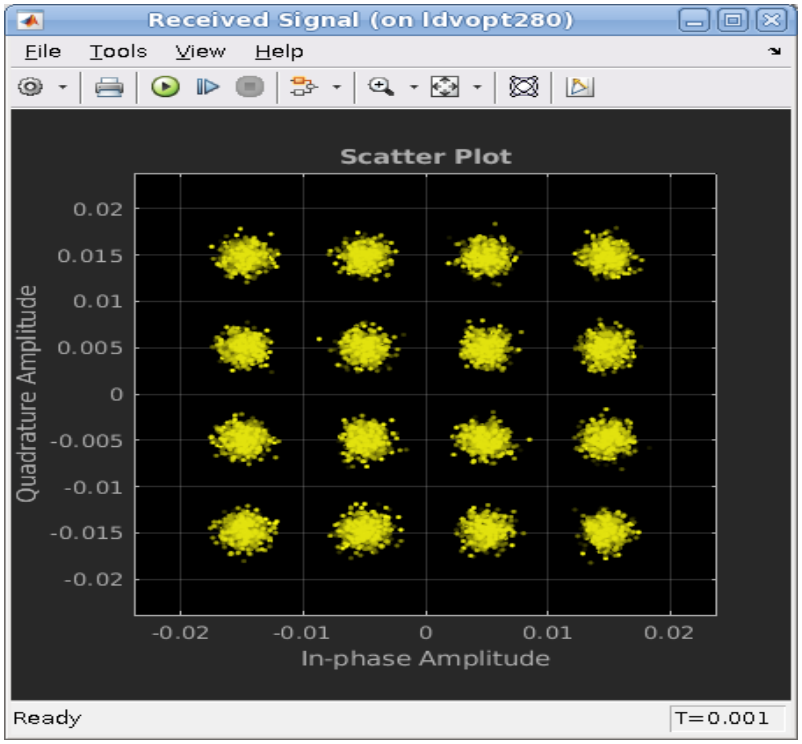
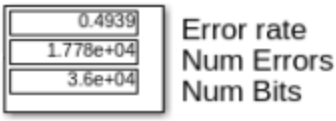
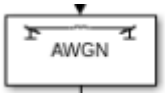


Figure 8: Bit Error Rate (BER) display of 'unmodified' testbench

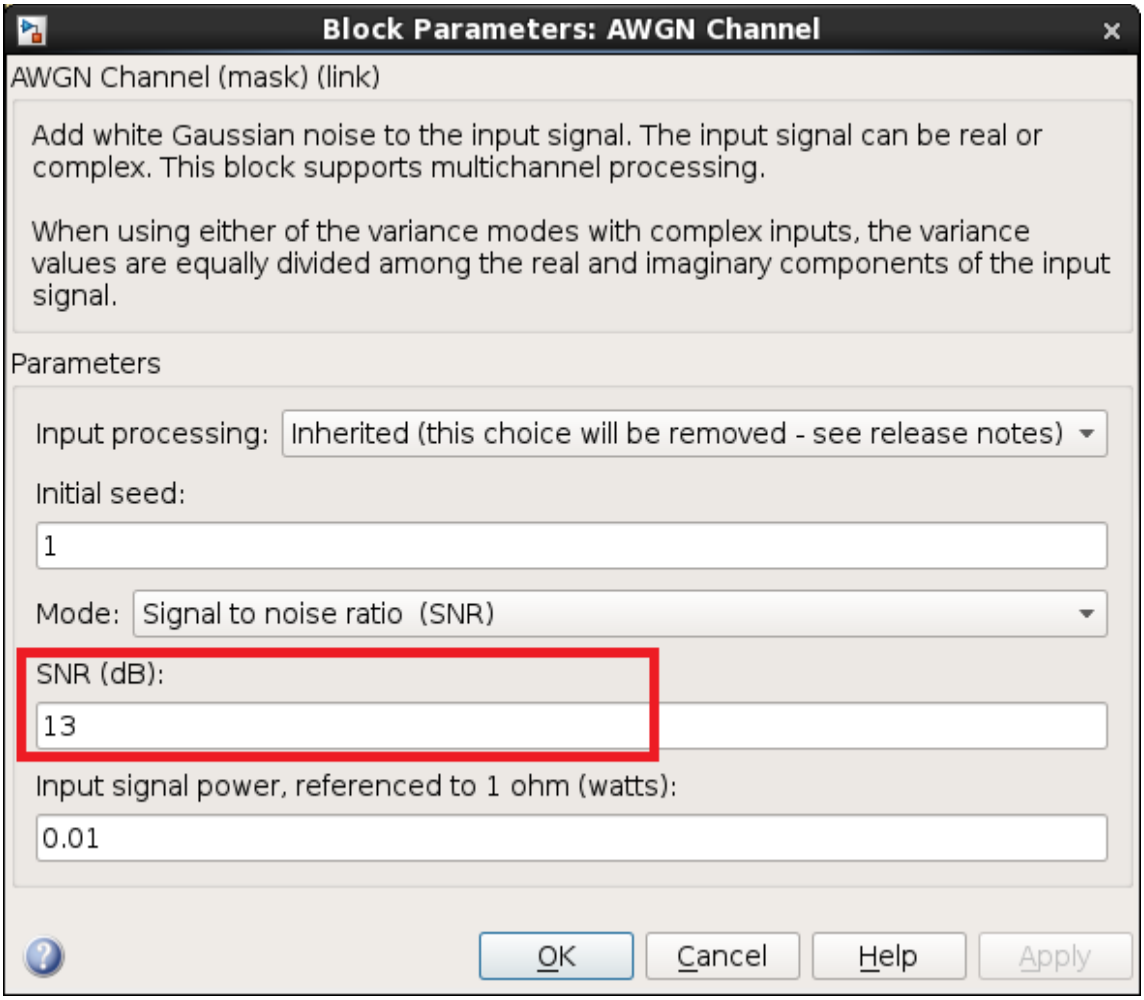


6. **(Optional)**

   a. Double-click the `AWGN` block 

from '`unmodified`' testbench. The `Block Parameters: AWGN Channel` form opens.
   b. Increase the noise, by changing the parameter `Signal to Noise Ratio 'SNR'(dB)` from 20 (original) to 13 (new value) as shown in Figure 9.
   c. Click '`Apply`' and then '`OK`' the `Block Parameters: AWGN Channel` form.

Figure 9: AWGN Channel block parameter setting form



7. Click `Simulation -> Run`, (OR) hit the green 'Run'  button to start the simulation again.

   The scatter plot of '`Received Signal`' (i.e. the `Constellation Diagram`) is now distorted as shown in Figure 11 and after some seconds, `Bit Error Rate (BER)` display also changes as shown in Figure 12. The `Spectrum Scope` is also now changed as shown in below Figure 10 (with respect to the one shown in Figure 6).

Note: You may need to use the '`Zoom-In`'  feature available in the below plots to see the required region of interest.

Figure 10: New Spectrum Scope of 'unmodified' testbench

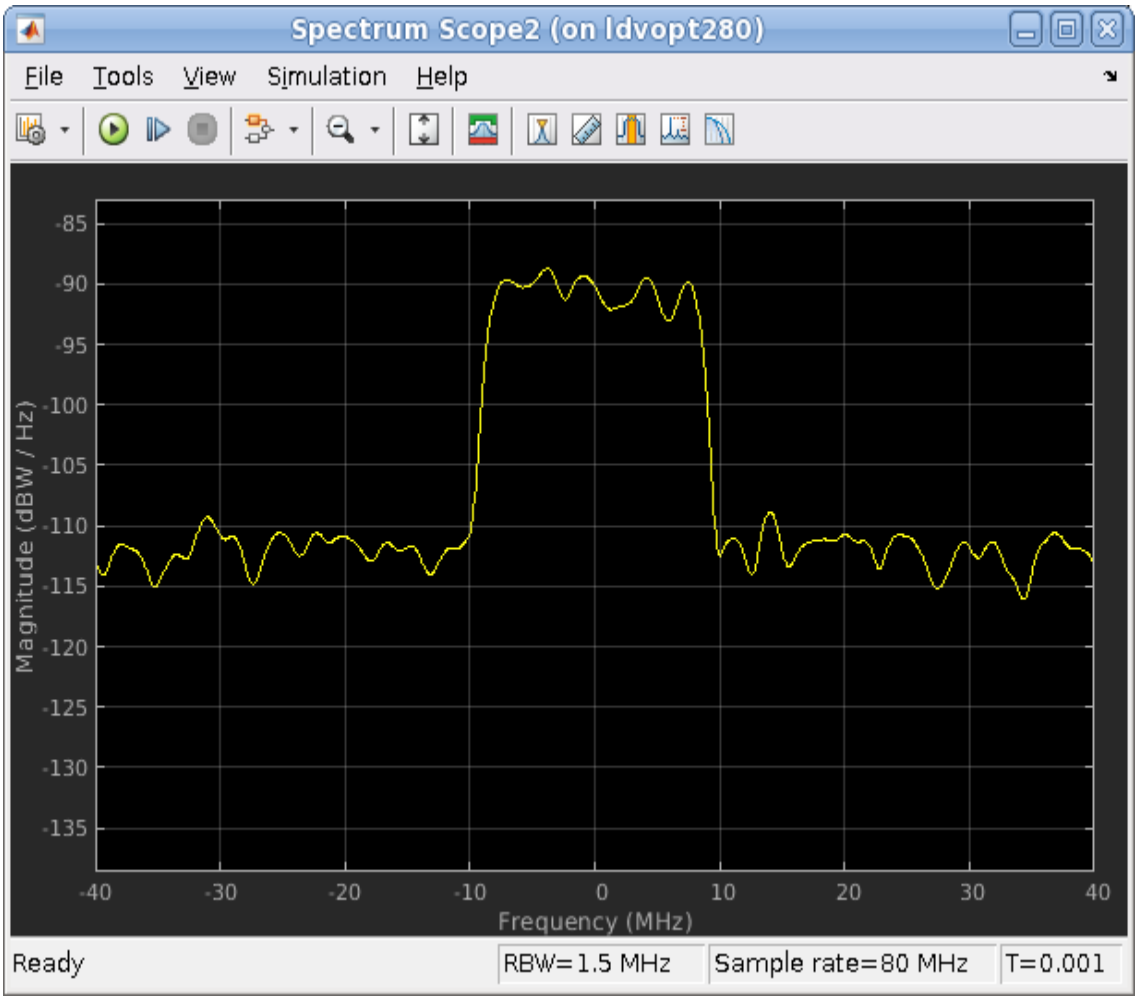


Figure 11: Distorted Scatter Plot of 'unmodified' testbench

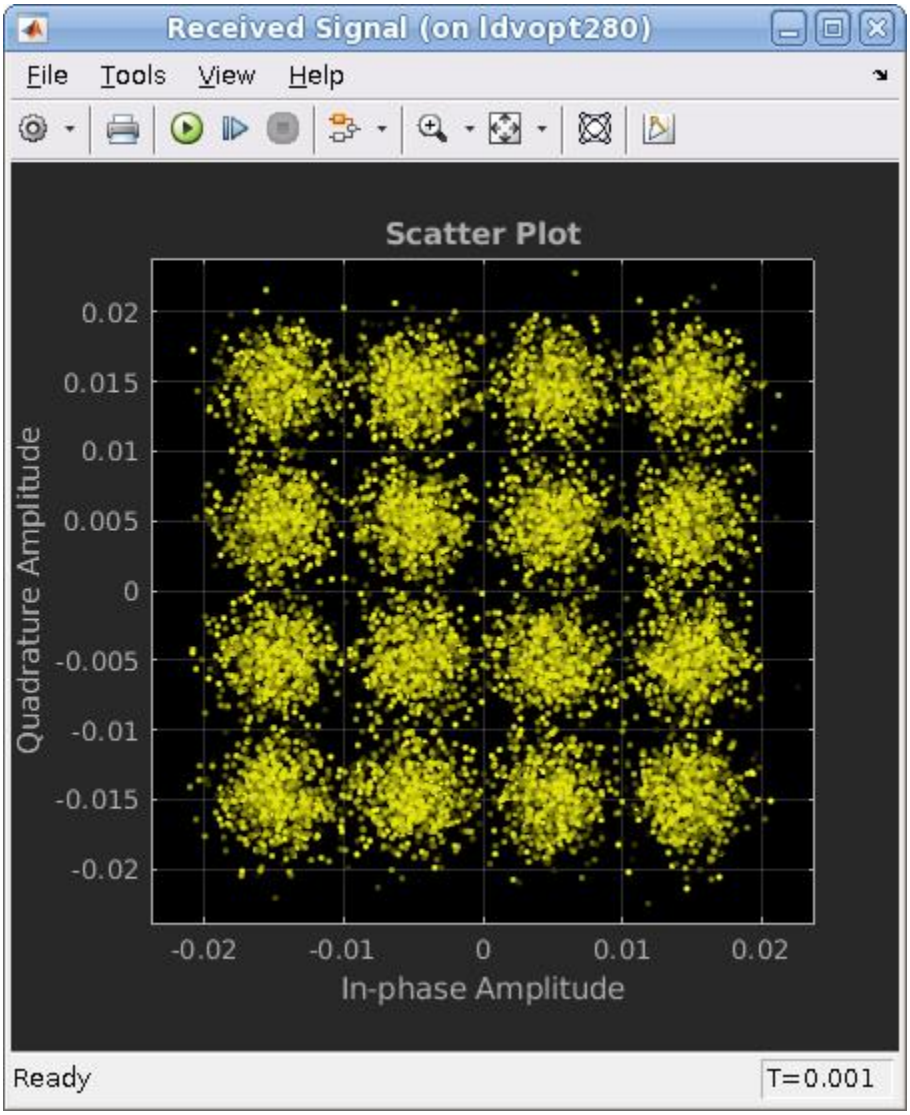
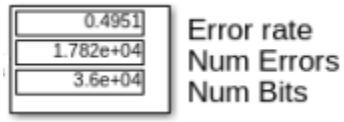


Figure 12: New Bit Error Rate (BER) display of 'unmodified' testbench

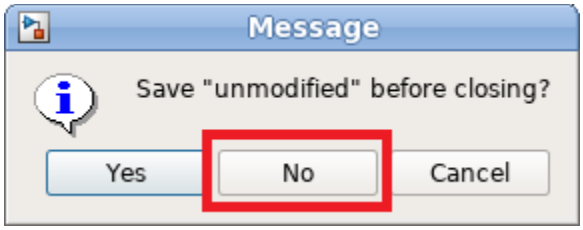


8. Close all the above opened `Scatter Plot` and `Spectrum Scope` windows.

   Close the '`unmodified`' Simulink testbench by clicking the right-top most '`Close Window`'  .

You may want to select 'No' option in the 'Save "unmodified" before closing?' pop-up form if you want to close the testbench without saving any changes as shown in Figure 13.

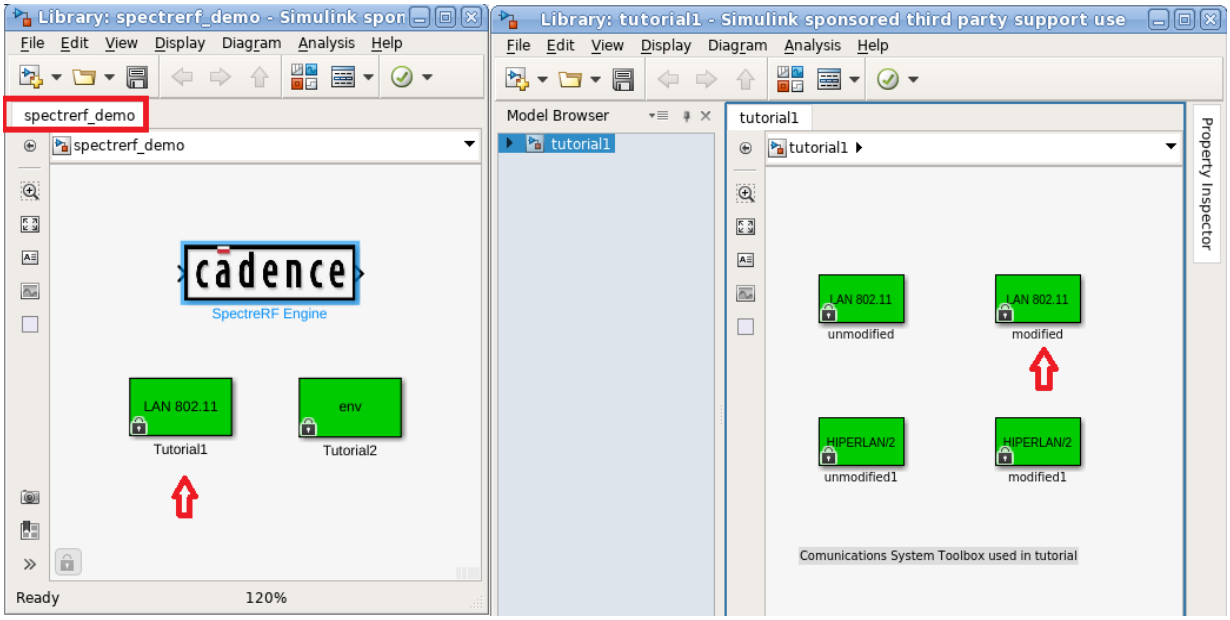
Figure 13: Save testbench pop-up form



Modifying the design and Setting up the cosimulation from MATLAB

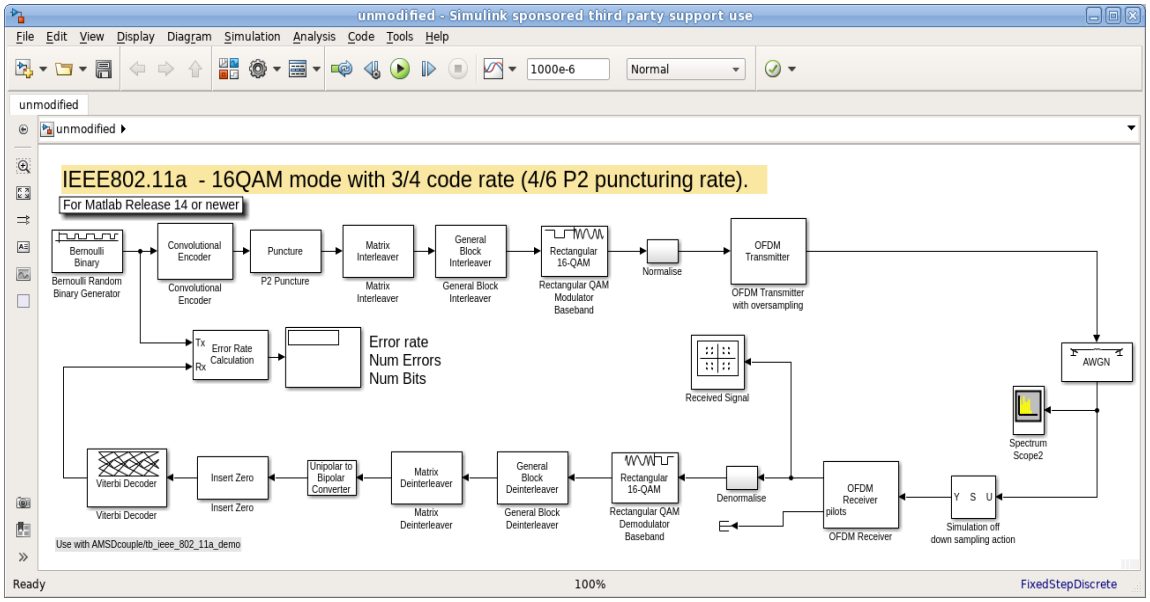
The following steps are setup for the cosimulation. There is a complete setup in the LAN 802.11 Tutorial1 of 'spectrerf_demo', named as 'modified' for 'LAN 802.11' block already available that can be used as reference, as shown in Figure 14.

Figure 14: 'modified' testbench of LAN 802.11 tutorial1 in 'spectrerf_demo' Library



9. Assuming you have already closed the 'unmodified' Simulink design testbench from above steps, launch the 'unmodified' LAN 802.11 tutorial1 testbench again to perform the below steps by double clicking on it as shown in Figure 15.

Figure 15: 'unmodified' LAN 802.11 Simulink testbench



10. From the '`spectrerf_demo`' Library, copy (by RMB on it, and selecting '`Copy`' option) the `SpectreRF Engine` block (the block that contains the `Cadence` logo, located on the top of green `Tutorial1` and `Tutorial2` boxes shown in Figure 16) into the '`unmodified`' testbench and place the block at the top right corner.

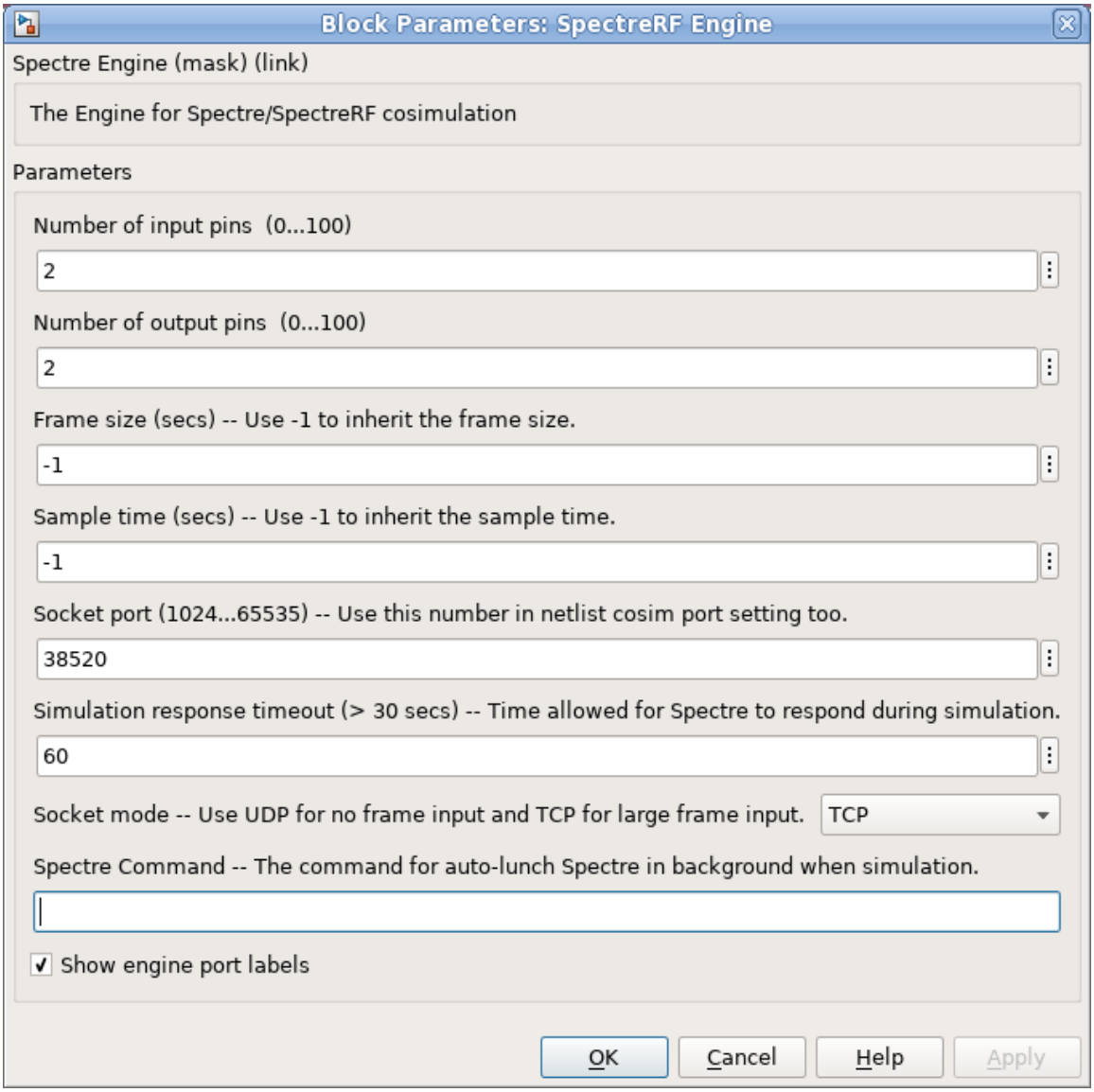
Figure 16: SpectreRF Engine block

This step inserts the analog transmitter RF-front end within the system testbench between the `OFDM Transmitter` and `AWGN` channel.

11. Double-click the `SpectreRF Engine` block.

The '`Block Parameters: SpectreRF Engine`' window opens, as shown in Figure 17.

Figure 17: Block Parameters window



The fields inside this '`Block Parameters`' form have the following meanings as shown in Table1:

Table 1: SpectreRF Engine Block Parameters fields and their details

Field	Meaning
Number of input pins (0…100)	The number of input pins to the engine block (0…100). The input to the block is the input signal from Simulink.

Number of output pins (0…100)	The number of output pins from the engine block (0…100). The output from the block is the resulting signal from the SpectreRF simulator.
Frame size (secs)	The number of samples per frame (any positive integer or -1; default: -1). Set this value to -1 to inherit the frame size from the input.
Sample time (secs)	Sample period of the block, in seconds (default: -1). The Sample time is the frame period, not the time between the samples inside the frame. Set this value to -1 to inherit the sample period of the connected blocks.
Socket port (1024…65535)	The number of the service that is identified by TCP/IP. Normally, the system reverses port numbers less than 1024, so this value can range from 1024 to 65535 (default: 38520).
Simulation response timeout (>30 secs)	Maximum time, in seconds, to wait for an answer from the SpectreRF simulator during simulation. During simulation, if MATLAB does not receive a response from Spectre before the timeout period expires, simulation ends with an error.
Socket mode	TCP or UDP. Choose UDP if frame size is less than 50. TCP and UDP are two different translate protocols in TCP/IP. Without CRC, UDP is usually faster in a good network environment with small data packages and TCP is usually better for large data packages.
Spectre Command	Used to run a Spectre simulation. This parameter enables a use mode where Spectre is called internally by the Simulink simulation.
Show engine port labels	If checked, the SpectreRF Engine shows the label information.

12. In this '`Block Parameters`' form,

    a. Set the '`Number of input pins`' to '`2`'.

b. Set the 'Number of output pins' also to '2'.

c. Set the 'Frame size (secs)' field to -1, to use an inherited frame size. This Simulink testbench uses framed signals.

d. Leave the 'Sample time (secs)' field set to default -1.

e. Set the parameter 'Socket port' to any value from 1024 to 65535. Here, we have set it to 38520.

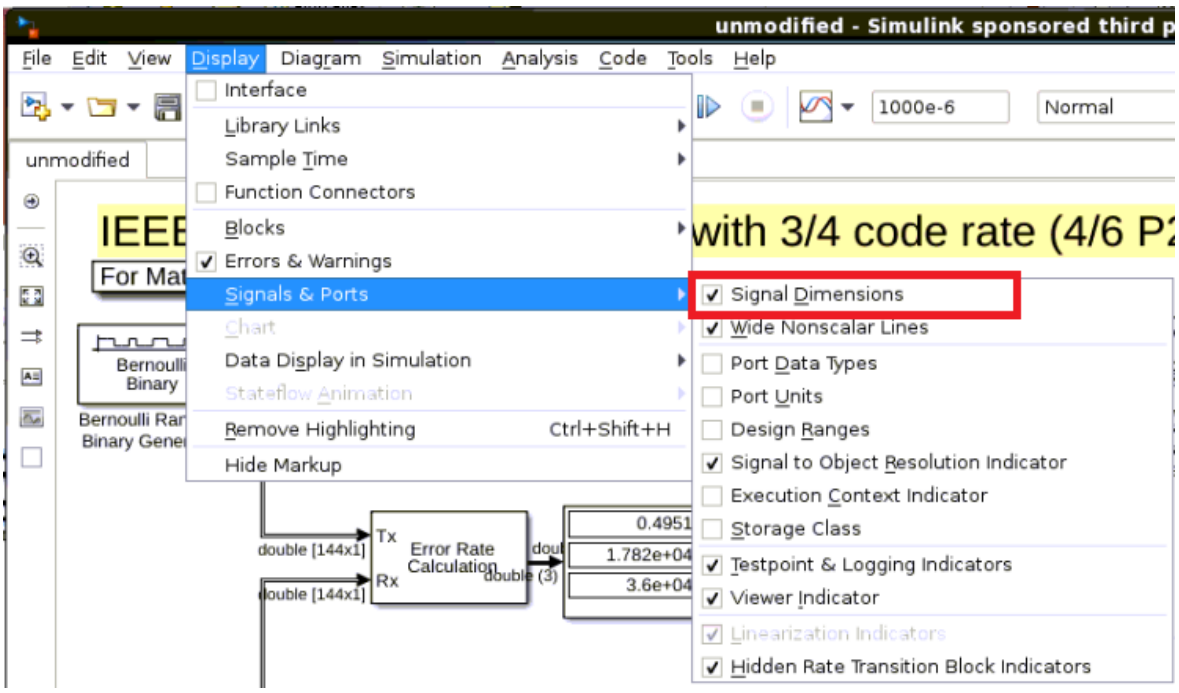
f. Leave the 'Simulation response timeout (>30 secs)' field set to default 60.

g. Ensure that the 'Socket mode' is set to 'TCP', and tick the checkbox next to 'Show engine port labels' option.

h. Click 'Apply' and then 'OK' on this Block Parameters form. The form closes, and the 'SpectreRF Engine' block in the 'unmodified' testbench is updated with the correct number of pins.

13. From the testbench window, ensure that the Display -> Signals & Ports -> Signal Dimensions option is checked, like the one shown in Figure 18.

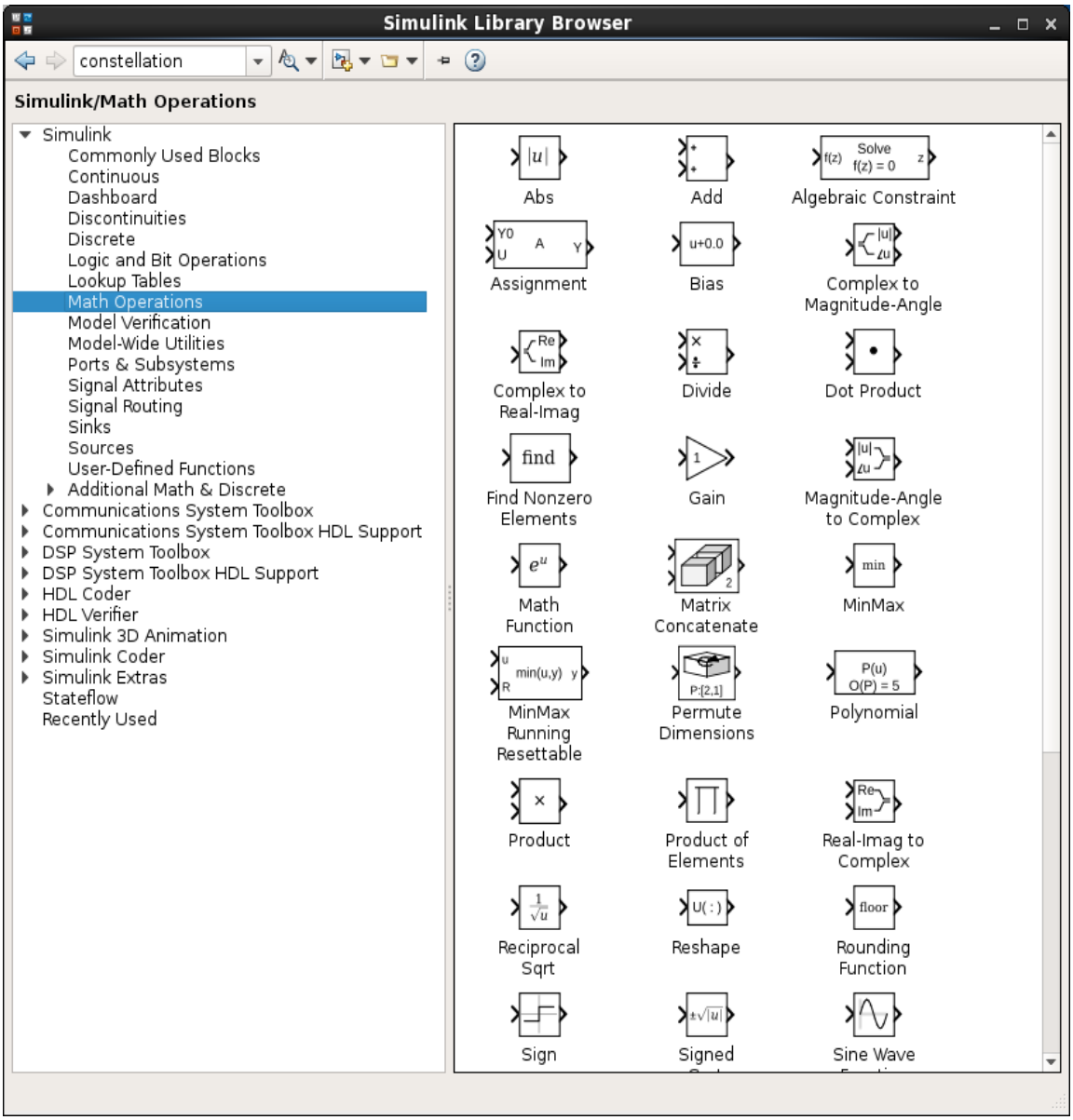
Figure 18: Set the Signal Dimensions display



The signal dimension display is switched ON, so you can see details of the framed signals used. In this case, the frame size between `OFDM Transmitter` (OFDM Transmitter with oversampling) and `AWGN` is 320.

14. From the testbench, select `View -> Library Browser`. It will take a while to launch '`Simulink Library Browser`'. Click on the '`Math Operations`' under the '`Simulink`' section as shown in Figure 19.

Figure 19: Simulink Library Browser



The Simulink/Math Operations library appears. This tutorial uses complex valued signals. Before the signal is transmitted to SpectreRF Engine block, the complex

signals must be split into their real and imaginary parts. Simulink provides a converter for this purpose. The desired converters are located at the bottom right corner of this Math library.

15. Drag and drop the `Complex to Real-Imag` block (Figure 20) **before** the `SpectreRF Engine` block, and then drag and drop the `Real-Imag to Complex` block (Figure 21) **after** the `SpectreRF Engine` block in the simulink testbench.

Figure 20: Complex to Real-Imag block

Figure 21: Real-Imag to Complex block

16. Insert an ideal '`Gain`' block (Figure 22) at the output of the `AWGN` coupler block. This Gain block is available in the same Simulink/Math Operations library.

Figure 22: Gain block

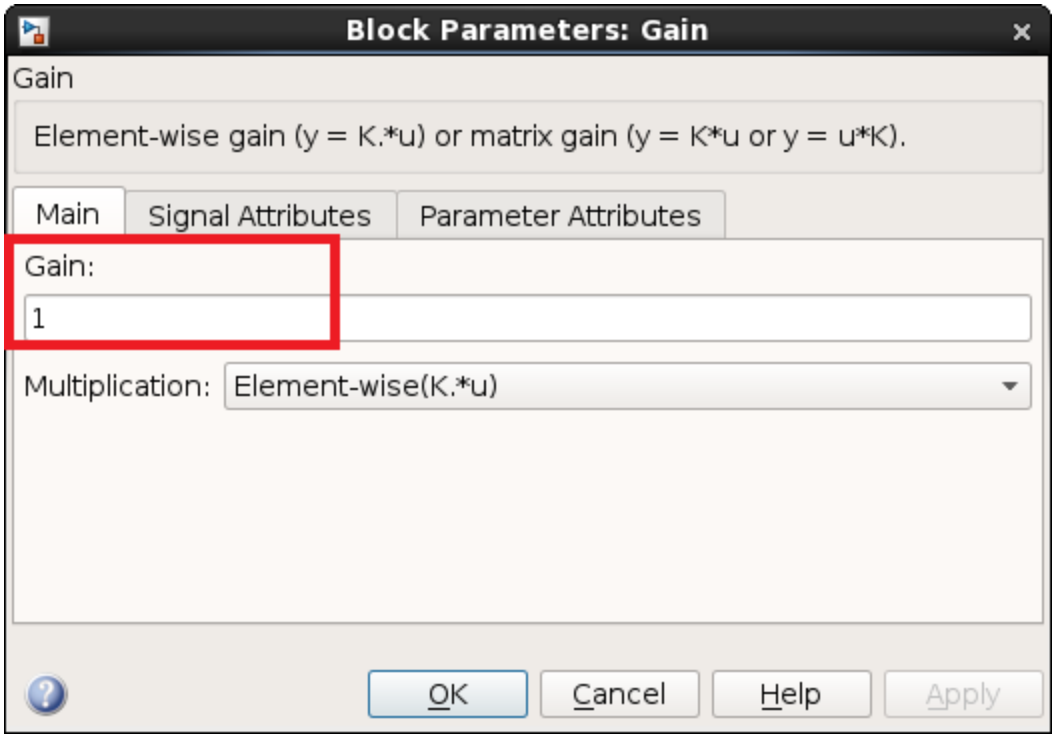
Because the analog/RF subsystem changes the signal level, the Gain block is required to adapt the signal level to the properties of the digital baseband receiver.

17.

    a. Double-click the placed '`Gain`' block and set the '`Gain`' to its default '`1`' value as shown in Figure 23.

    b. Click `Apply` and then click `OK`.

Figure 23: Setting Gain block parameters



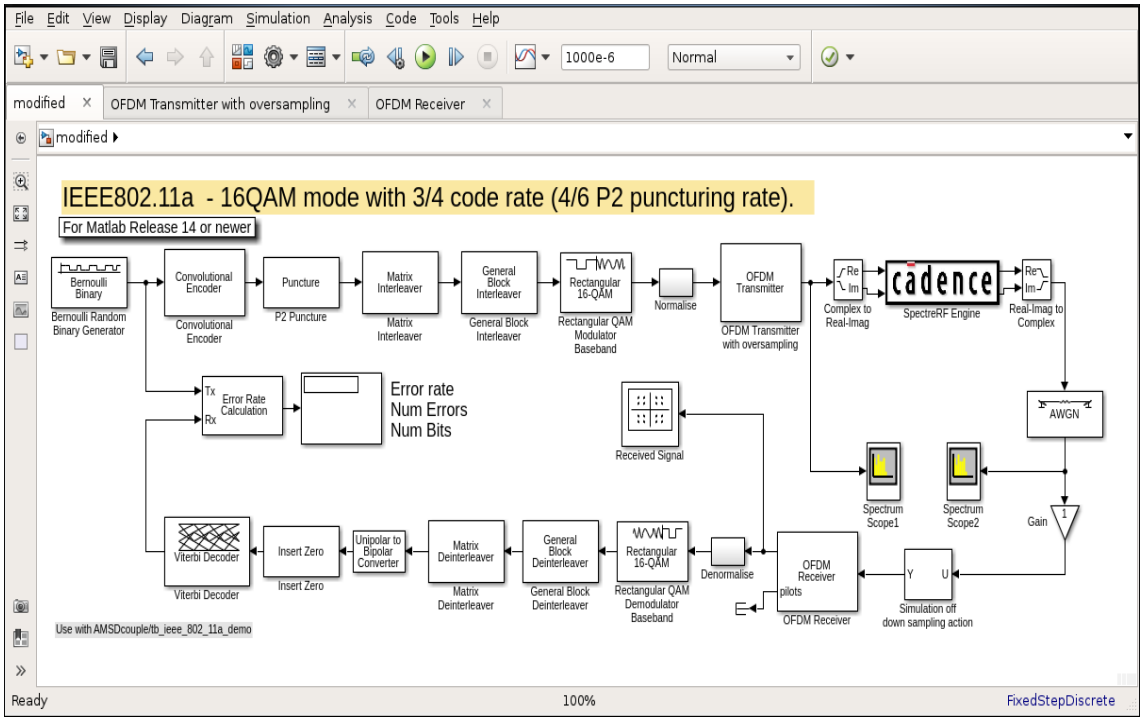
18. From the '`Simulink Library Browser`' search for the keyword '`Spectrum Analyzer`', corresponding Spectrum Analyzer block (Figure 24) will appear. Drag and drop the `Spectrum Analyzer` block and connect it to the output of '`OFDM Transmitter`' in the simulink testbench.

Figure 24: Spectrum Analyzer

19. Connect all the signals to make the final '`unmodified`' testbench, as shown in Figure 25. This should be exactly same to the reference '`modified`' testbench already available.

    Note: To make each signal connection, move the mouse pointer over the module pin. The pointer changes to a cross. Press the left mouse button, move the cursor to the destination pin and release the button.

Figure 25: Final updated 'unmodified' testbench for cosimulation



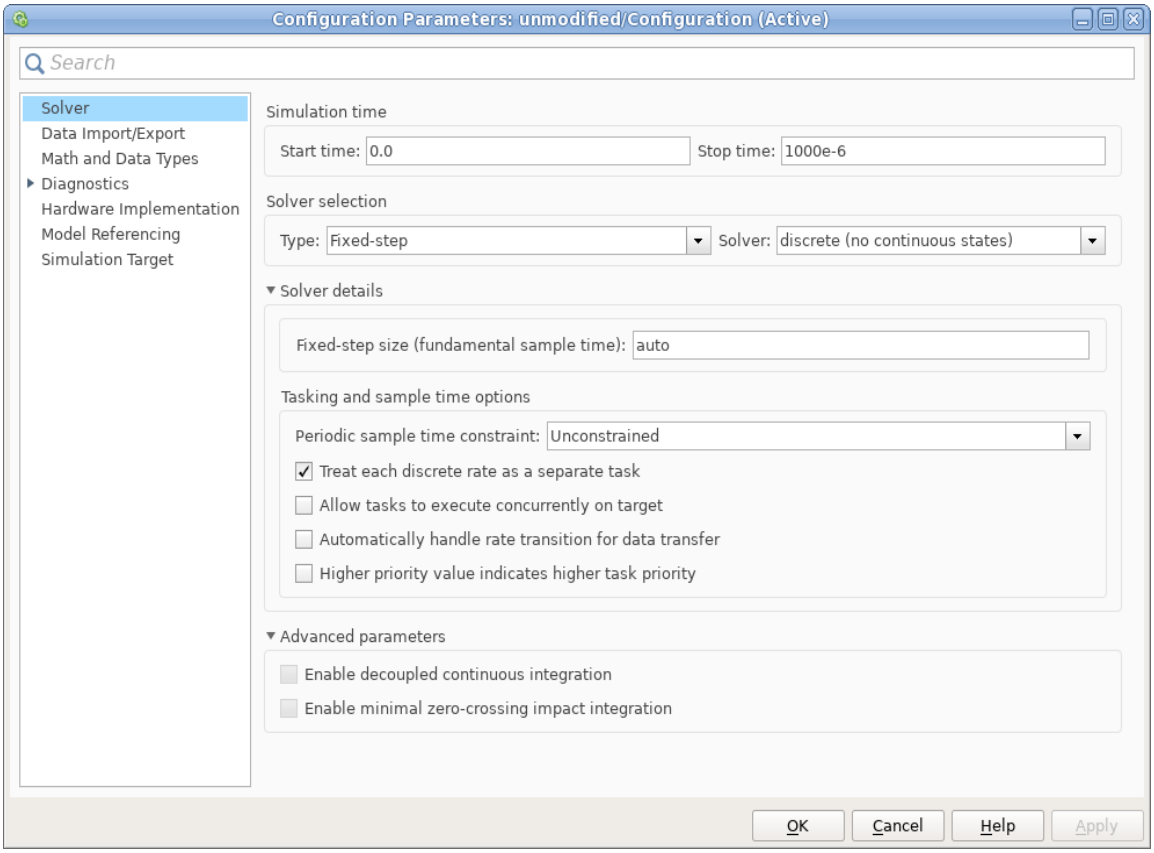
20. Save the final testbench, by clicking on the '`Save`' button  or choose `File -> Save` option from the testbench.

    The Simulink model is now ready for cosimulation. All the other information about frame size and sampling time is detected automatically with the Simulink model.

21. From the '`unmodified`' testbench, choose `Simulation -> Model Configuration Parameters` option.

    The '`Configuration Parameters`' form opens, as shown in Figure 26.

Figure 26: Configuration Parameter settings for cosimulation



22. In the '`Configuration Parameters`' form:

    a. Set the stop time of the Simulink simulation to `1000e-6 in 'Stop time'` field.

    b. Leave all other settings to their default, and click '`OK`' to close the '`Configuration Parameters`' form.

23. Save the changes if any to the testbench. You can also Save the '`unmodified`' design with a different name, by navigating to `File -> Save As…` option and give it a desired name.

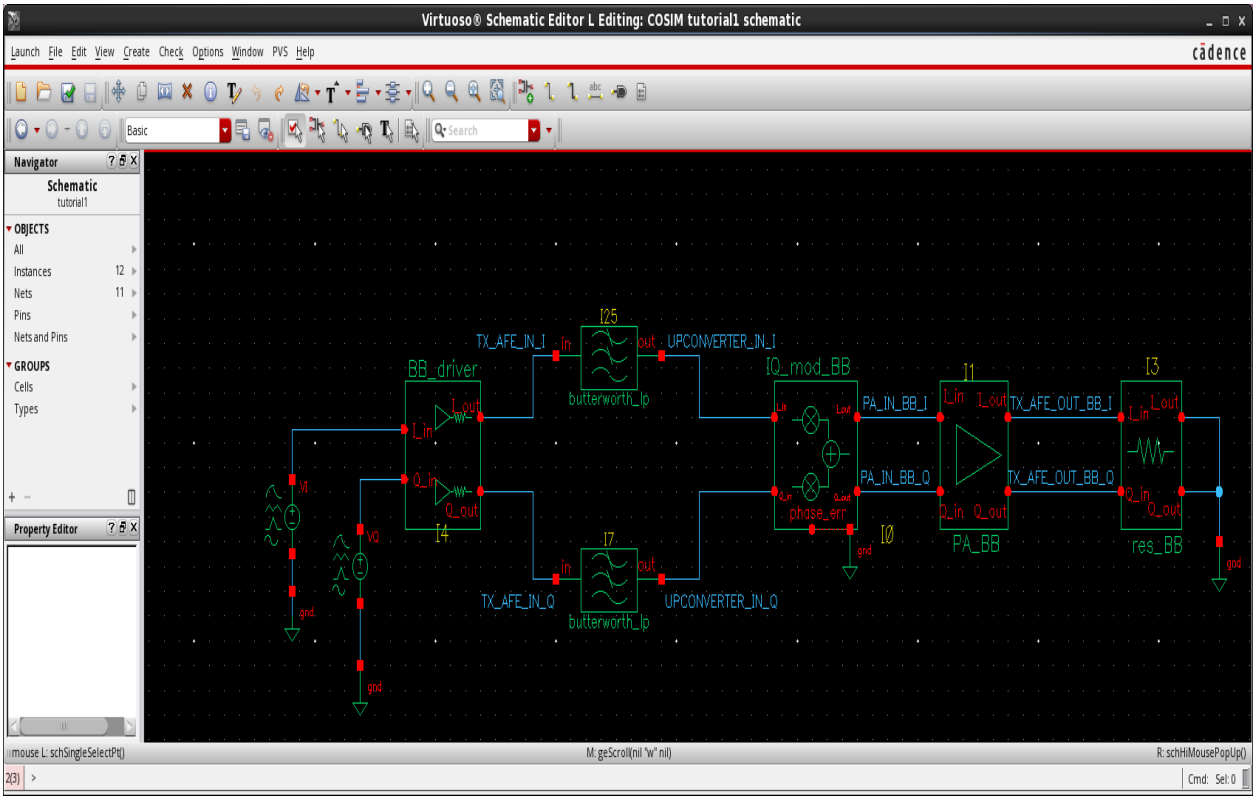
This completes the necessary modifications on the MATLAB/Simulink side. To continue with the further steps in this RAK, you can use the updated Simulink '`unmodified`' testbench that you have just created from above steps (`OR`) use the already available '`modified`' testbench in the `Tutorial1 'spectrerf_demo'` database of '`LAN 802.11`' block.

Setting up the cosimulation from ADE

This section describes how to setup a cosimulation using the Virtuoso `ADE`. If you are using the standalone version of SpectreRF, skip this section and go to the following section "`Setting up the cosimulation from Standalone SpectreRF`".

In this section, the schematic shown in Figure 27 includes an RF transmitter module design with filters, and up-converting mixer, and an amplifier. To speed-up the simulation, the RF components are modeled in the complex base-band domain using Verilog-A. You can use the same approach to perform equivalent behavioral passband and transistor-level simulations.

Figure 27: Baseband model for RF Front-End Transmitter



Note: This schematic has two `pwl` '`vsource`' components at the input. These are used as input to test the model. For cosimulation, these two sources are overloaded and generate output with the input data from MATLAB.

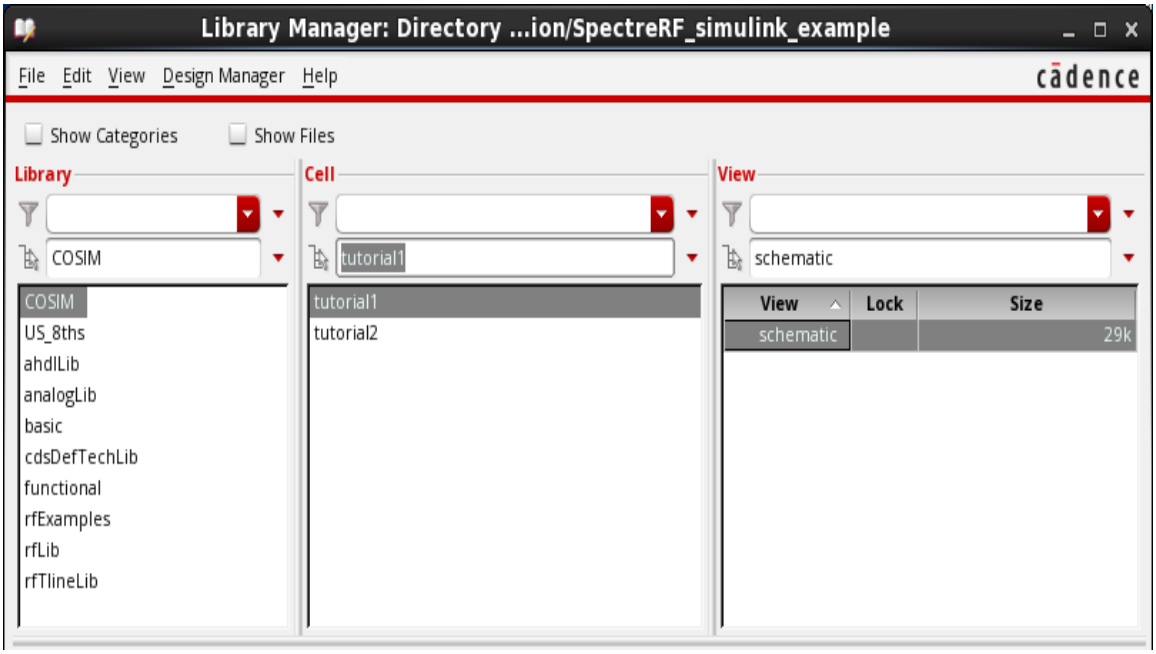
24. Ensure that you are in the `SpectreRF_simulink_example` directory. If not, then navigate to this directory, using below command.

25. From this directory, launch '`virtuoso`':

```
Unix> virtuoso &
```

26. From the `Library Manager`, navigate to `COSIM -> tutorial1`, and open its '`schematic`' view, as shown in Figure 28.

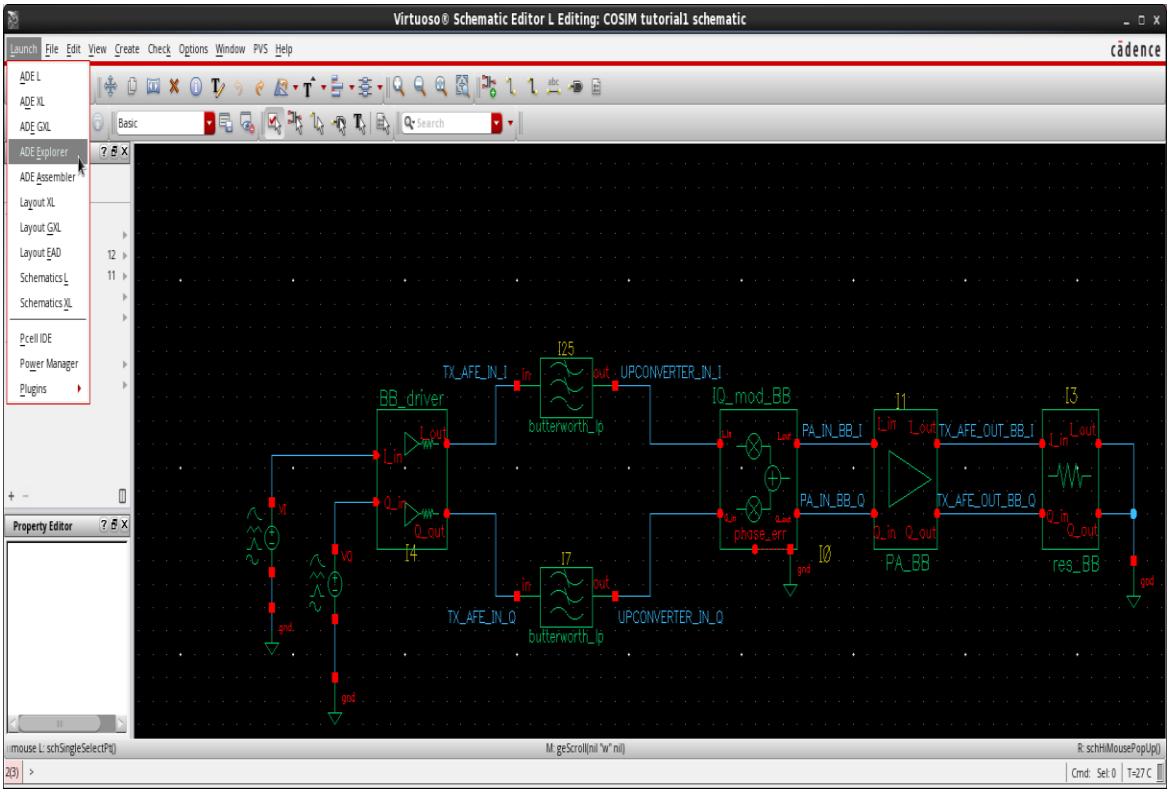
Figure 28: Library Manager



The `schematic` view is opened.

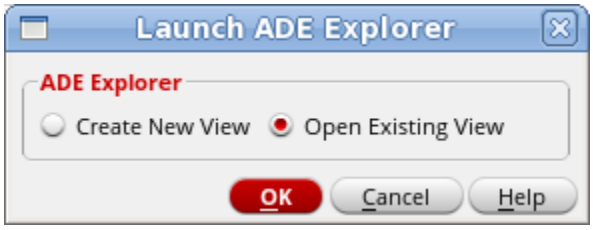
27. From the `schematic`, navigate to `Launch -> ADE Explorer` option, as shown in Figure 29.

Figure 29: 'schematic' of COSIM tutorial1



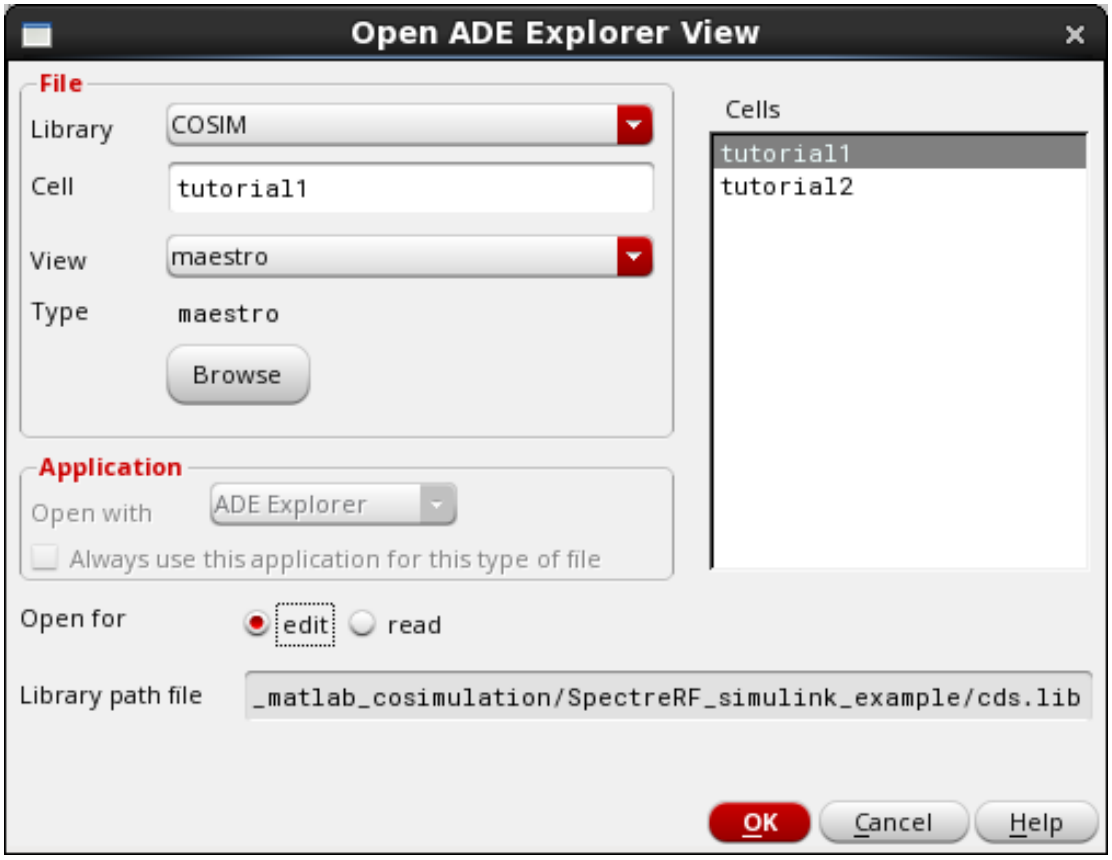
28. In the '`Launch ADE Explorer`' pop-up form as shown in Figure 30, choose '`Open Existing View`' and select '`maestro`' View of '`tutorial1`' Cell from `COSIM` Library, as shown in Figure 31.

Figure 30: Launch ADE Explorer form



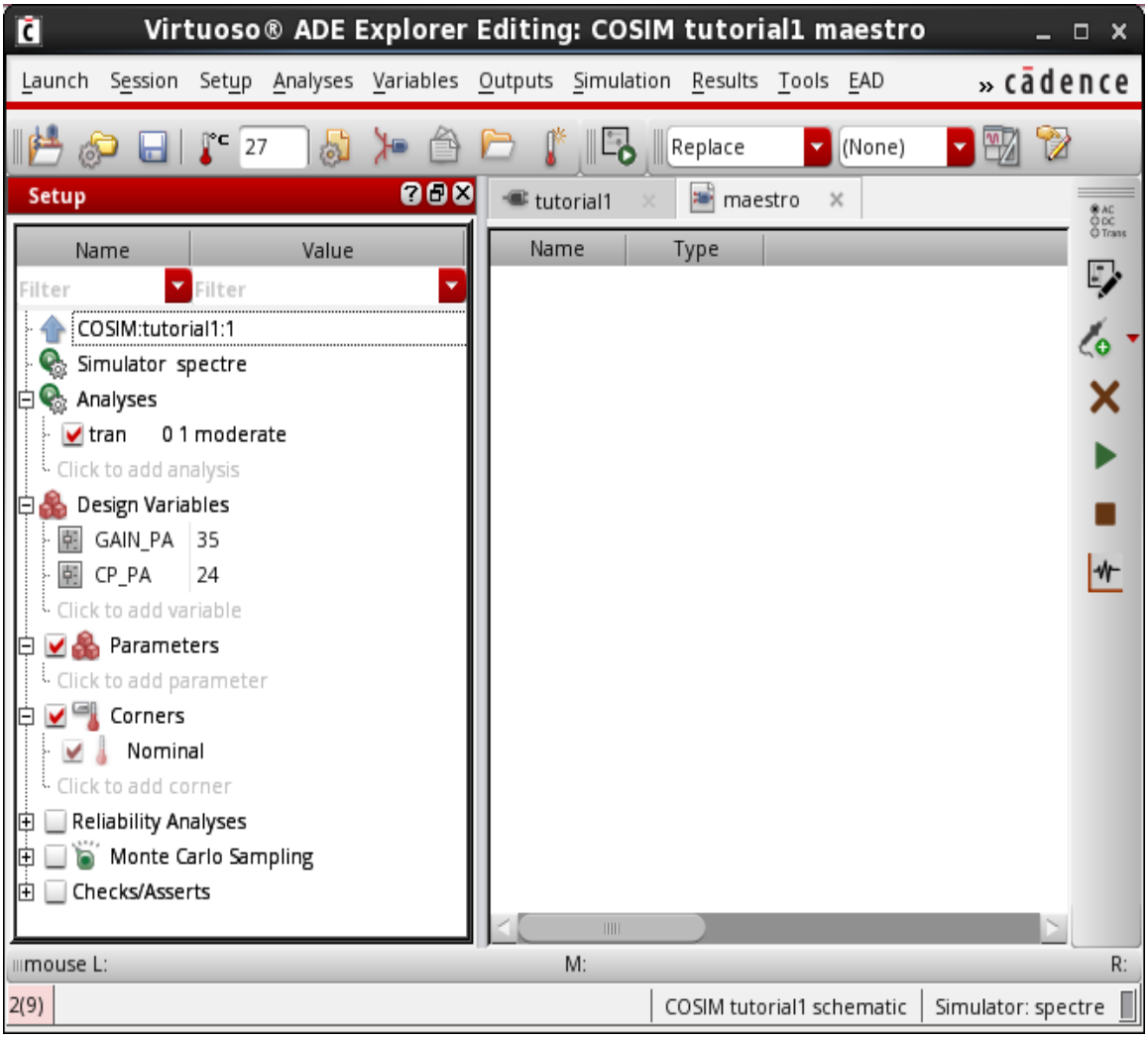
Click `OK` on this form.

Figure 31: Opening maestro view of tutorial1 cell from COSIM Library in ADE Explorer



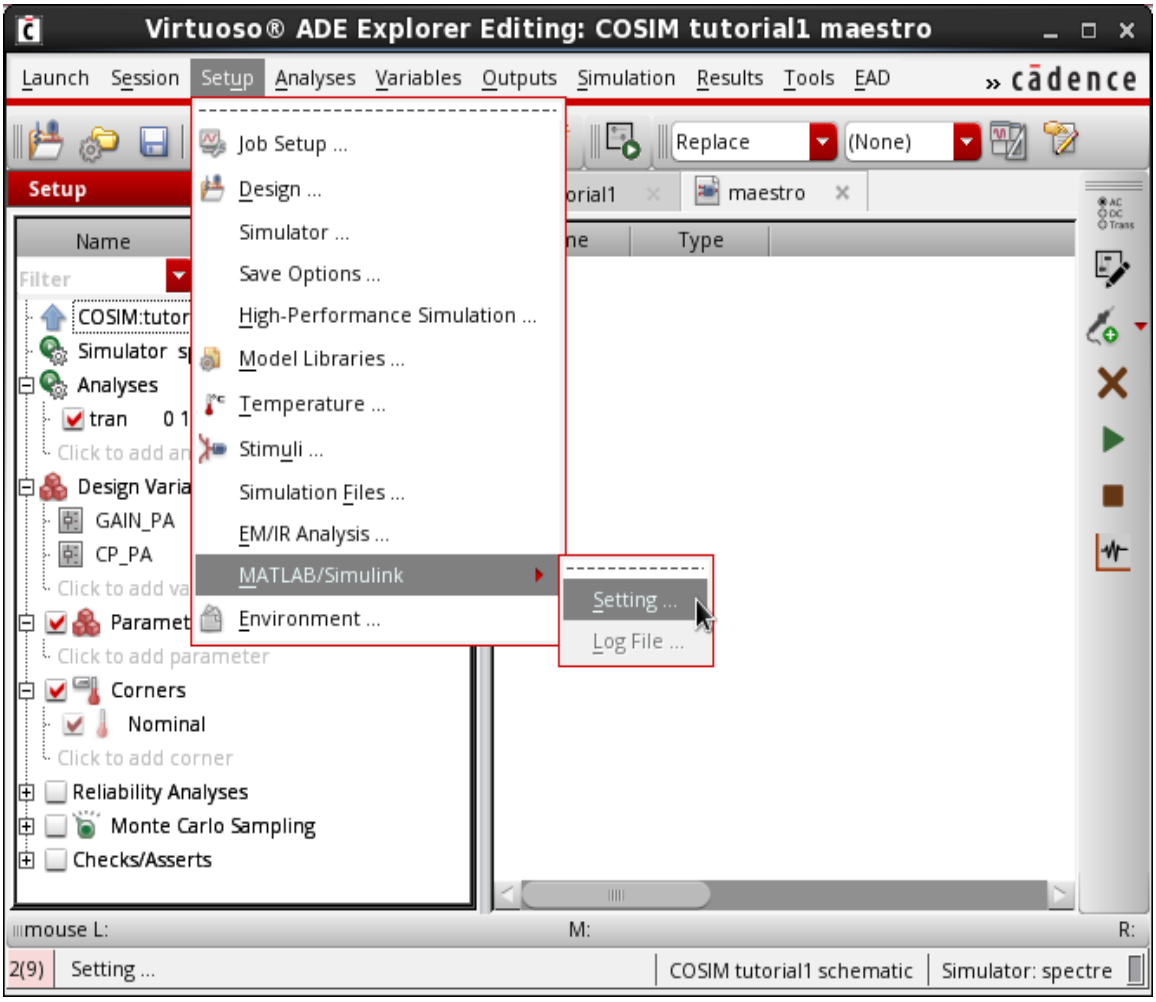
The corresponding '`maestro`' (ADE Explorer) view is opened, as shown in Figure 32.

Figure 32: ADE Explorer window after opening the maestro view of tutorial1 cell



29. From the `ADE Explorer` window, navigate to `Setup -> MATLAB/Simulink -> Setting…` option as shown in Figure 33.

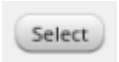
Figure 33: MATLAB/Simulink Settings



30. The '`Cosimulation Options`' form appears as shown in Figure 34.

Figure 34: Cosimulation Options form for Tutorial1

Here all the '`Cosimulation Options`' settings are pre-filled in the form, but the Cosimulation Options fields can be set using the following procedure.

a. In the `Cosimulation Options` form, click the '`Select`'  button located beside '`Cosimulation inputs`' field.

b. The corresponding '`schematic`' design view will be automatically opened, where you will see the following information below the schematic window.

   `Select source instance as cosimulation inputs. Press Esc when done.`

Select both the 'vsource' components, VI and VQ one after another, and then press the *Esc* key.

c. The corresponding 'Cosimulation inputs' field will be filled with an entry:

```
VI:wave VQ:wave
```

d. In the Cosimulation Options form, now click the 'Select' button located beside 'Cosimulation outputs'.

e. The corresponding 'schematic' view will be automatically opened, where you will see the following information below the schematic window.

```
Select Net/Terminal as cosimulation outputs. Press Esc
when done.
```

Select the TX_AFE_OUT_BB_I and TX_AFE_OUT_BB_Q nets one after the another, and then press the *Esc* key.

The corresponding 'Cosimulation outputs' field will be filled with an entry as mentioned:

```
TX_AFE_OUT_BB_I TX_AFE_OUT_BB_Q
```

f. Ensure that the value of 'Cosimulation socket port' field is the same as the port value of the SpectreRF Engine block defined in the previous 'Modifying the design and Setting up the cosimulation from MATLAB' section.

In this case, it is set to '38520'.

g. In the 'Cosimulation timeout' field, the value entered is the same as the 'Simulation response timeout (>30 secs)' field of SpectreRF Engine block defined in the previous 'Modifying the design and Setting up the cosimulation from MATLAB' section.

Hence, it is set to '60' seconds.

h. Keep the 'MATLAB start host' option set to default 'localhost'.

i. Under the 'ADDITIONAL SETTING', make sure to keep the 'Start MATLAB' option set to 'no' for this section. You may try the other 'Start MATLAB' options such as 'before Simulation' and 'now' depending on your current requirement.

Note: More details and usage of the 'ADDITIONAL SETTING' fields are described in the next section, 'Tutorial 2'.

j. At the bottom of this Cosimulation Options form, turn on the 'Enabled' checkbox.

   The final 'Cosimulation Options' form should look as shown above in Figure 34.

k. Click 'Apply' to apply the changes, and 'OK' to close the form.

31. Switch back to the 'maestro' tab, if you are on the 'tutorial1' schematic tab.

32. From the ADE Explorer, navigate to Analyses -> Choose… as shown in Figure 35. The 'Choosing Analyses - ADE Explorer' form opens as shown in Figure 36.

Figure 35: Choose Analyses

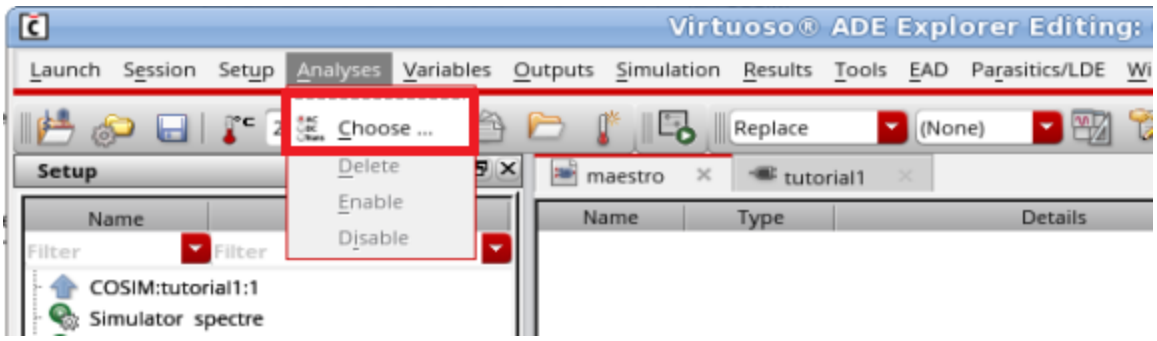
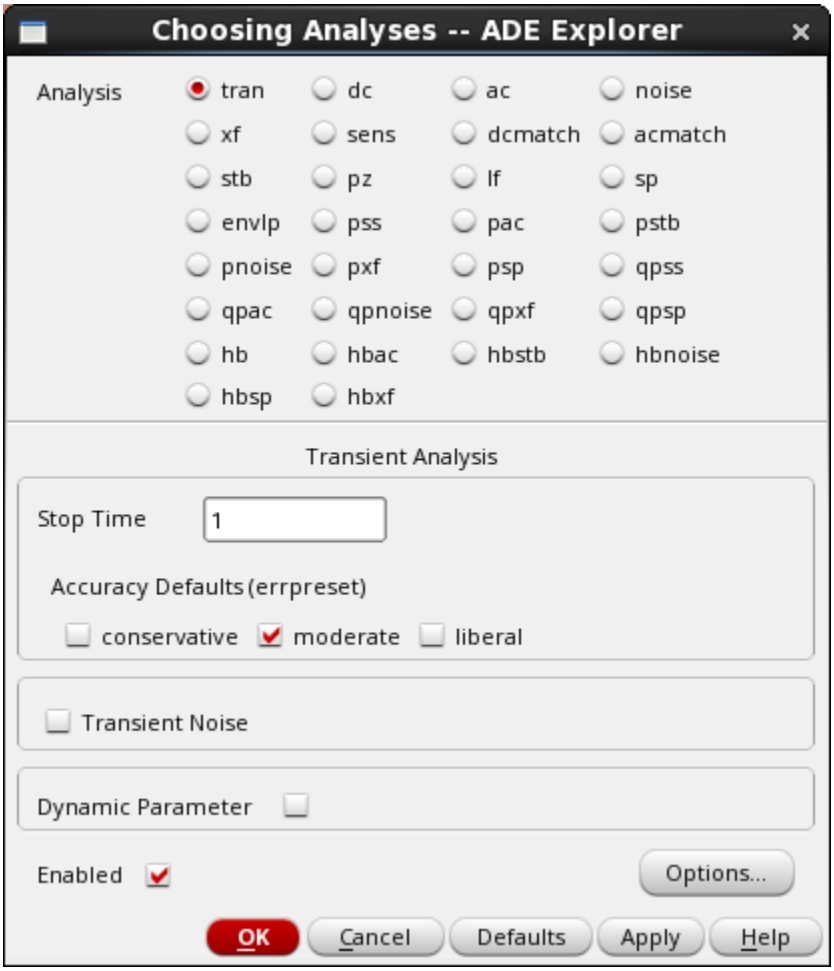


Figure 36: Setup for the transient analysis



33. Setup a '`tran`' analysis.

    a. The '`Stop Time`' can be any value. Here we have set it to '`1`' second. The SpectreRF simulator synchronizes the stop time with Simulink.

    b. The default accuracy is set to '`moderate`'.

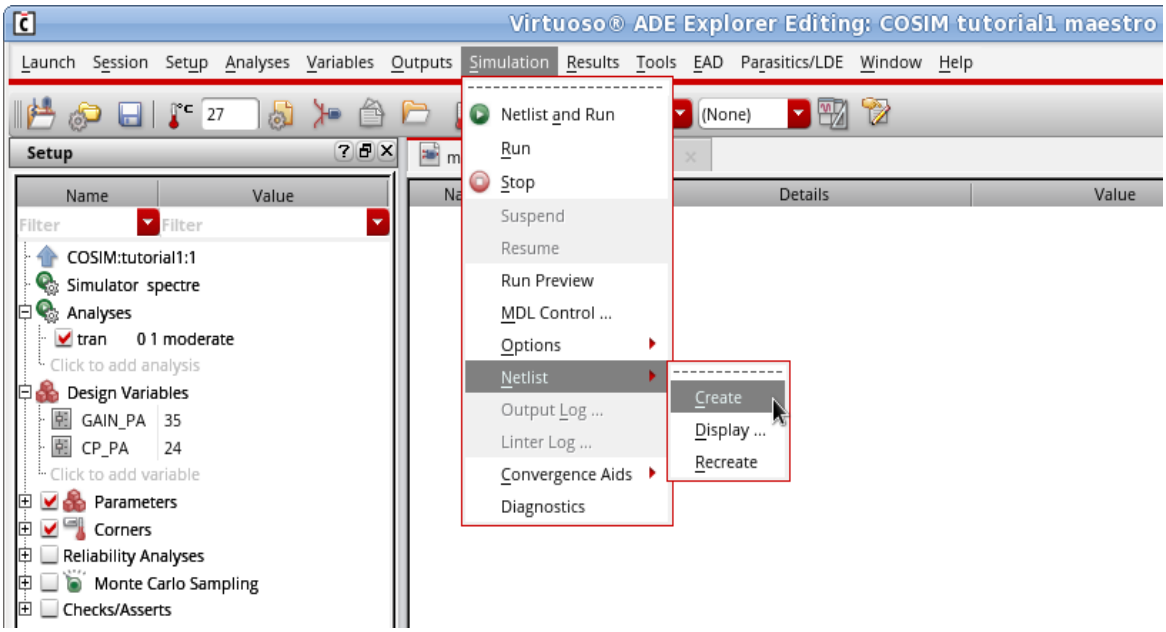
    c. Ensure that the '`Enabled`' checkbox at the bottom of this form is selected. The final '`Choosing Analyses`' window will look as shown in Figure 36.

    d. Click '`OK`' on this form.

34. Ensure that the `Design Variables`, '`GAIN_PA`' and '`CP_PA`' are assigned a default value of '`35`' and '`24`' respectively in ADE as shown in Figure 37.

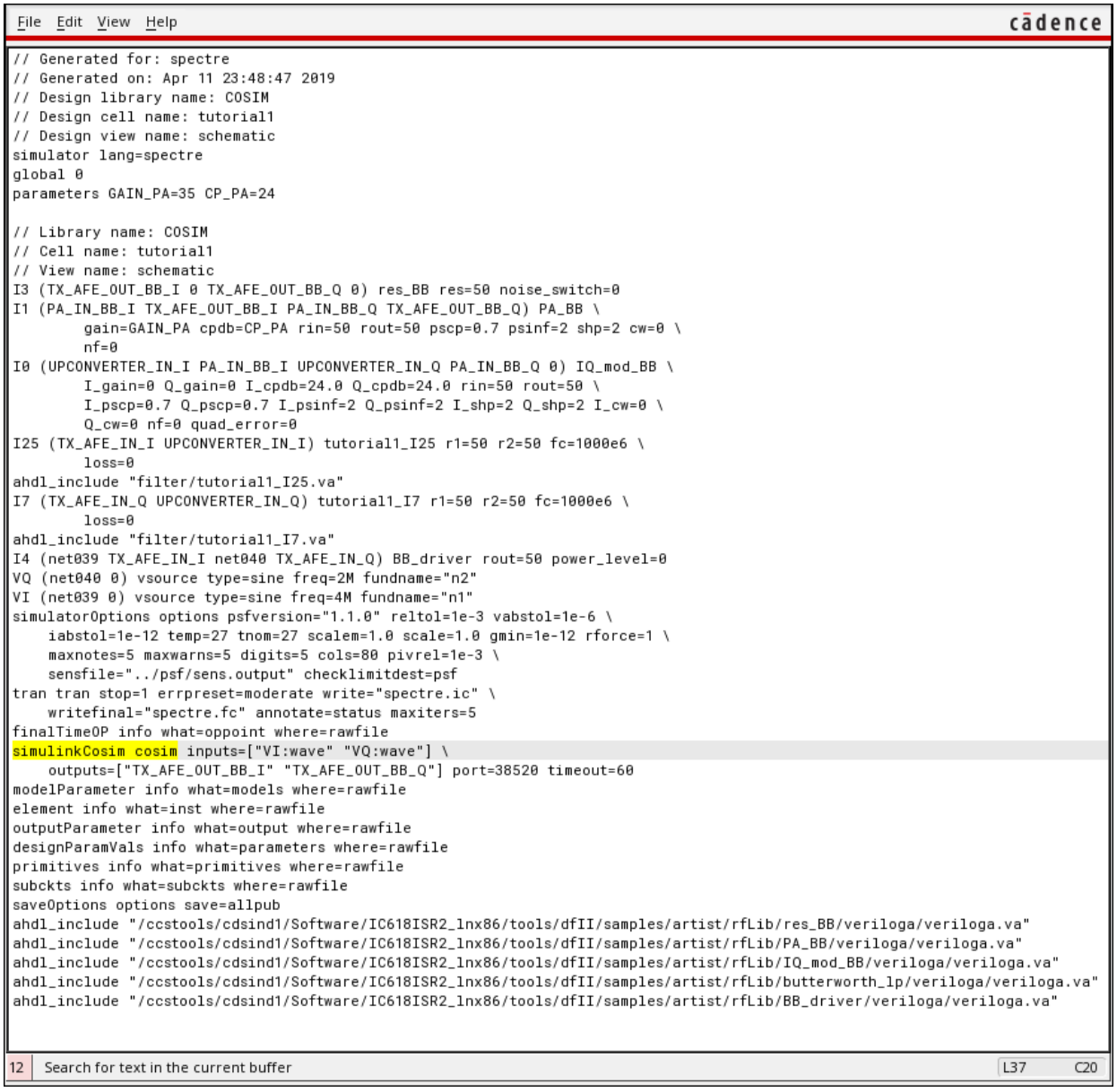
35. From the `ADE Explorer (maestro)` view, navigate to `Simulation -> Netlist -> Create`, as shown in Figure 37.

Figure 37: Netlist generation for transient simulation



36. Make sure the generated netlist has the following `simulinkCosim cosim` statement appearing in it, as shown in Figure 38.

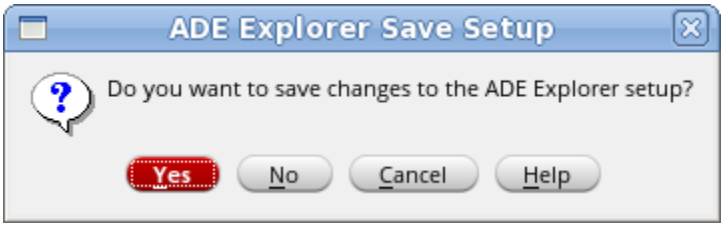
Figure 38: Netlist containing the cosimulation statement



37. Close the netlist generated window and corresponding '`maestro`' view.

38. Save the changes to your maestro view by clicking '`Yes`' on the 'ADE Explorer Save Setup' form that appears as shown in Figure 39.

Figure 39: ADE Explorer Save Setup form



This step completes the section of '`Setting up the cosimulation from ADE`'.

Setting up the cosimulation from Standalone SpectreRF

39. If you have already generated a netlist using ADE, skip this section and go directly to the next section, '`Running the Cosimulation with SpectreRF`'.

    The netlist used in this tutorial is based on the behavioral RF Front-End transmitter model.

40. Assuming you are in the '`SpectreRF_simulink_example`' directory, from the above section, navigate to the '`tutorial1`' directory inside it, as shown.

    ```
    Unix> cd tutorial1
    ```

41. Inside '`tutorial1`' directory, you would see the behavioral verilogA modules for each of the SpectreRF front-end transmitter model, along with the `tutorial1.scs` netlist.

    ```
    [pashish@noi-pashish tutorial1]$ ls
    BB_driver  butterworth_lp  filter  IQ_mod_BB  PA_BB  res_BB  tutorial1.scs
    ```

49. Open the netlist file `tutorial1/tutorial1.scs` in edit mode.

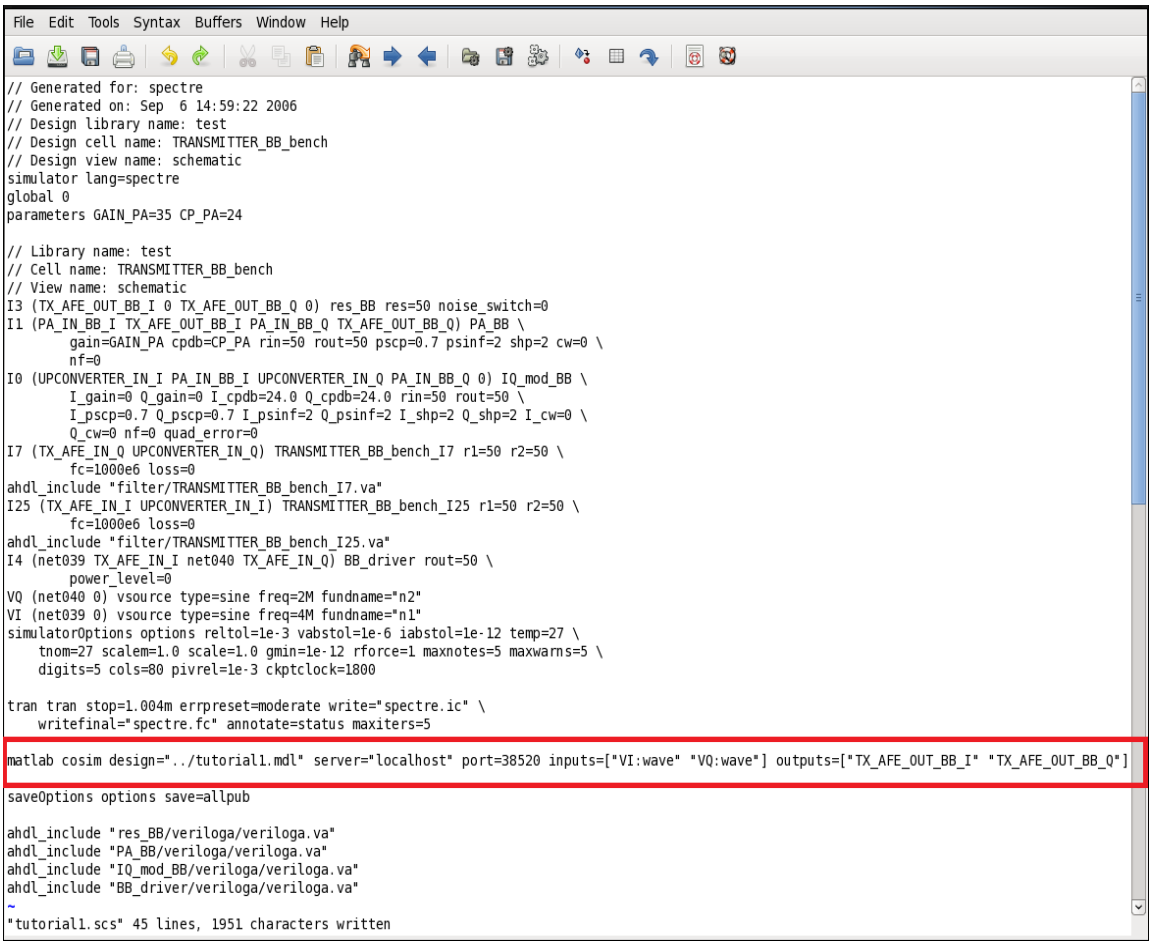
50. Add a '`matlab cosim`' statement (if it is not already present), after the transient analysis declaration inside the netlist, as shown.

    For example,

    ```
    matlab cosim design="../tutorial1.mdl" server="localhost"
    port=38520 inputs=["VI:wave" "VQ:wave"]
    outputs=["TX_AFE_OUT_BB_I" "TX_AFE_OUT_BB_Q"]
    ```

51. The updated netlist, `tutorial1.scs` will now look as shown in Figure 40.

Figure 40: tutorial1.scs netlist



```
File  Edit  Tools  Syntax  Buffers  Window  Help
// Generated for: spectre
// Generated on: Sep  6 14:59:22 2006
// Design library name: test
// Design cell name: TRANSMITTER_BB_bench
// Design view name: schematic
simulator lang=spectre
global 0
parameters GAIN_PA=35 CP_PA=24

// Library name: test
// Cell name: TRANSMITTER_BB_bench
// View name: schematic
I3 (TX_AFE_OUT_BB_I 0 TX_AFE_OUT_BB_Q 0) res_BB res=50 noise_switch=0
I1 (PA_IN_BB_I TX_AFE_OUT_BB_I PA_IN_BB_Q TX_AFE_OUT_BB_Q) PA_BB \
        gain=GAIN_PA cpdb=CP_PA rin=50 rout=50 pscp=0.7 psinf=2 shp=2 cw=0 \
        nf=0
I0 (UPCONVERTER_IN_I PA_IN_BB_I UPCONVERTER_IN_Q PA_IN_BB_Q 0) IQ_mod_BB \
        I_gain=0 Q_gain=0 I_cpdb=24.0 Q_cpdb=24.0 rin=50 rout=50 \
        I_pscp=0.7 Q_pscp=0.7 I_psinf=2 Q_psinf=2 I_shp=2 Q_shp=2 I_cw=0 \
        Q_cw=0 nf=0 quad_error=0
I7 (TX_AFE_IN_Q UPCONVERTER_IN_Q) TRANSMITTER_BB_bench_I7 r1=50 r2=50 \
        fc=1000e6 loss=0
ahdl_include "filter/TRANSMITTER_BB_bench_I7.va"
I25 (TX_AFE_IN_I UPCONVERTER_IN_I) TRANSMITTER_BB_bench_I25 r1=50 r2=50 \
        fc=1000e6 loss=0
ahdl_include "filter/TRANSMITTER_BB_bench_I25.va"
I4 (net039 TX_AFE_IN_I net040 TX_AFE_IN_Q) BB_driver rout=50 \
        power_level=0
VQ (net040 0) vsource type=sine freq=2M fundname="n2"
VI (net039 0) vsource type=sine freq=4M fundname="n1"
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 ckptclock=1800

tran tran stop=1.004m errpreset=moderate write="spectre.ic" \
    writefinal="spectre.fc" annotate=status maxiters=5

matlab cosim design="../tutorial1.mdl" server="localhost" port=38520 inputs=["VI:wave" "VQ:wave"] outputs=["TX_AFE_OUT_BB_I" "TX_AFE_OUT_BB_Q"]

saveOptions options save=allpub

ahdl_include "res_BB/veriloga/veriloga.va"
ahdl_include "PA_BB/veriloga/veriloga.va"
ahdl_include "IQ_mod_BB/veriloga/veriloga.va"
ahdl_include "BB_driver/veriloga/veriloga.va"
~
"tutorial1.scs" 45 lines, 1951 characters written
```

The parameters for the '`matlab cosim`' statement used for cosimulation are described below in Table 2.

Table 2: Virtuoso ADE MATLAB cosimulation parameters and their details

Parameter	Meaning
`design`	MATLAB/Simulink design name. Basically, it refers to the file, Simulink associates with the netlist. For example, `design="tutoria1.mdl"`

	means the testbench whose file is tutorial1.mdl
`inputs=[…]`	Input vector to identify the flow from MATLAB to SpectreRF Engine. The format is `["instance_name:wave" "instance_name1:wave" …]` where, `wave` is a keyword. **Note:** The sequence of instances in square brackets follows the same one as the SpectreRF Engine port labels. This label information can be found by double-clicking the `SpectreRF Engine` and then by enabling the `'Show engine port labels'` checkbox at the bottom, in the Block Parameters of the SpectreRF Engine, as shown.  Once the inputs are set in the netlist, the sources (such as `VI` and `VQ`, shown in Step 2 of this section) associated with the inputs are meaningless and their parameters (such as `type=sine freq=2M fundname="n2"` for `VQ` source) do not affect the simulation.
`outputs=[…]`	Output vector from SpectreRF Engine to MATLAB. It can be the **net name** or **terminal name** of interest. **Note:** If the net name is selected, only voltage is given; If terminal name is set, only current is output. To get current in an envelope (`envlp`) analysis, you must add probes in the topology.
`port`	The one on the server running MATLAB. Its value should be the same as the value in the SpectreRF Engine in your current MATLAB design. For example, port=38520 is being used here as the co-simulink listen port.

server	The name of the machine that MATLAB starts. It can be set with the machine name or with the IP address of that machine. The accepted form is server = "155.110.110.110" and server = "bj2lnx20" server = "localhost" means that Spectre and MATLAB use the same machine.
silent	Tells MATLAB whether to open the window during simulation. If silent=yes, the testbench window is opened. Basically, it launches MATLAB with parameter -nodesktop. Its possible values are no and yes.
timeout	The period of time MATLAB spends waiting for a response from the Spectre simulator. timeout=60 s stops MATLAB if it does not receive any response. i.e. it is the socket timeout in seconds. Default value is 60 seconds.
ratio	It's the Hold time ratio between two sample points. Default value is 0.

After performing the above steps from this section, 'Setting up the cosimulation from Standalone SpectreRF', the netlist, tutorial1.scs is now ready for cosimulation.

Running the cosimulation

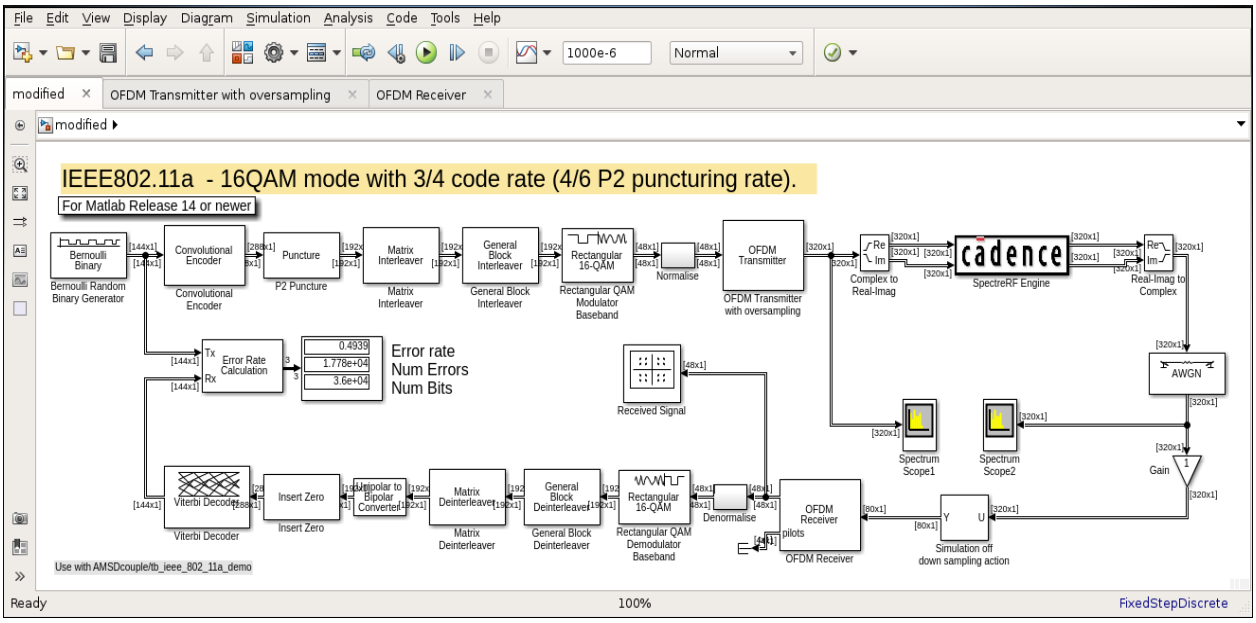
52. Open the 'unmodified' testbench for 'LAN 802.11' block from 'tutorial1' of 'spectrerf_demo' Library, having the SpectreRF Engine component instantiated for running the cosimulation, if you have not closed the already opened Matlab session from previous section, 'Modifying the design and Setting up the cosimulation from MATLAB'.

If you have closed Matlab from previous section, ensure that you are in the '`SpectreRF_simulink_example`' directory, and launch '`matlab`' from there, using the below command.

```
Unix> matlab&
```

Note: You can use the testbench just saved from the above section or use the already available '`modified`' testbench of `LAN 802.11` block from '`tutorial1`' Simulink library having the `SpectreRF Engine` component, for running the cosimulation. Figure 41 shows a snapshot of the '`unmodified`' testbench to be used in the following steps.

Figure 41: 'unmodified' tutorial1 testbench for running cosimulation



53. From the opened '`unmodified`' `tutorial1` testbench of MATLAB design window, click `Simulation -> Run` or hit the green Run button directly from the MATLAB window.

The MATLAB desktop issues the following message:

```
block 'unmodified/SpectreRF Engine': (COSIM_OK) Waiting for
incoming connection on port 38520, timeout: 60 sec ...
```

The above message can be seen by navigating from the MATLAB window to `View -> Diagnostic Viewer`.

54. Immediately after pressing the  button, move to Step 55.

    Note: The time interval between `Step 53` and `Step 54` must be within the `Simulation response timeout` value defined as '`Block Parameters`' of the `SpectreRF Engine` component (you may check by double-clicking the `SpectreRF Engine` block).

    Here, in this case, it is '`60`' seconds.

55. You may run the simulation from **ADE** or from the Unix **command-line**, depending on your interest.

A. From ADE

To run the simulation from `ADE`, please ensure that the corresponding '`maestro`' view of `COSIM -> tutorial1` schematic cell is already opened from the previous '`Setting up the cosimulation from ADE`' section.

i. If not, then launch `virtuoso` first from the '`SpectreRF_simulink_example`' directory:

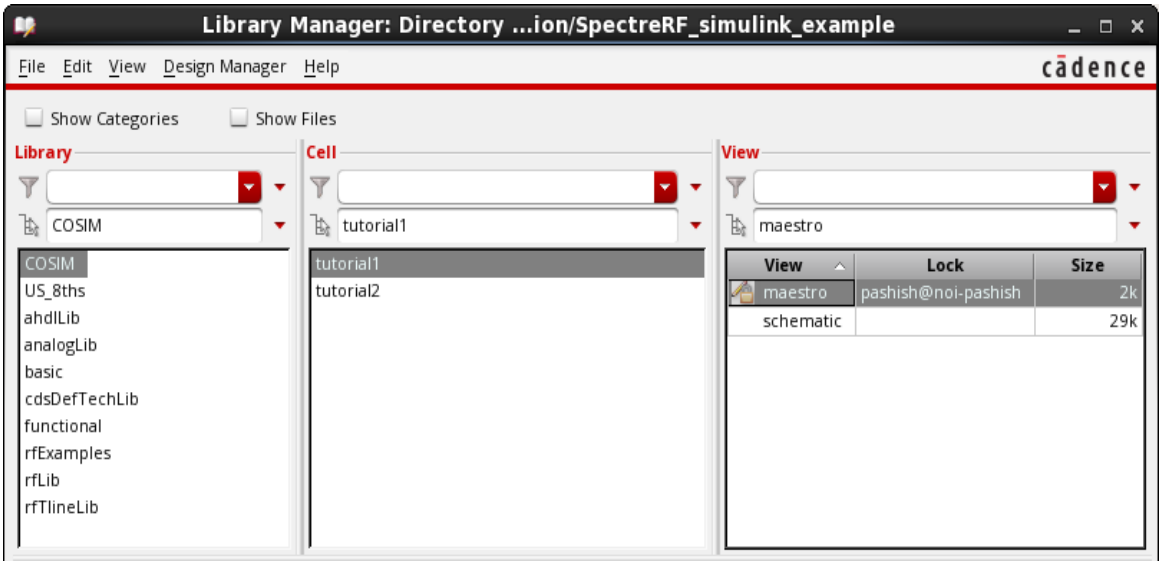
```
Unix> virtuoso &
```

ii. Then navigate from the `Library Manager` to

```
Library -> COSIM

Cell -> tutorial1

View -> maestro
```

as shown in Figure 42.

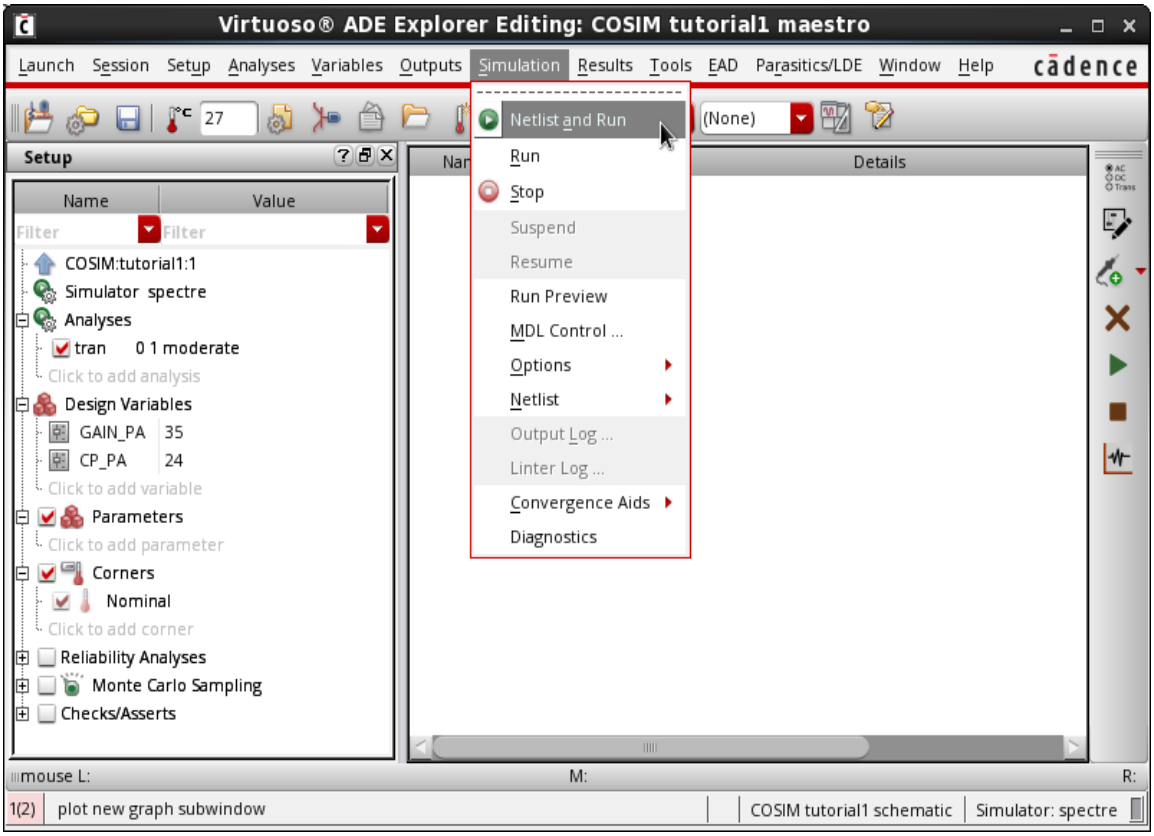
Figure 42: Library Manager for COSIM tutorial1 testbench



The corresponding `ADE Explorer` window with `maestro` view opens.

iii. Hit the green 'Run Simulation'  button from the right edge of `ADE Explorer` or click on `Simulation -> Netlist and Run` option from menu bar, as shown in Figure 43.

Figure 43: Running cosimulation from ADE



B. From Standalone SpectreRF

i. To run the simulation from Unix `command-line`, please navigate from your current working '`SpectreRF_simulink_example`' directory to the '`tutorial1`' directory.

ii. Run the simulation using the '`spectre`' run command with the `tutorial1.scs` netlist available (which you have modified from the previous section, '`Setting up the cosimulation from Standalone SpectreRF`', having the '`matlab cosim`' statement inside it), as shown.

```
Unix> spectre tutorial1.scs
```

iii. The cosimulation immediately begins. The MATLAB desktop issues a message like the following, when simulation ends.

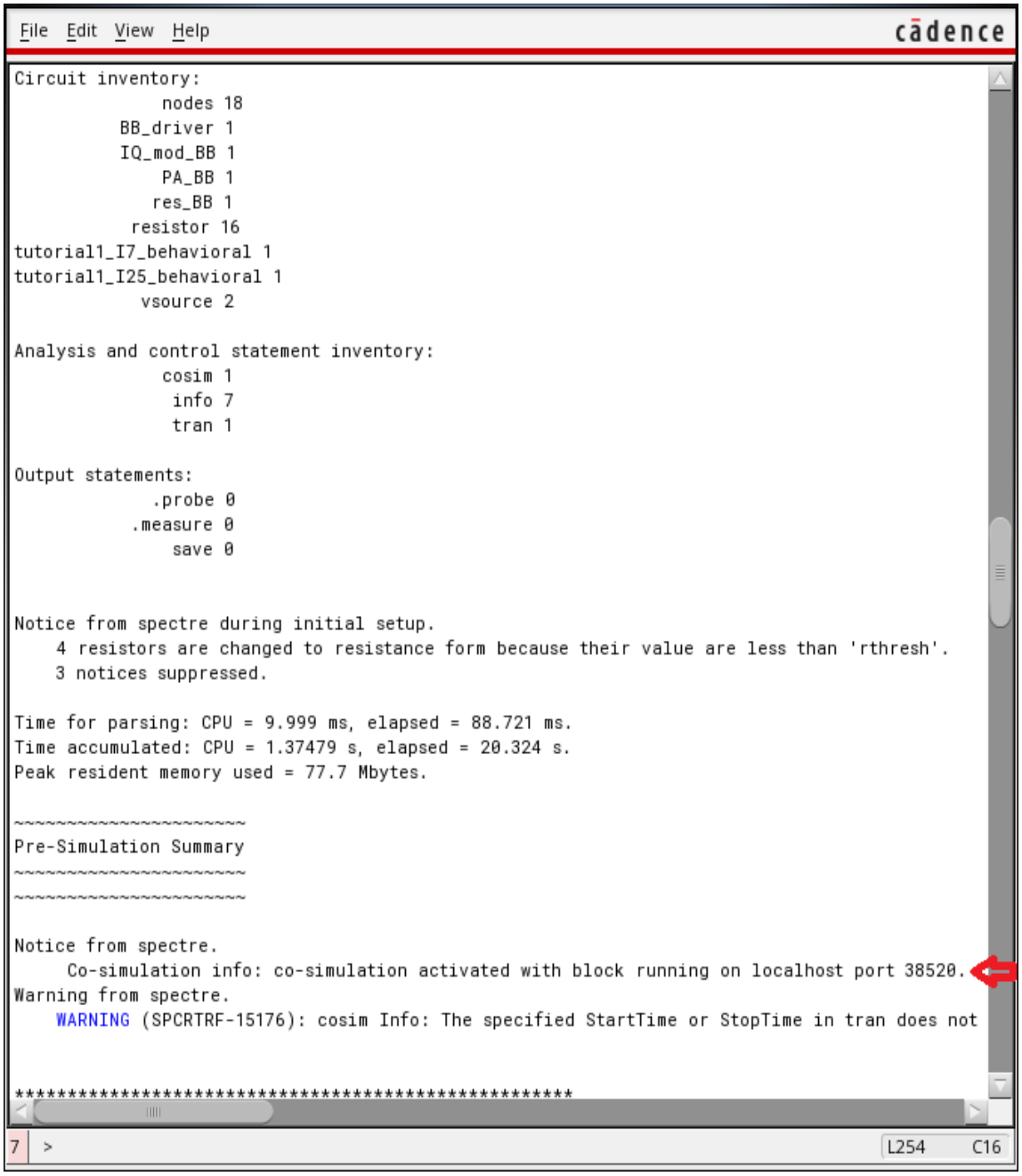
```
block 'unmodified/SpectreRF Engine': (COSIM_OK)
Simulation finished
```

The above message can be seen by navigating from the MATLAB window to `View -> Diagnostic Viewer`.

56. After running the simulation from `ADE` or through Unix `command-line`, you would see a message as shown in Figure 44 (this is from ADE) generated in spectre output log, that ensures the cosimulation has happened successfully:

```
Co-simulation info: co-simulation activated with block
running on localhost port 38520.
```

Figure 44: Generated spectre.out log after cosimulation



57. The MATLAB result plots are generated from `Spectrum Scope1` and `Spectrum Scope2` as shown in Figure 45 and Figure 46.

Note: You may need to use the 'Zoom-In'  feature available in the below Matlab plots to see the required region of interest.

Figure 45: Spectrum Scope1

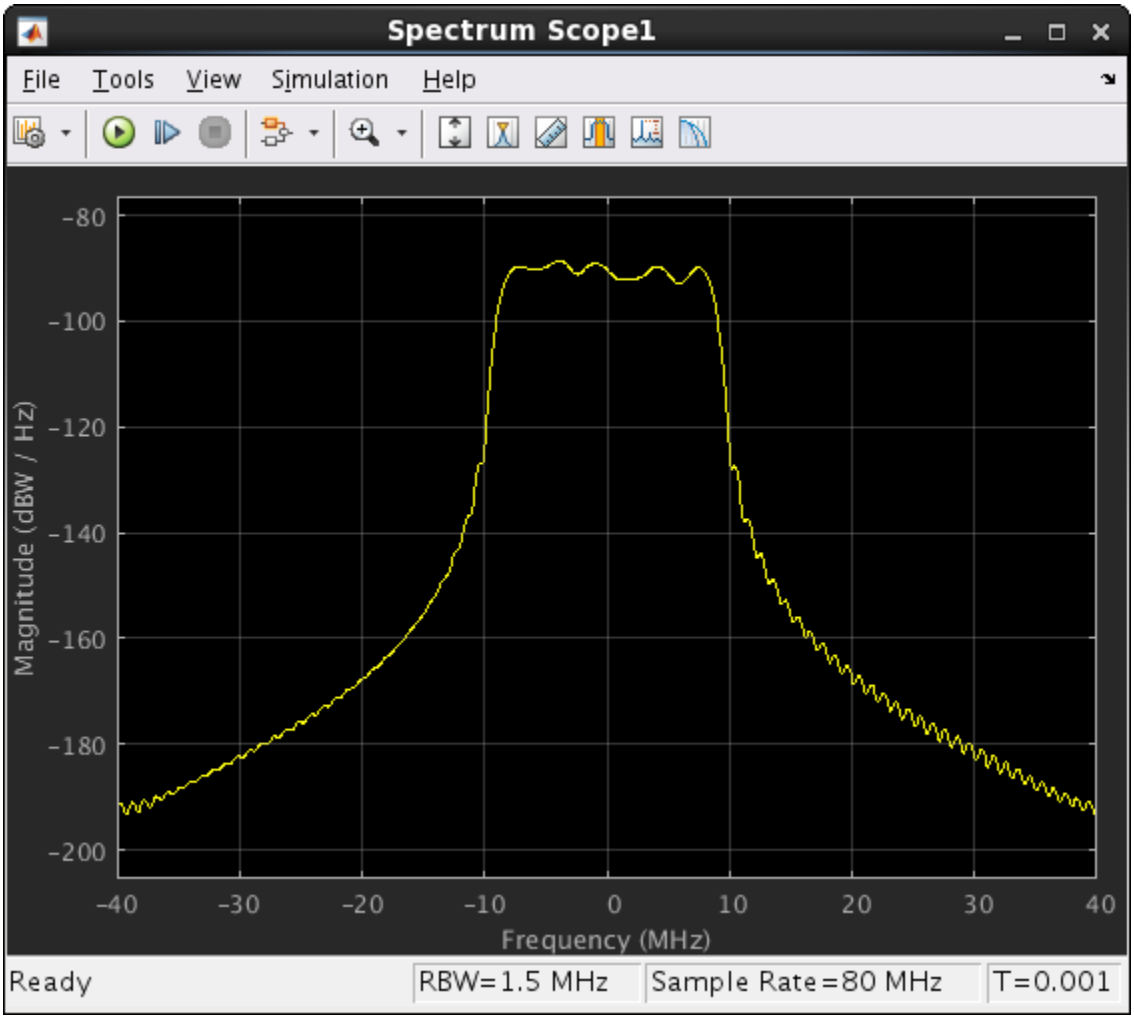
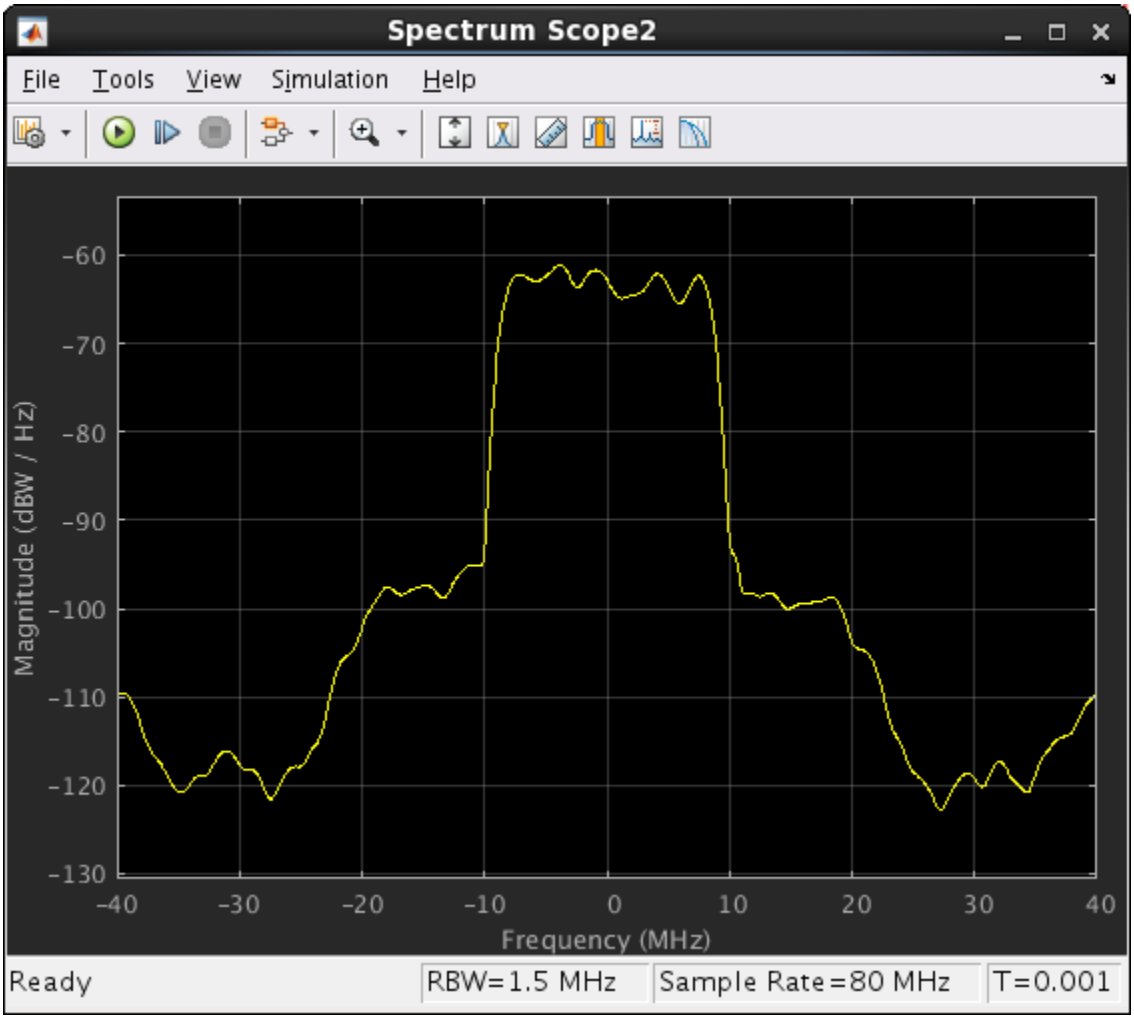


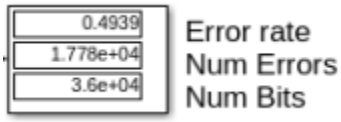
Figure 46: Spectrum Scope2



58. After the `Spectre` and `Matlab` cosimulation finishes, review the MATLAB/Simulink output.

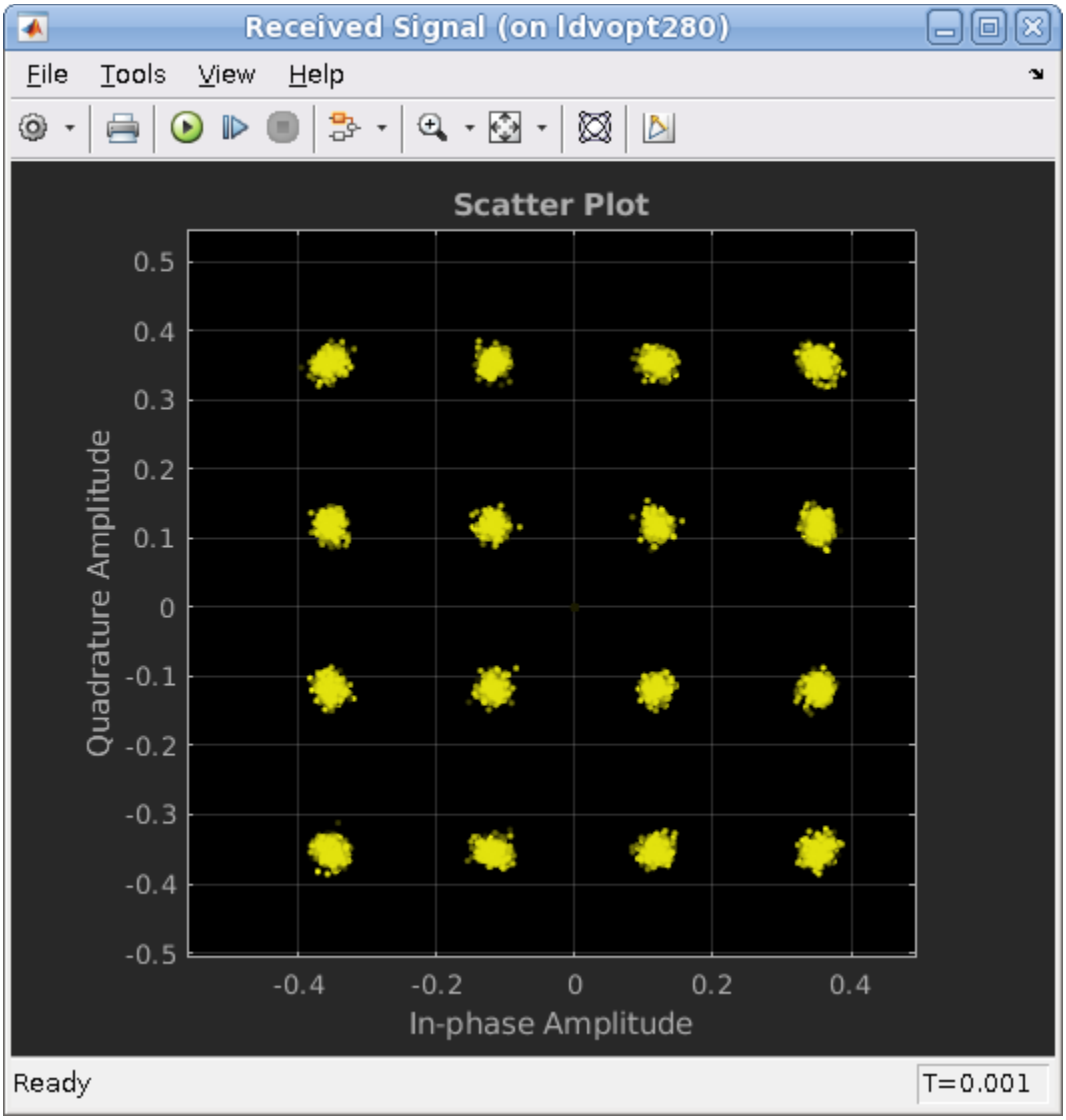
    Bit error does occur with these settings as shown in Figure 47. The input spectrum and output spectrum are different because the RF transmitter chain is non-ideal.

Figure 47: Bit Error Rate (BER) after cosimulation



The results are captured in the `Scatter Plot` (`Received Signal`), (also known as Constellation Diagram) as shown in Figure 48 with design parameters `GAIN_PA=35` and `CP_PA=24`.

Figure 48: Scatter Plot (Received Signal) after cosimulation



59. Close all the MATLAB graphs, its associated `'unmodified' or 'modified' LAN 802.11 tutorial1` testbench, and exit from Matlab environment.

    Save the MATLAB testbench workspace if any changes done.

60. Close the associated '`Virtuoso`' `ADE Explorer` '`maestro`' and '`schematic`' view of `COSIM tutorial1`, (`SAVE` the changes if any), and exit from your

virtuoso session, by navigating from the Virtuoso Command Interpreter Window (CIW) to `File -> Exit…` as shown in Figure 49.

Figure 49: Exit from Virtuoso session

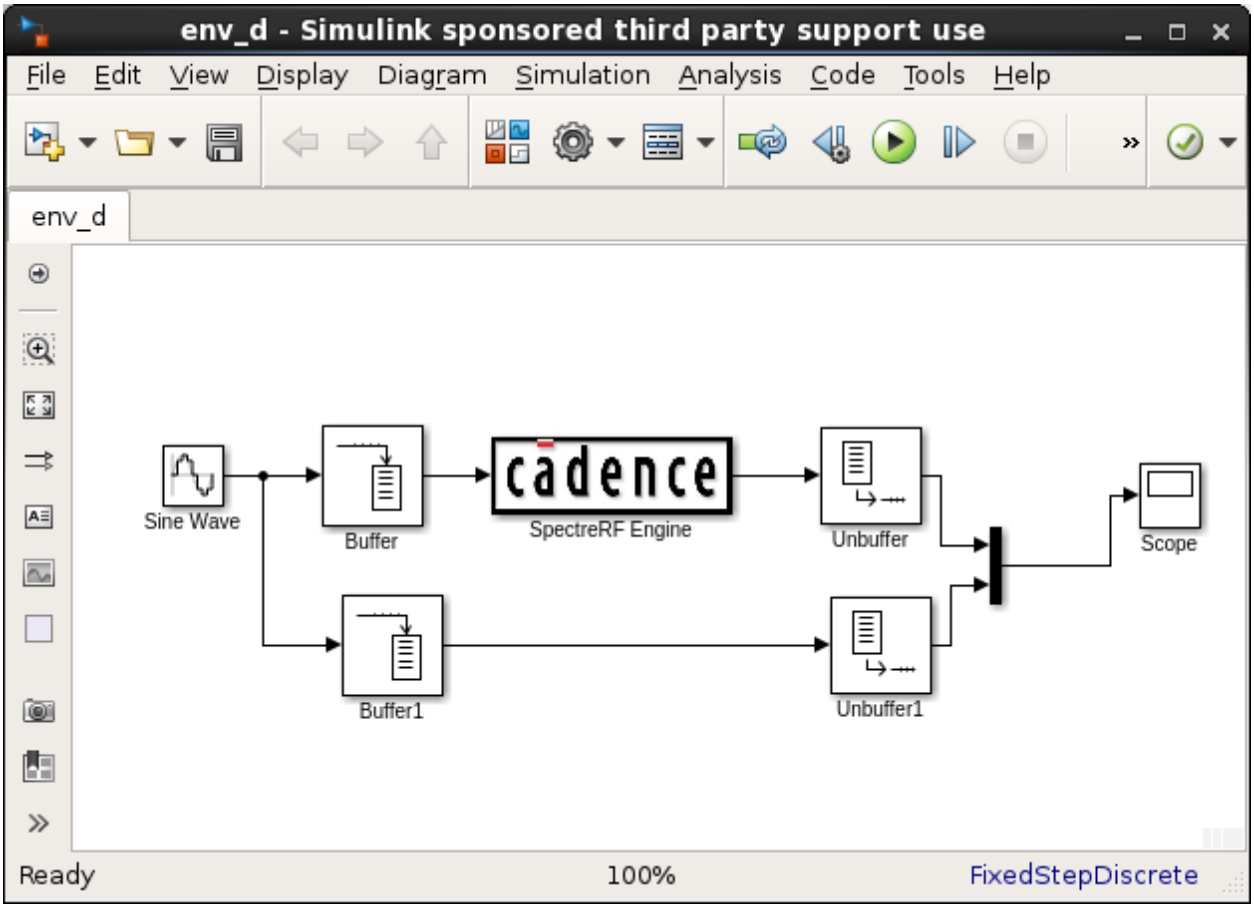
`Tutorial1` shows the basic steps involved in setting-up and running a SpectreRF <-> MATLAB cosimulation. In this tutorial, you start with two applications separately and then use a transient analysis to solve the circuit.

The next tutorial illustrates additional use modes and shows how to use an envelope `(envlp)` analysis in SpectreRF simulator co-simulating with Matlab/Simulink.

Tutorial 2

`Tutorial 2` demonstrates Matlab/Simulink and SpectreRF cosimulation, while the SpectreRF simulator runs an envelope (`envlp`) analysis. The MATLAB schematic, named as '`env_d`', used in this tutorial is shown in Figure 50, in which the '`Sine Wave`' block outputs a sine wave into two chains.

Figure 50: 'env_d' schematic testbench of Tutorial2 used for envelope analysis in cosimulation



The top chain in the above MATLAB schematic contains `Buffer`, `SpectreRF Engine`, and `Unbuffer` components. The bottom chain contains only `Buffer1` and `Unbuffer1` components, that provides the reference for the top chain.

The results are displayed in the `Scope` component.

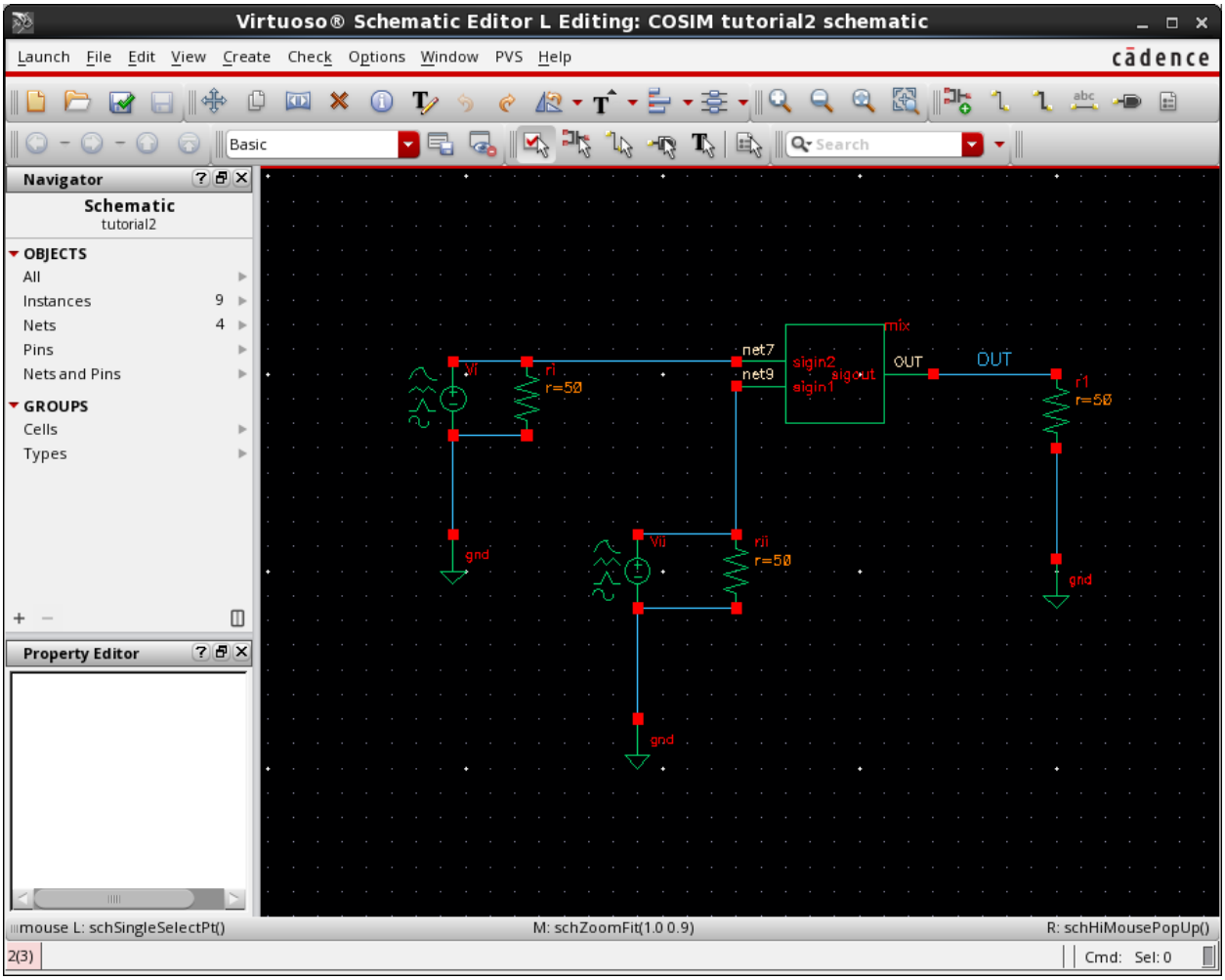
In this Simulink testbench in `Tutorial2`, if you see that by default some of the block names are hidden, please follow the steps to display all the block names in your Simulink model:

a. Click on the "`Display`" tab on the menu options at the top of the Simulink model editor.

b. Then uncheck the "`Hide Automatic Names`" option.

In this tutorial the `SpectreRF Engine` component instantiated inside Simulink testbench performs the mixing function. Figure 51 shows the schematic view of the tutorial2 cell from the COSIM library.

Figure 51: tutorial2 schematic of mixing circuit for modulating signal



The low frequency signal in MATLAB is fed into the SpectreRF modulating circuit, mixed there with a high frequency signal, and then output back to MATLAB. With the envelope `(envlp)` analysis, the envelope is obtained at the terminal of interest from MATLAB.

Tutorial 2 illustrates the following use mode:

a. First use Mode:

Manually running `Spectre` simulation from `ADE` and calling `MATLAB` from ADE.

b. Second use Mode:

Manually running `Matlab/Simulink` and activating `Spectre` simulation through Simulink.

Each of these modes is described in detail in the following sections.

First use Mode

Setting up the cosimulation from ADE

If you are using `SpectreRF` standalone, skip this section and go the next "`Second use Mode (Setting up and Running the cosimulation from Standalone SpectreRF)`" section.

1. Ensure that you are in the '`SpectreRF_simulink_`example' directory, as shown.

   ```
   Unix> pwd

   /…/home/…/RAK/SpectreRF_simulink_example
   ```

   If you are not in this directory, then navigate to the same.

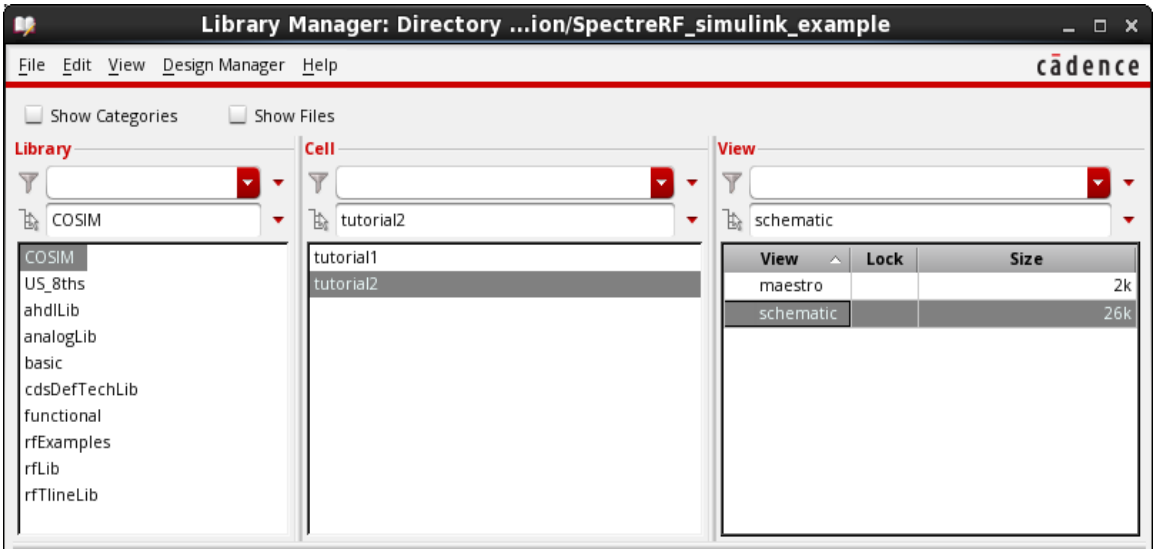
2. From this directory, launch `virtuoso`, using the below command.

   ```
   Unix> virtuoso &
   ```

3. From the `Library Manager`, navigate to `COSIM -> tutorial2`, and open its '`schematic`' view, as shown in Figure 52.

   ```
   Library -> COSIM

   Cell -> tutorial2

   View -> schematic
   ```

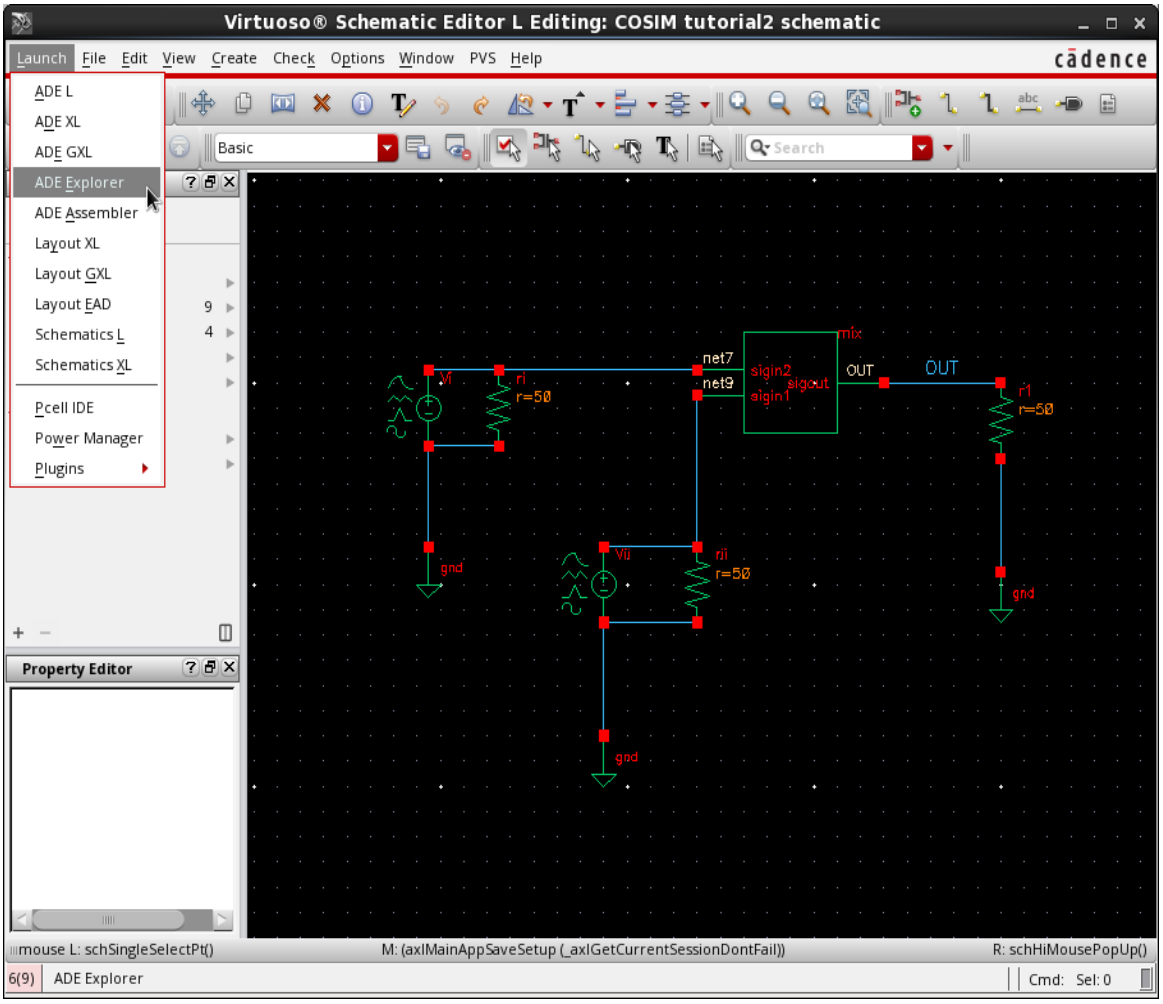
Figure 52: Opening schematic view of tutorial2 cell in COSIM library from Library Manager



The `schematic` view is opened.

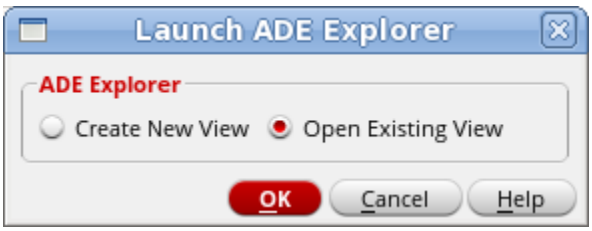
4. From the schematic window, navigate to `Launch -> ADE Explorer` option, as shown in Figure 53.

Figure 53: Launch ADE Explorer from 'schematic' of tutorial2 testbench



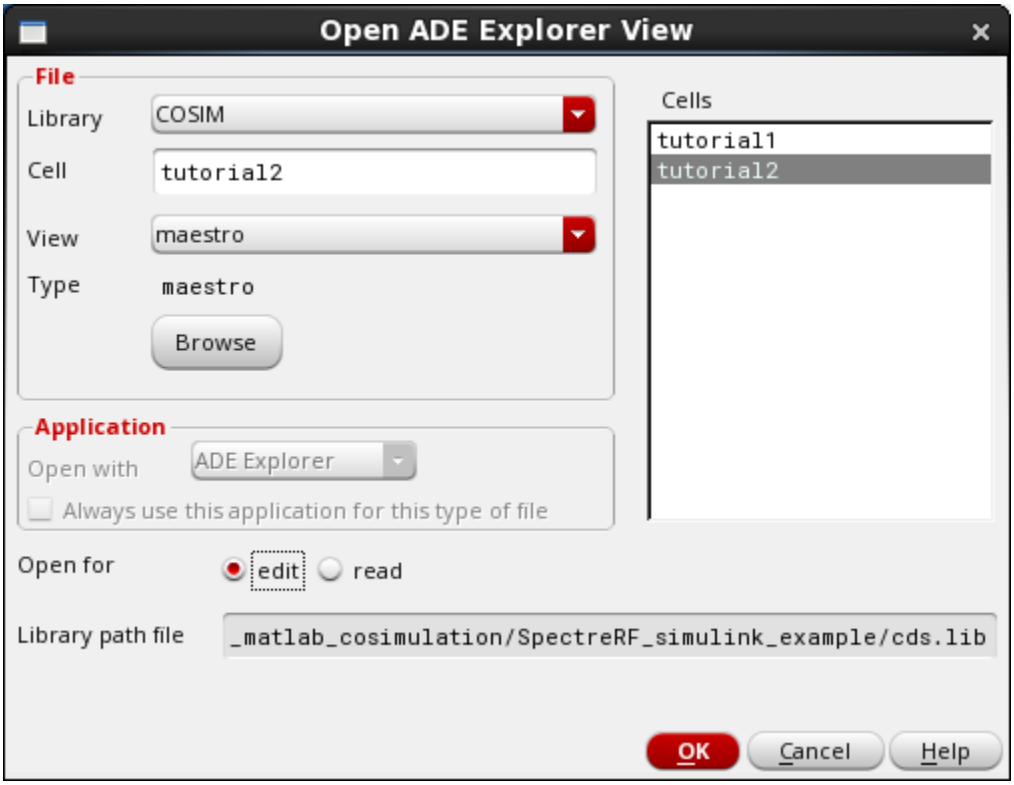
5. In the '`Launch ADE Explorer`' pop-up form as shown in Figure 54, choose '`Open Existing View`' and select '`maestro`' view of '`tutorial2`' Cell from `COSIM` Library, as shown in Figure 55.

Figure 54: Launch ADE Explorer form



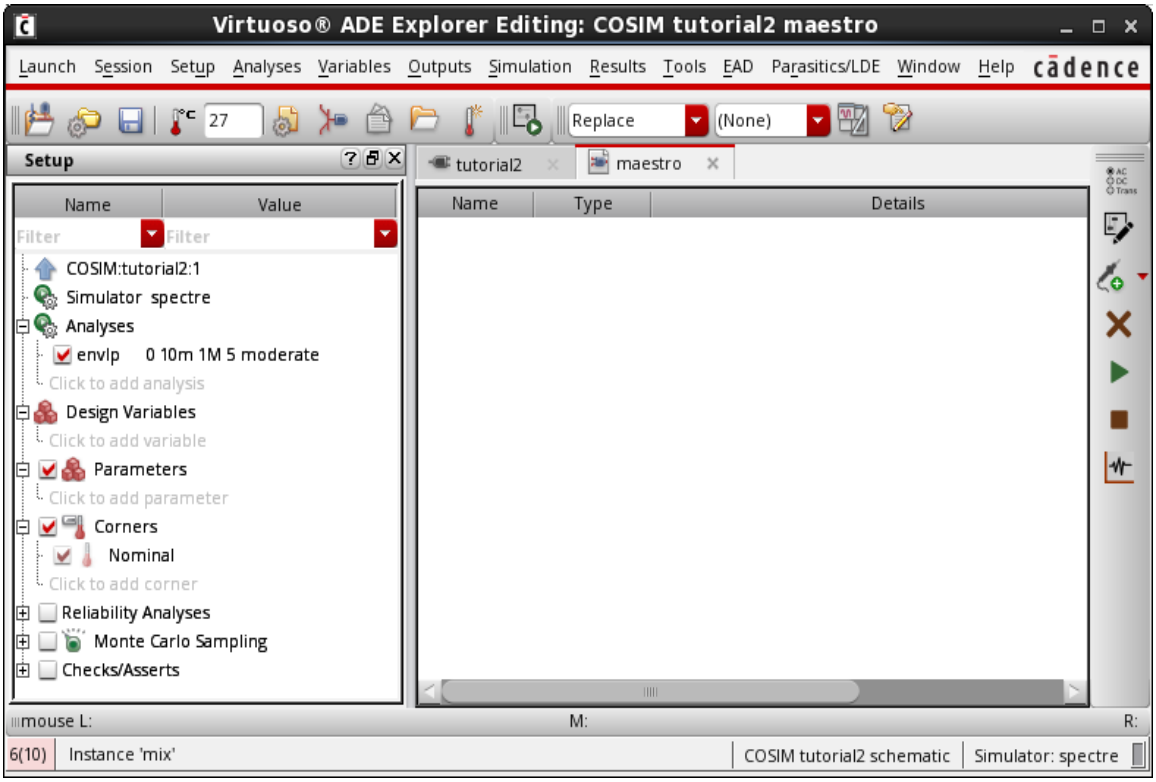
Click '`OK`' on this form.

Figure 55: Selecting 'maestro' view of tutorial2 testbench



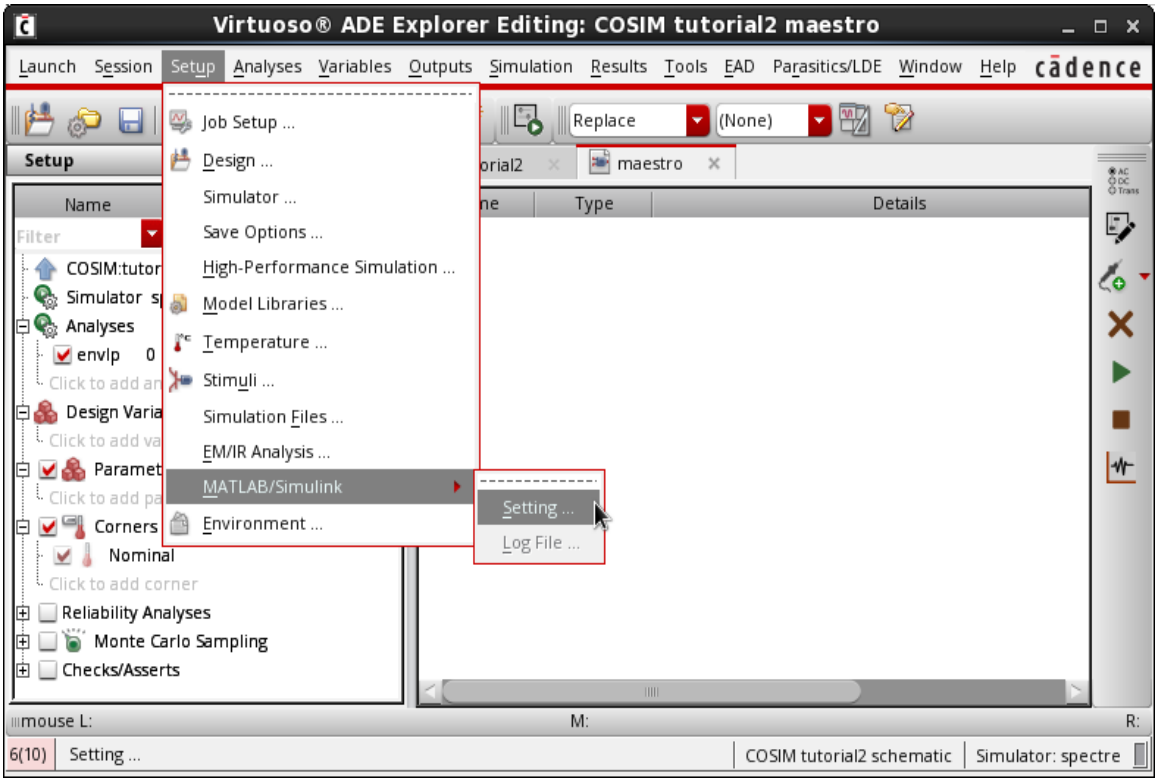
The corresponding `ADE Explorer` window is opened, as shown in Figure 56.

Figure 56: ADE Explorer window showing 'maestro' view of tutorial2 cell



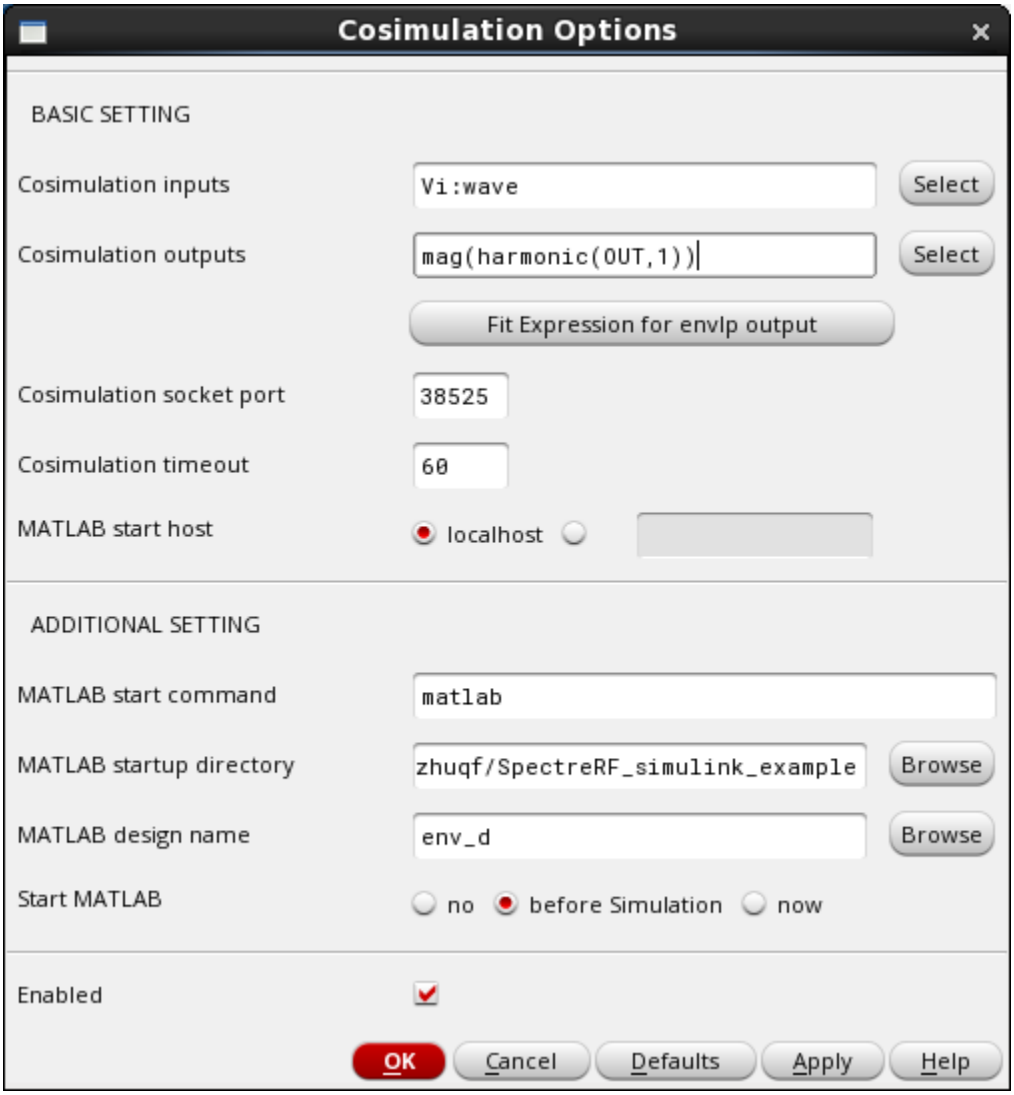
6. From the `ADE Explorer`, navigate to `Setup -> MATLAB/Simulink -> Setting…` option as shown in Figure 57.

Figure 57: Matlab/Simulink Settings from ADE



The '`Cosimulation Options`' form appears as shown in Figure 58.

Figure 58: Cosimulation Options form for tutorial2



Here all the '`Cosimulation Options`' settings are already pre-filled in the form, but the cosimulation option fields can be set as described in the following steps.

a. First, examine the possible values for the '`Start MATLAB`' field present under the '`ADDITIONAL SETTING`' section as shown in Figure 59.

   The behavior of each option is described in the following Table 3.

Figure 59: 'Start MATLAB' options from ADE

Table 3: Options to start MATLAB from ADE window and their details

Value	Behavior
`now`	ADE launches a MATLAB session and opens a MATLAB desktop that is no different than it would be if you launched it manually. When you use the `now` value, you must start the simulation in the Simulink design before starting the SpectreRF simulation. Using the `now` value is a good way to test whether ADE can start MATLAB and open the design successfully.
`before Simulation`	ADE launches an internal MATLAB session, and no MATLAB desktop appears. ADE opens the specified MATLAB design and initiates the MATLAB simulation before Spectre simulation starts. ADE starts MATLAB once but cannot start it again before you close the first session. Cadence suggests using the `now` value as a test before you run with the `before Simulation` value. The `now` and `before Simulation` values share one log file, which can be opened by choosing `Setup -> MATLAB/Simulink -> Log File…` option from the ADE. You can use the log file to monitor the simulation when no MATLAB desktop is visible.
`no`	When this option is selected, the `MATLAB start command`, `MATLAB startup directory`, and `MATLAB design name` fields become deactivated. When you use the `no` value, you must manually start the simulation in the Matlab/Simulink design before starting the SpectreRF simulation.

b. In this `Cosimulation Options` form, under the ‘ADDITIONAL SETTING’ section, for the ‘Start MATLAB’ field, select ‘`now`’.

The MATLAB start command, MATLAB startup directory and MATLAB design name fields become active in the Cosimulation Options form.

c. For the MATLAB start command field, enter matlab.

d. For the MATLAB startup directory, enter the complete *absolute* path to your current Matlab startup directory.

   For example, in this case, it is set to

   /home/…/SpectreRF_simulink_example

   In case, if there is any other path set, ensure to change it properly to point to your current working directory.

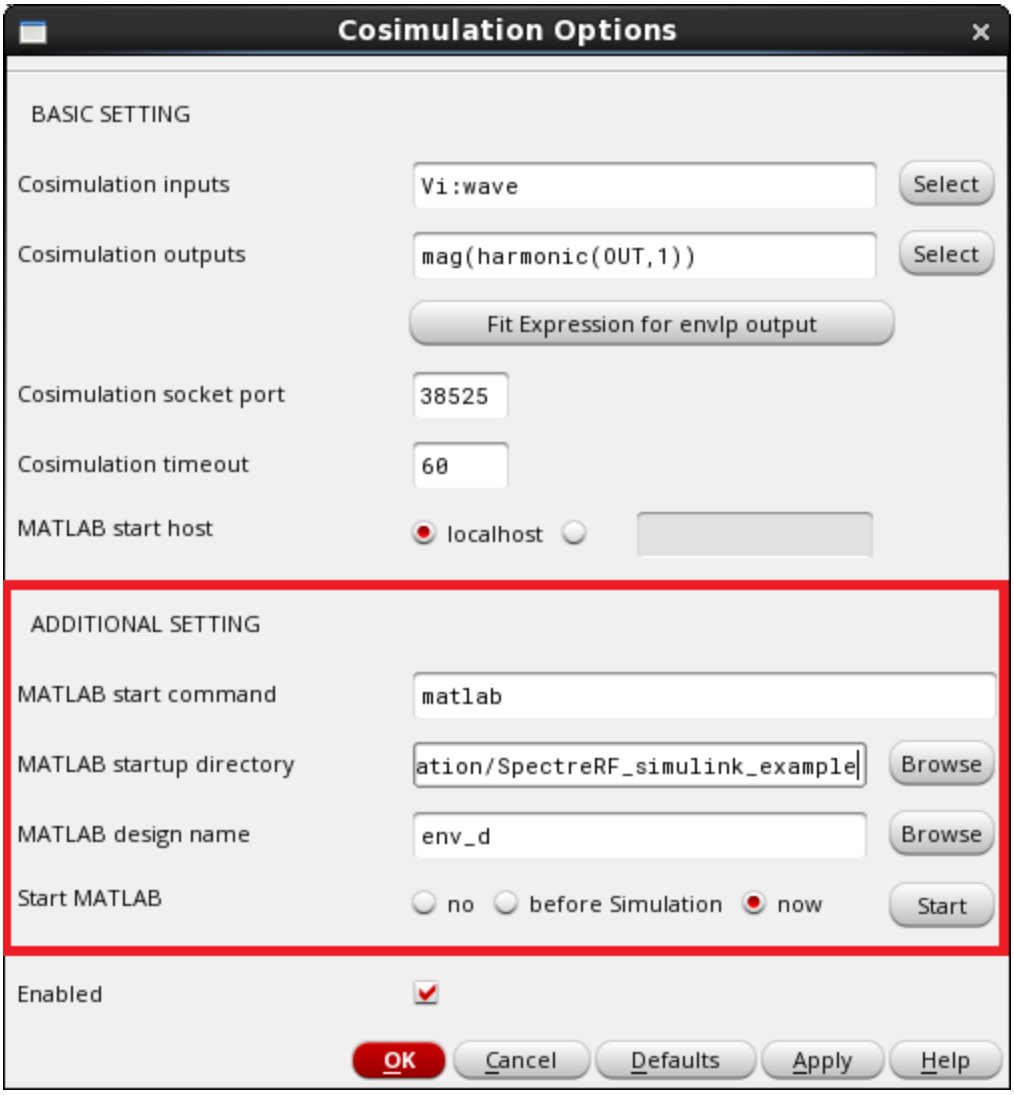
e. Type env_d in the MATLAB design name field.

   where, 'env_d' is your Simulink block present under Tutorial2.

Note: No need to add the extension, *.mdl to the above 'env_d' Matlab design, as the tool will automatically pick the design corresponding to its SpectreMDL file (here, in this case, it is env_d.mdl, present in your 'SpectreRF_simulink_example' directory)

After these steps, the Cosimulation Options form should look as shown in Figure 60.

Figure 60: Cosimulation Options settings for invoking Tutorial 2 in Matlab

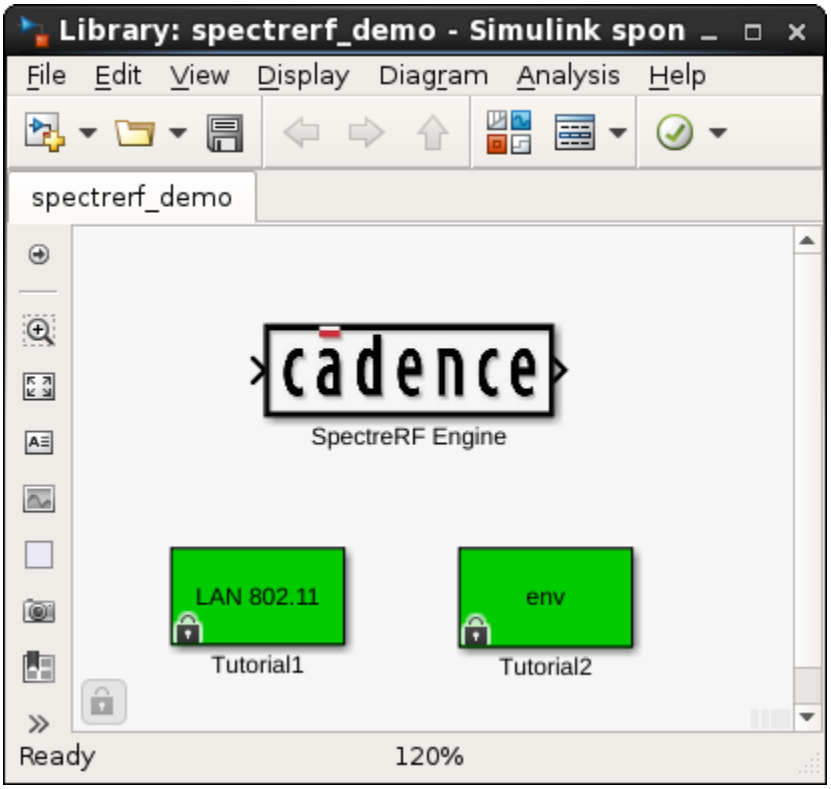


f. In the same `Cosimulation Options` form, with '`now`' option being selected, click the '`Start`' button located beside it for the '`Start MATLAB`' field. You would see the following message in `Virtuoso CIW` session.

```
INFO: executing:"matlab"
```

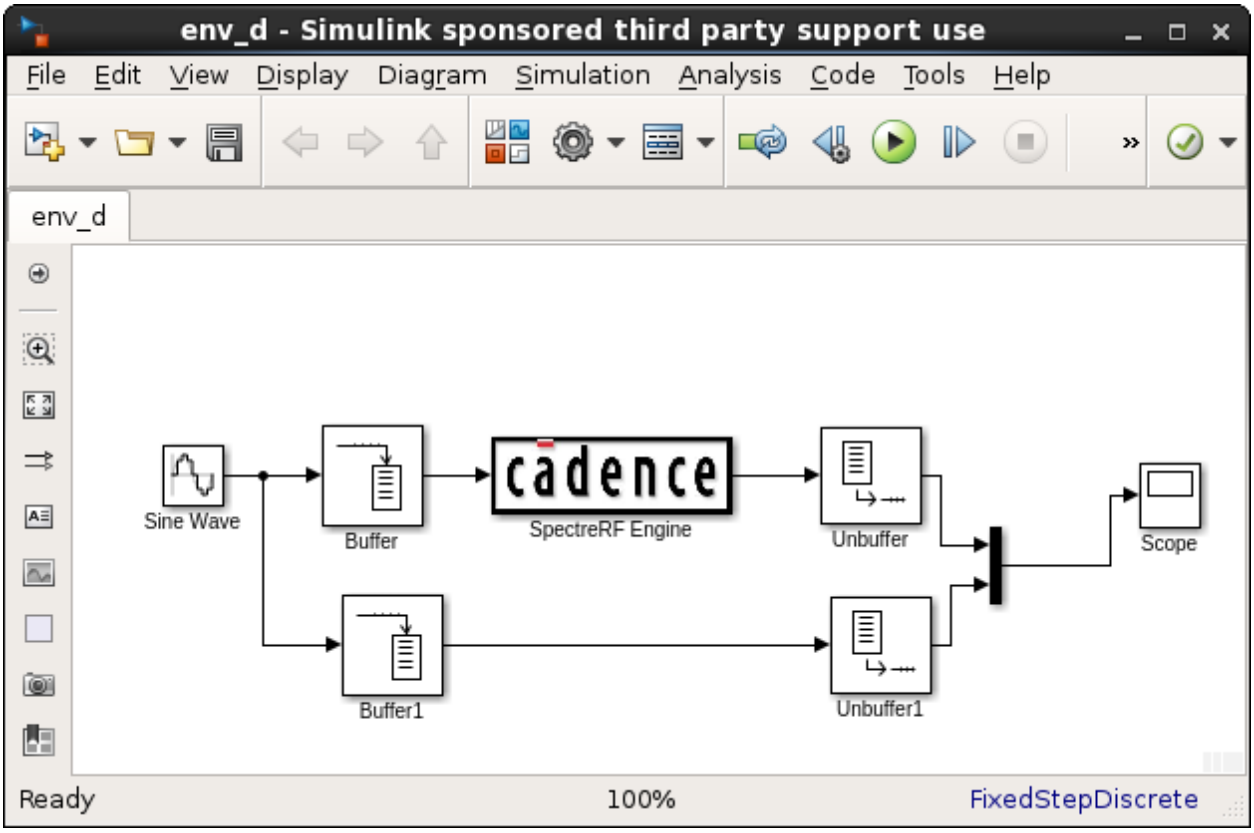
A new `MATLAB` application starts and helps you open the design `env_d`, from `Tutorial2` as shown in Figure 61.

Figure 61: spectrerf_demo Library with Tutorial2 testbench



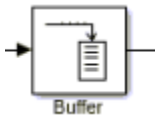
The corresponding '`spectrerf_demo`' Library having `Tutorial2` testbench is also opened in the background as shown in Figure 62.

Figure 62: Tutorial2 'env_d' testbench for cosimulation



7. From the ‘env_d’ testbench opened inside Simulink, double-click on ‘Buffer’ component (Figure 63).

Figure 63: 'Buffer' component of 'env_d' testbench

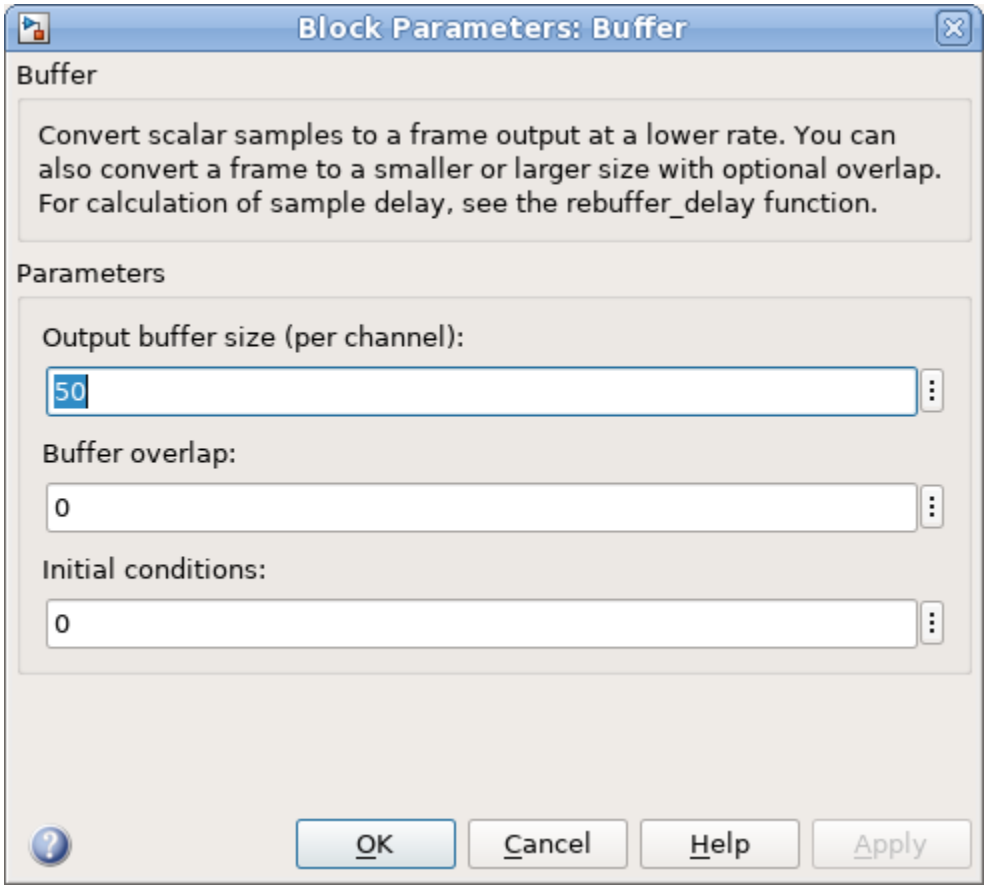


The corresponding ‘Block Parameters: Buffer’ form opens.

a. Set the ‘Output buffer size (per channel)’ field to 50 and set other fields as shown in Figure 64.

   The same ‘`Buffer`’ component property settings apply for the ‘`Buffer1`’ component.

Figure 64:Setting properties of 'Buffer" component



b. Click OK when done.

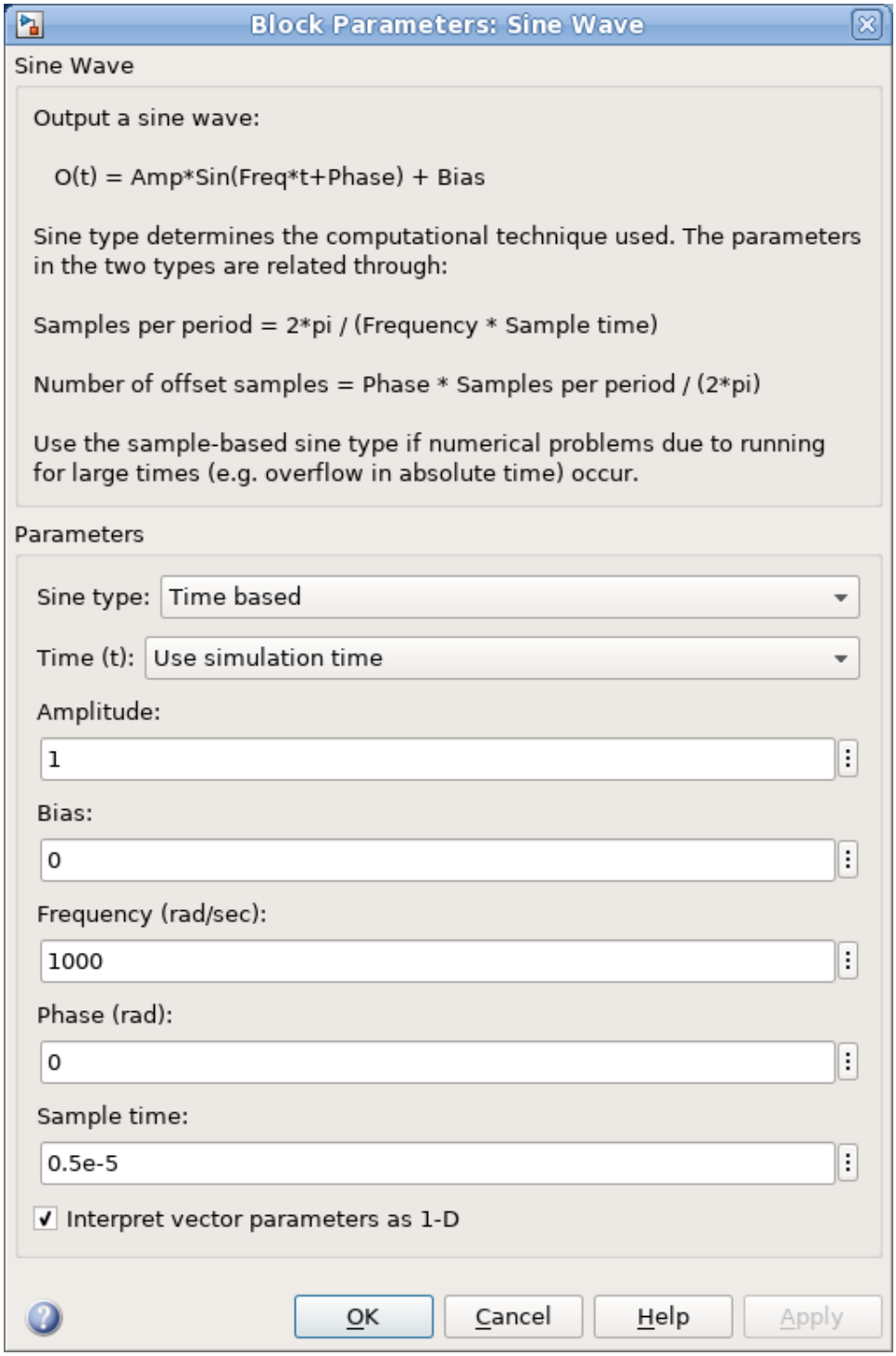
8. Double-click on the ‘Sine Wave’ component in the `env_d` testbench.

   a. Set the ‘Frequency (rad/sec)’ field is to ‘1000’.

   b. Set the ‘Sample time’ field to ‘0.5e-5’.

   c. Set rest of the form as shown in Figure 65.

Figure 65: Setting 'Sine Wave' component properties

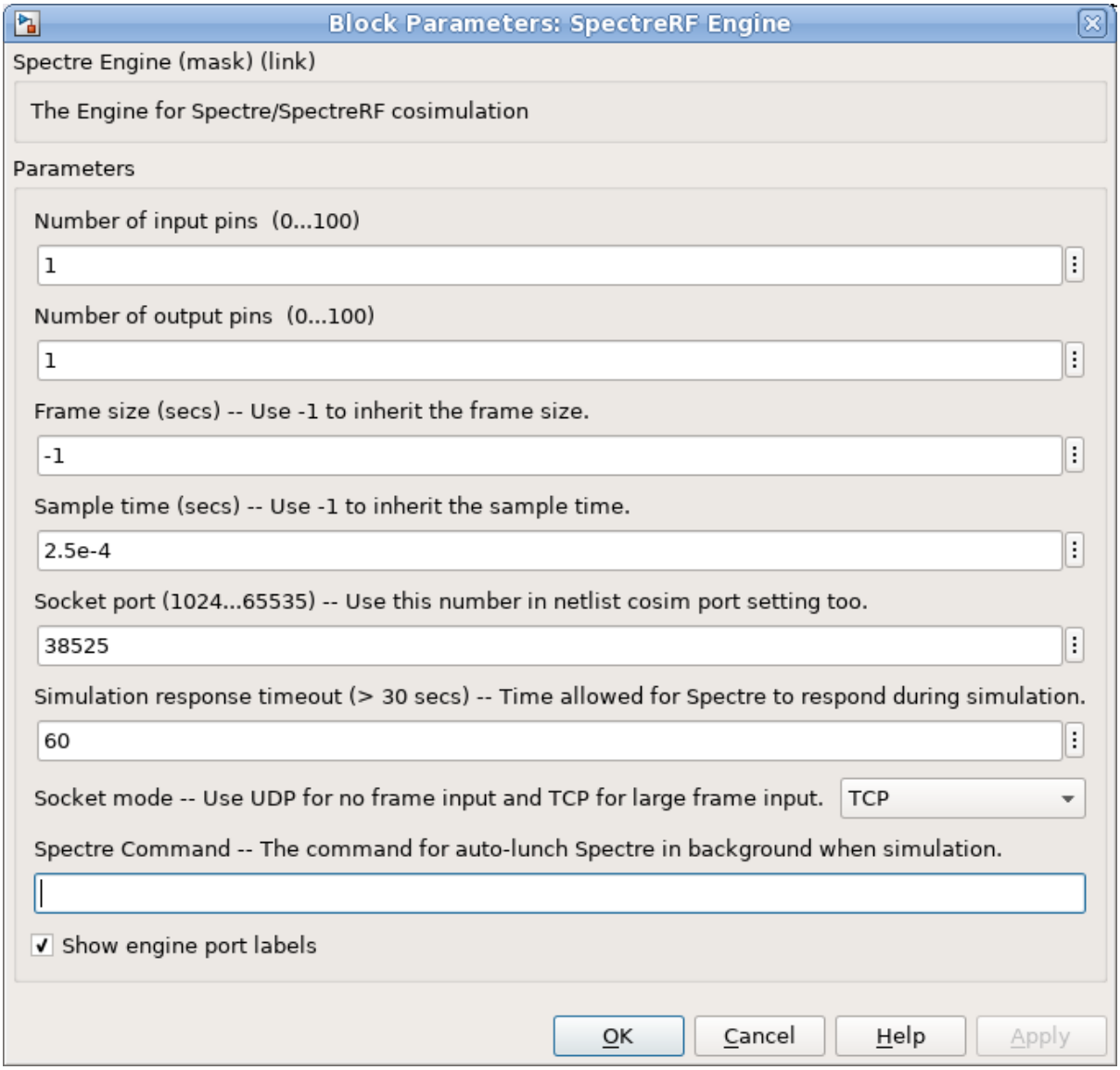


d. Click OK when done.

9. Double-click the `SpectreRF Engine` block (shown in Figure 66) and set the parameters in it as shown in following Figure 67.

Figure 66: 'SpectreRF Engine' component

Figure 67: Block Parameters setup of 'SpectreRF Engine' component in Tutorial 2



10. In this 'Block Parameters: SpectreRF Engine' form,

    a. Set the 'Number of input pins (0…100)' field to '1'

    b. Set the 'Number of output pins (0…100)' field to '1'.

    c. Let the 'Frame size (secs)' field be set to default '-1'.

    d. Set the 'Sample time (secs)' field to '2.5e-4', which is the sample time of [Sine Wave(=0.5e-5) * 50 * Frame size].

    e. The Sample time can also be set to '-1', if you are uncertain about the appropriate time to use.

    f. Set the 'Socket port (1024…65535)' field to '38525', i.e. to the value you set in the Virtuoso ADE maestro 'Cosimulation Options' form (Setup -> MATLAB/Simulink -> Setting…) from the above steps.

    g. Leave the 'Simulation response timeout (>30 secs)' field set to '60' seconds and ensure that the 'Socket mode' to use as 'TCP'.

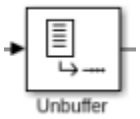
    h. Also, make sure to leave the 'Spectre Command' field empty. The use of this field will be explained in later sections.

    i. You might want to enable the 'Show engine port labels' checkbox to display the number of input / output ports on the 'SpectreRF Engine' component.

    j. Click 'OK' when done.

11. Now, double-click on the 'Unbuffer' component (shown in Figure 68) from 'env_d' testbench.

Figure 68: 'Unbuffer' component of 'env_d' testbench



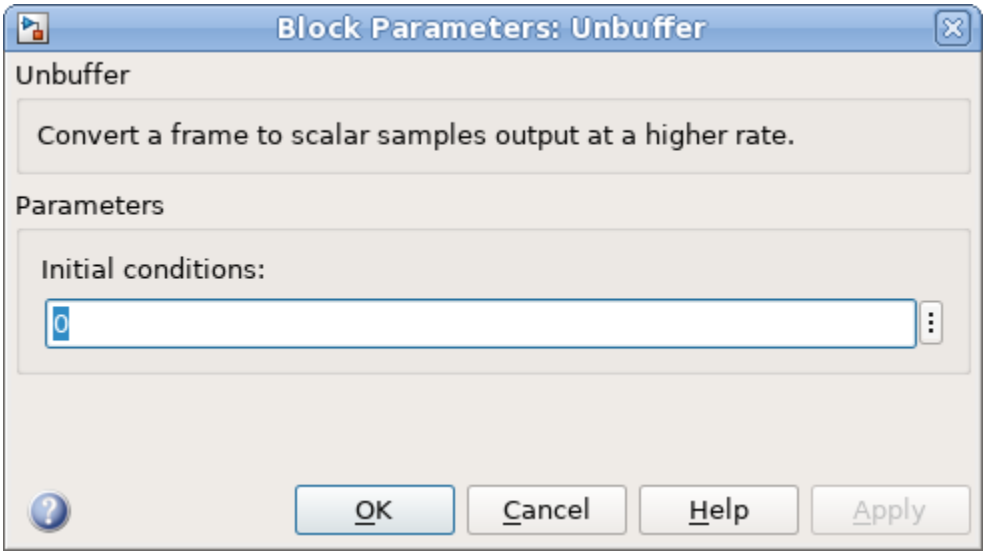
The corresponding 'Block Parameters: Unbuffer' form opens.

12.

a. Ensure that the '`Initial conditions:`' field is set to '`0`' as shown in Figure 69.

   The same '`Unbuffer`' component property settings apply for the '`Unbuffer1`' component as well.

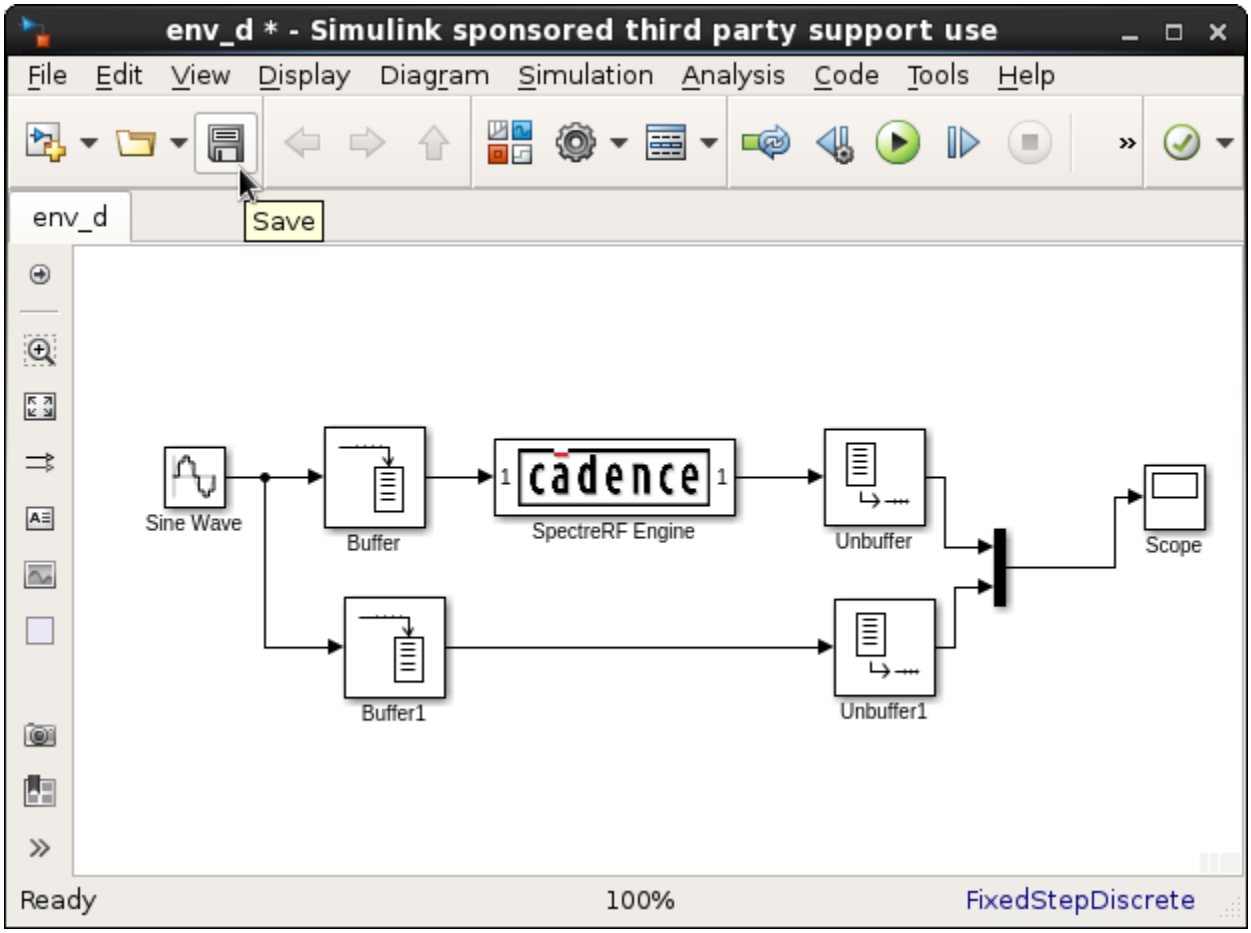
Figure 69: Setting properties of 'Unbuffer" component



b. Click '`OK`' when done.

13. Choose `File -> Save` from the '`env_d`' Simulink testbench, or click on the '`Save`' icon as shown in Figure 70.

Figure 70: Save the 'env_d' testbench

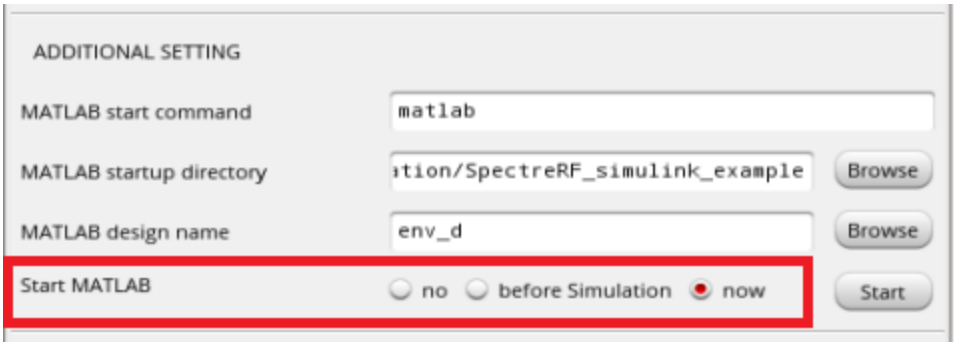


14. Now, switch to the `Virtuoso ADE Explorer` window and open the '`Cosimulation Options`' form from the '`maestro`' tab by navigating to `Setup -> MATLAB/Simulink -> Setting…` option.

15. In the '`Cosimulation Options`' form, ensure that the '`Start MATLAB`' option is set to '`now`' under the '`ADDITIONAL SETTING`' section as shown in Figure 71.

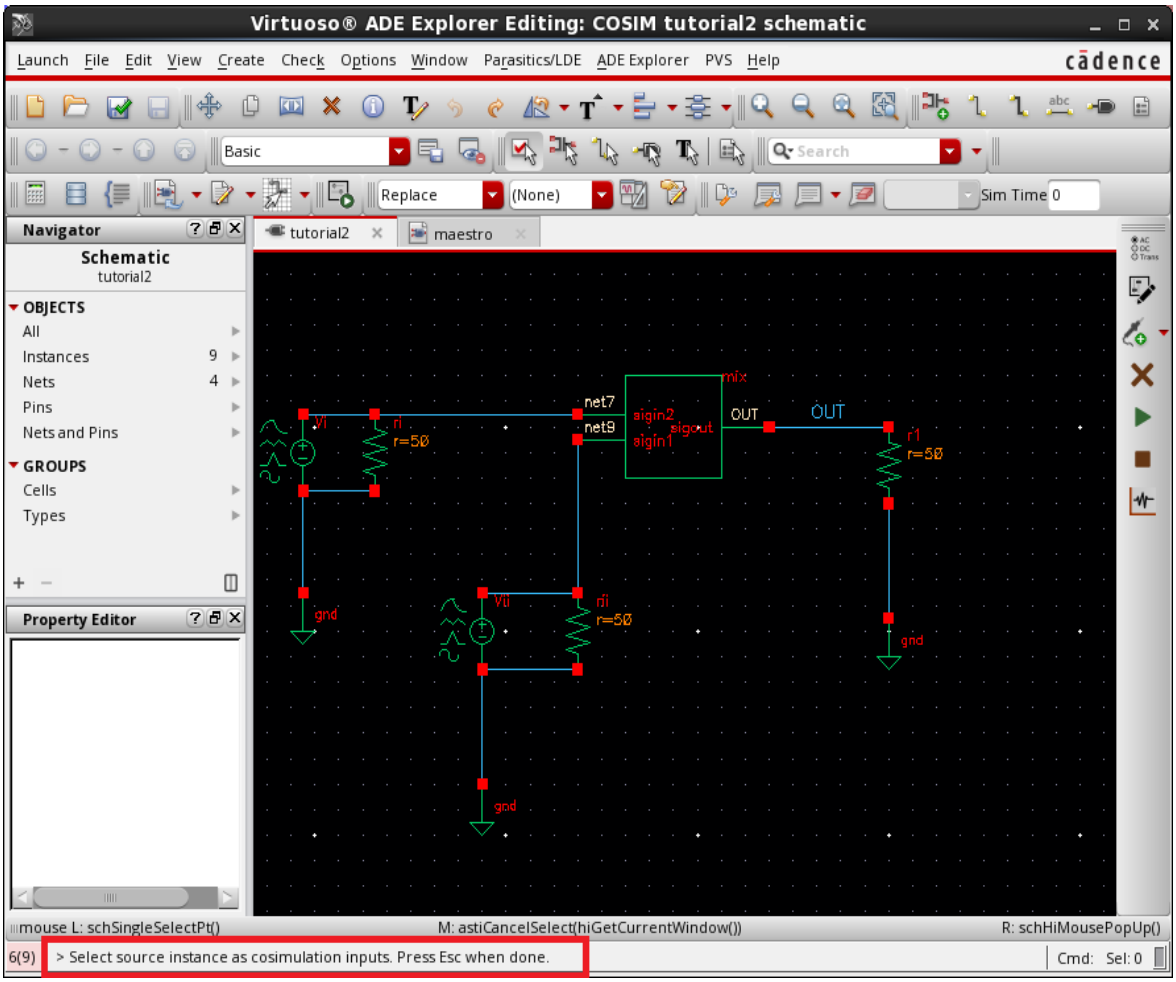
    Note: You can also set the '`Start MATLAB`' option to '`before Simulation`' here. The '`before Simulation`' mode launches `Matlab` automatically from the `Virtuoso ADE Explorer` window, brings up the `Simulink` testbench and then runs the cosimulation.

Figure 71: 'Start MATLAB' option is set to 'now'



16. In the same '`Cosimulation Options`' form, click the '`Select`' button besides `Cosimulation inputs` field. The corresponding `tutorial2` '`schematic`' view will be automatically opened as shown in Figure 72.

Figure 72: Selecting cosimulation inputs from tutorial2 schematic



In the schematic window shown in Figure 73 you will see the following information at the bottom of schematic window.

```
Select source instance as cosimulation inputs. Press Esc
when done.
```

17. On the schematic, select only the 'vsource' component, Vi, and then press the *Esc* key to finish.

    The corresponding 'Cosimulation inputs' field will be filled with an entry as shown.

    ```
    Vi:wave
    ```

18. From the Cosimulation Options form, now click the 'Select' button located beside 'Cosimulation outputs' field. The corresponding tutorial2 'schematic' view will be automatically opened, where you will see the following information at the bottom of schematic window.

    ```
    Select Net/Terminal as cosimulation outputs. Press Esc when
    done.
    ```

19. On the schematic window, select the net OUT, and then press the *Esc* key to finish.

    The corresponding 'Cosimulation outputs' field will be filled with an entry as shown.

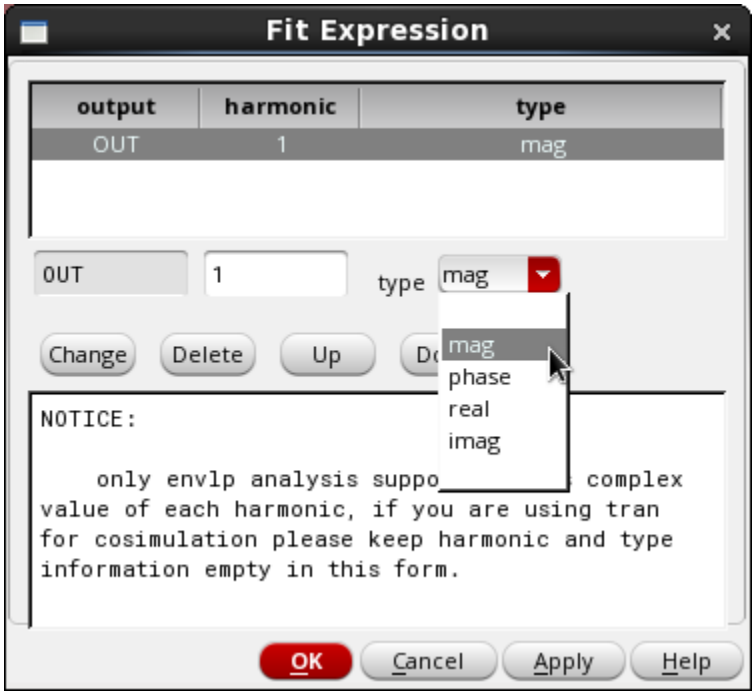
    ```
    OUT
    ```

20. Click the 'Fit Expression for envlp output' button available as shown in Figure 73. The Fit Expression form appears.

Figure 73: 'Fit Expression for envlp output' button

21. In the 'Fit Expression' form, select **output** OUT, change the **harmonic** to '1', and change the **type** to mag from the drop-down list, as shown in Figure 74.

Figure 74: 'Fit Expression' form for 'envlp' analysis output



22. Click the 

'button, and then click '`OK`', to close the `Fit Expression` form.

23. The corresponding '`Cosimulation outputs`' field is now changed to the expression

    ```
    mag(harmonic(OUT,1))
    ```

24. Ensure that the value of '`Cosimulation socket port`' field is the same as the port value of the `SpectreRF Engine` block defined in the above steps of this section.

    In this case, it is set to '`38525`'.

25. In the '`Cosimulation timeout`' field, the value entered is the same as the '`Simulation response timeout (>30 secs)`' field of `SpectreRF Engine` component defined in the above steps of this section.

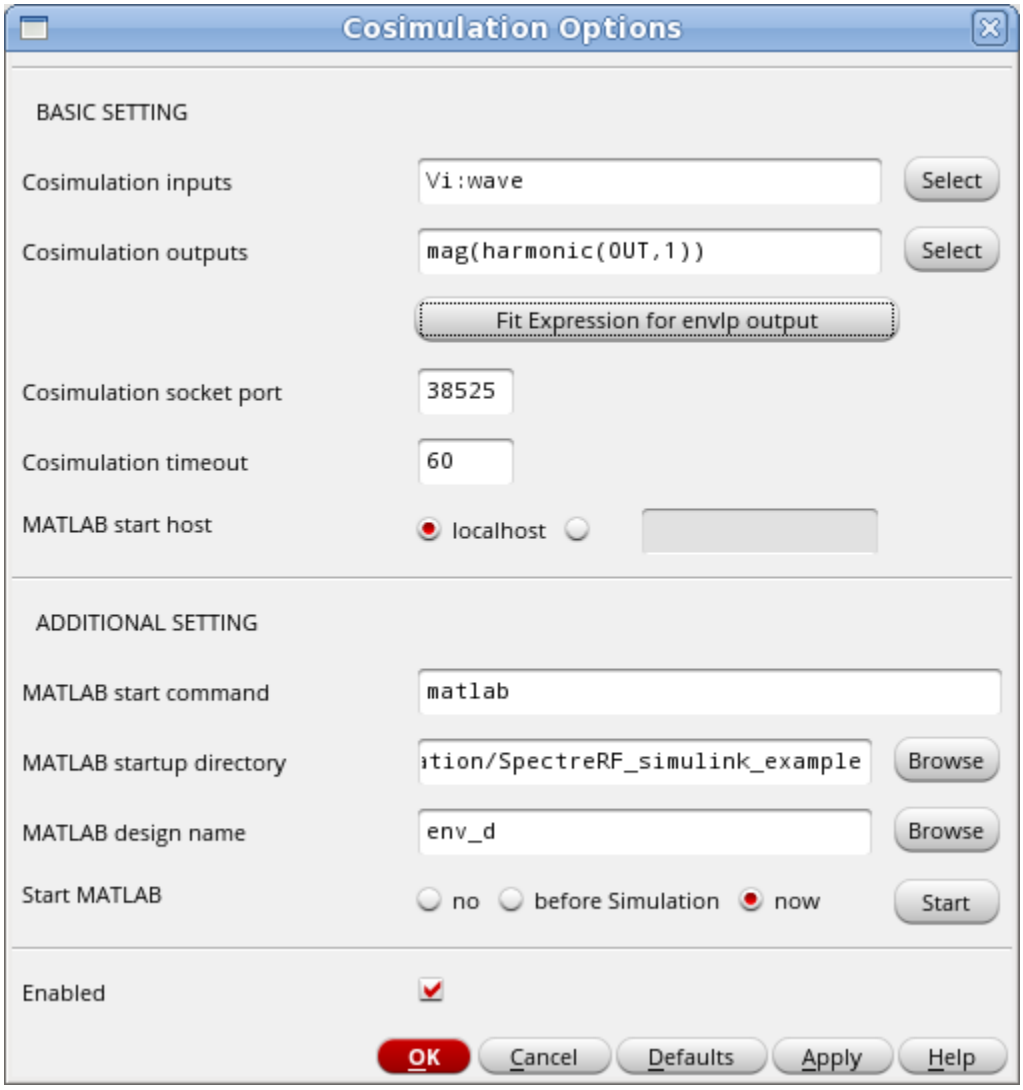
    Hence, it is set to '`60`' seconds.

    Keep the '`MATLAB start host`' option set to default '`localhost`'.

26. At the bottom of this `Cosimulation Options` form, turn on '`Enabled`' checkbox.

    The `Cosimulation Options` form after all the settings done from Step 15 to Step 26 should look as shown in Figure 75.

Figure 75: Cosimulation Options settings for tutorial2 testbench

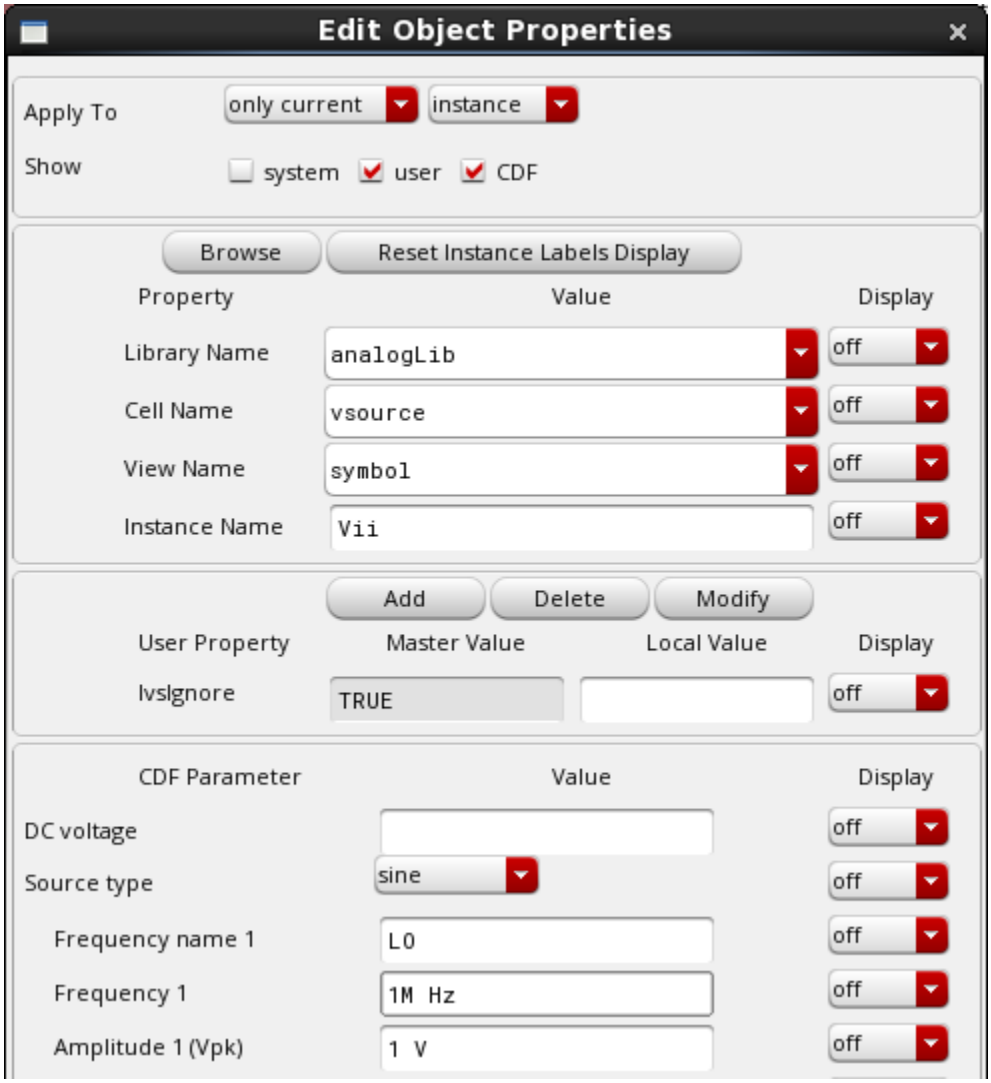


27. Click '`OK`' to apply the changes.

28. Switch to the '`schematic`' view of '`tutorial2`' and set the frequency of the '`vsource`' component '`Vii`' to '`1M`' as shown in Figure 76.

29. Click '`OK`' when done.

Figure 76: Setting 'Vii' source component properties



30. Perform '`Check and Save`'  on the `tutorial2` '`schematic`' testbench.

31. Switch back to the '`maestro`' tab, from the `tutorial2` '`schematic`' tab.

32. From the `ADE Explorer window`, navigate to `Analyses -> Choose…` option. Use the '`Choosing Analyses - ADE Explorer`' form to add an `envlp` analysis.

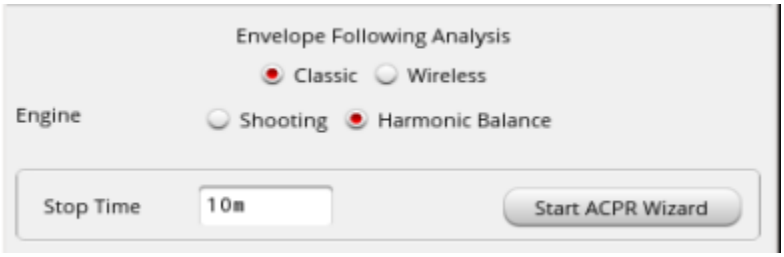
33. Select '`envlp`' analysis option from the `Choosing Analyses` form.

    a. Under the '`Envelope Following Analysis`' section, set it to '`Classic`' mode

b. Set the 'Engine' as 'Harmonic Balance'.

c. Make the 'Stop Time' field set to 10m as shown in Figure 77.

Figure 77: Classic Envelope analysis with Harmonic Balance (HB) engine



d. With the 'Tones' mode set to 'Frequencies', select the 'Number of Tones' as '1'.

e. To match the fast frequency of '1MHz' set for 'vsource' component 'Vii' in the schematic, set the Fundamental Frequency to '1M' under the 'Number of Tones' section for Tone1.

f. Set the Number of Harmonics to '5'. This value should be larger than '1', because the 1st Harmonic is used in the cosimulation.

g. Keep the 'Oversample Factor' set to '1', and 'Freqdivide Ratio' to '1' by default.

h. Make the 'Time Step Control' field to 'adaptive'.

i. For the accuracy, keep the default 'moderate' settings under the 'Accuracy Defaults (errpreset)' option.

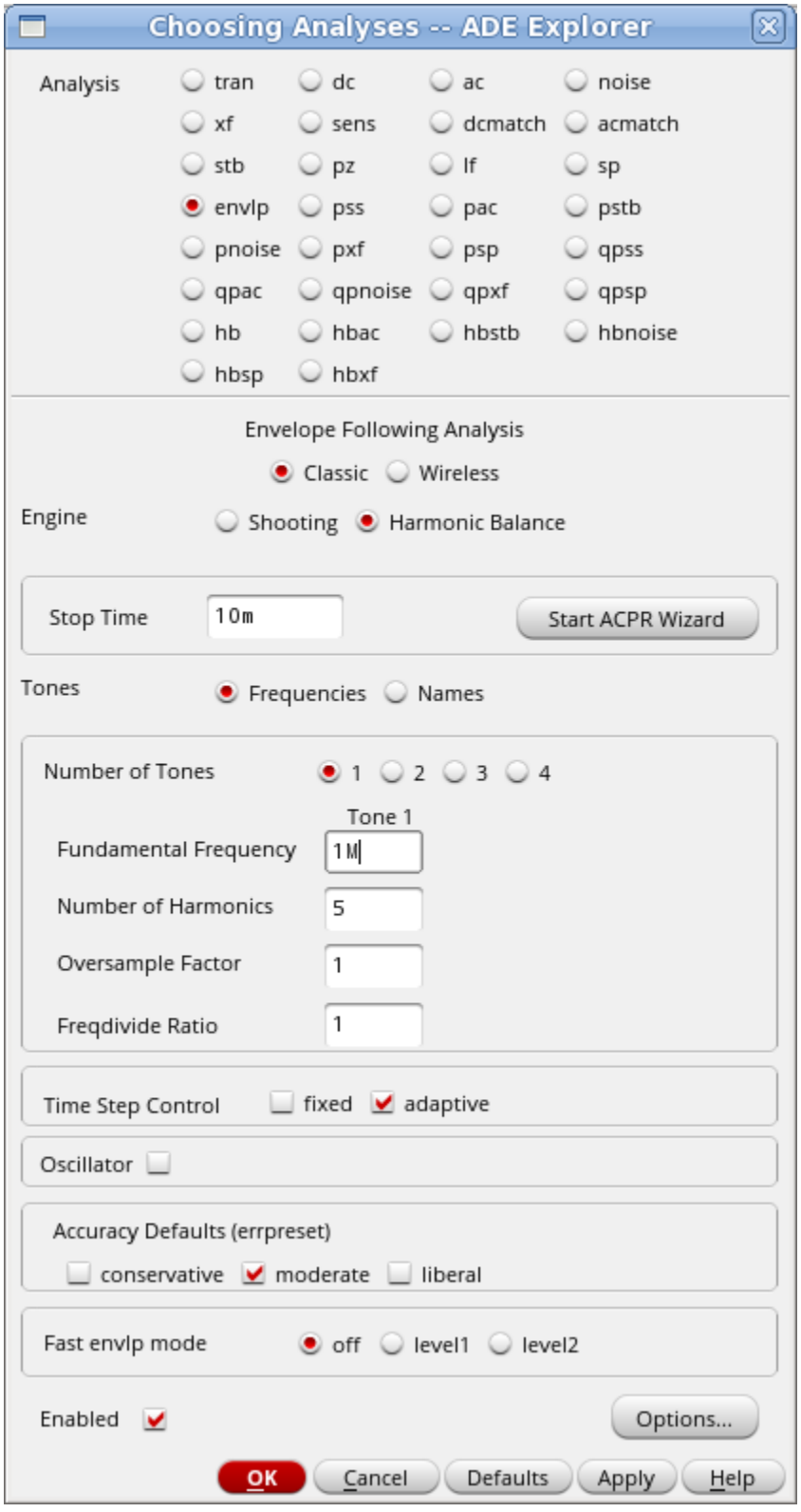
j. Here, we are setting the 'off' option as the 'Fast envlp mode'.

k. Click the 'Enabled' checkbox at the bottom of the 'envlp' analysis.

After performing the above steps, the Choosing Analyses – ADE Explorer form for 'envlp' analysis will look as shown in Figure 78.

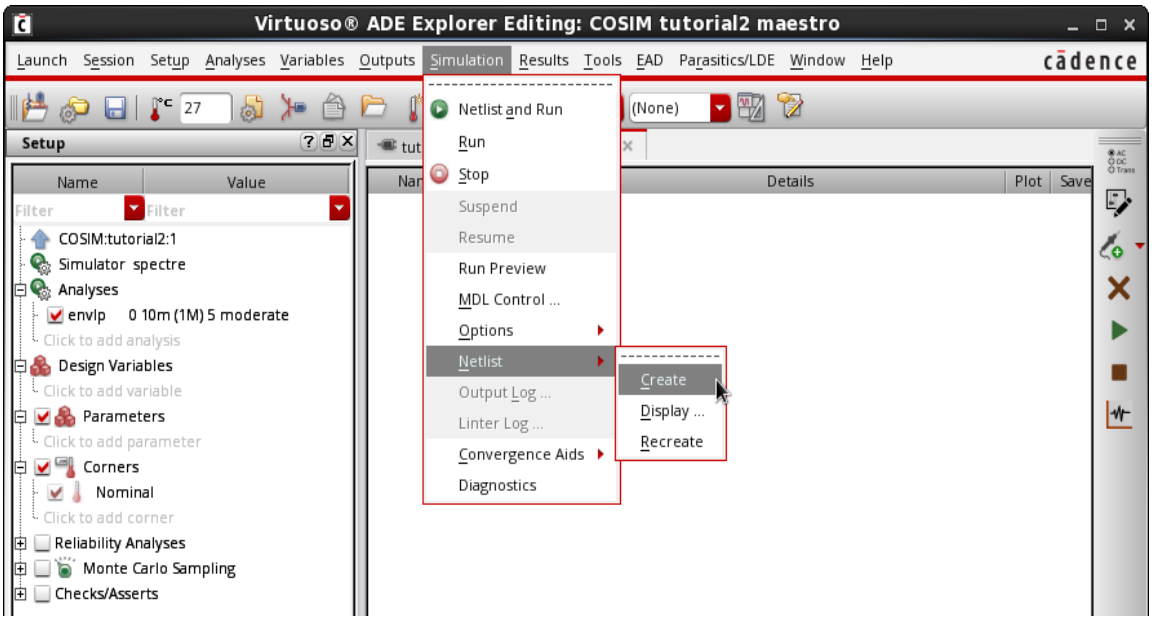
l. Click 'OK' on this 'envlp' analysis form to apply the changes.

Figure 78: Setup for the Envelope (envlp) analysis



34. In order to generate the netlist, from the `ADE Explorer` (`maestro`) view, navigate to `Simulation -> Netlist -> Create` as shown in Figure 79.

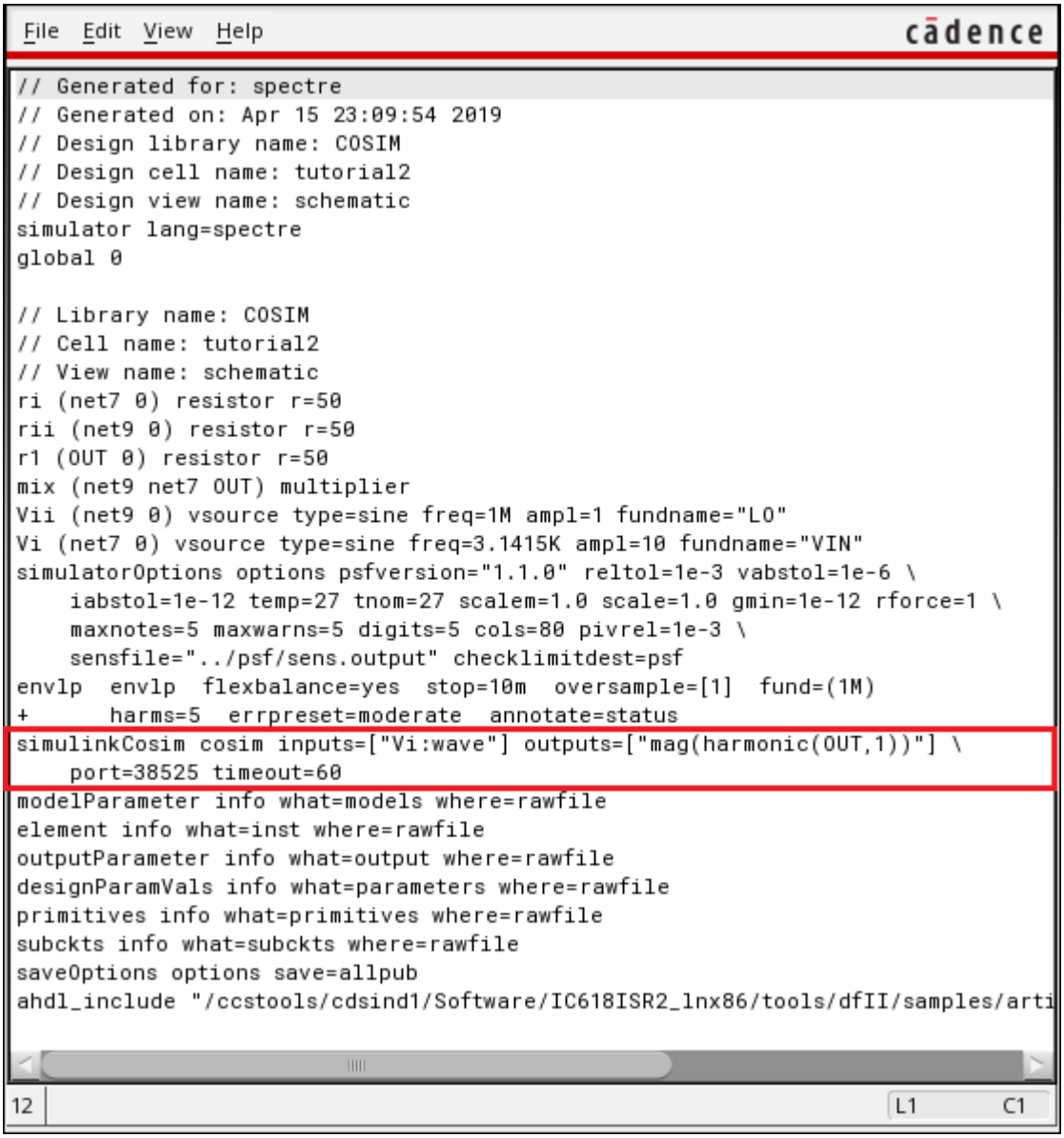
Figure 79: Netlist generation from ADE after setting 'envlp' analysis



35. Display the generated netlist and review it. It would look as shown in Figure 80.

    Make sure the `simulinkCosim cosim` statement to be used for cosimulation appears in it.

Figure 80: 'evnlp' analysis netlist with cosimulation statement

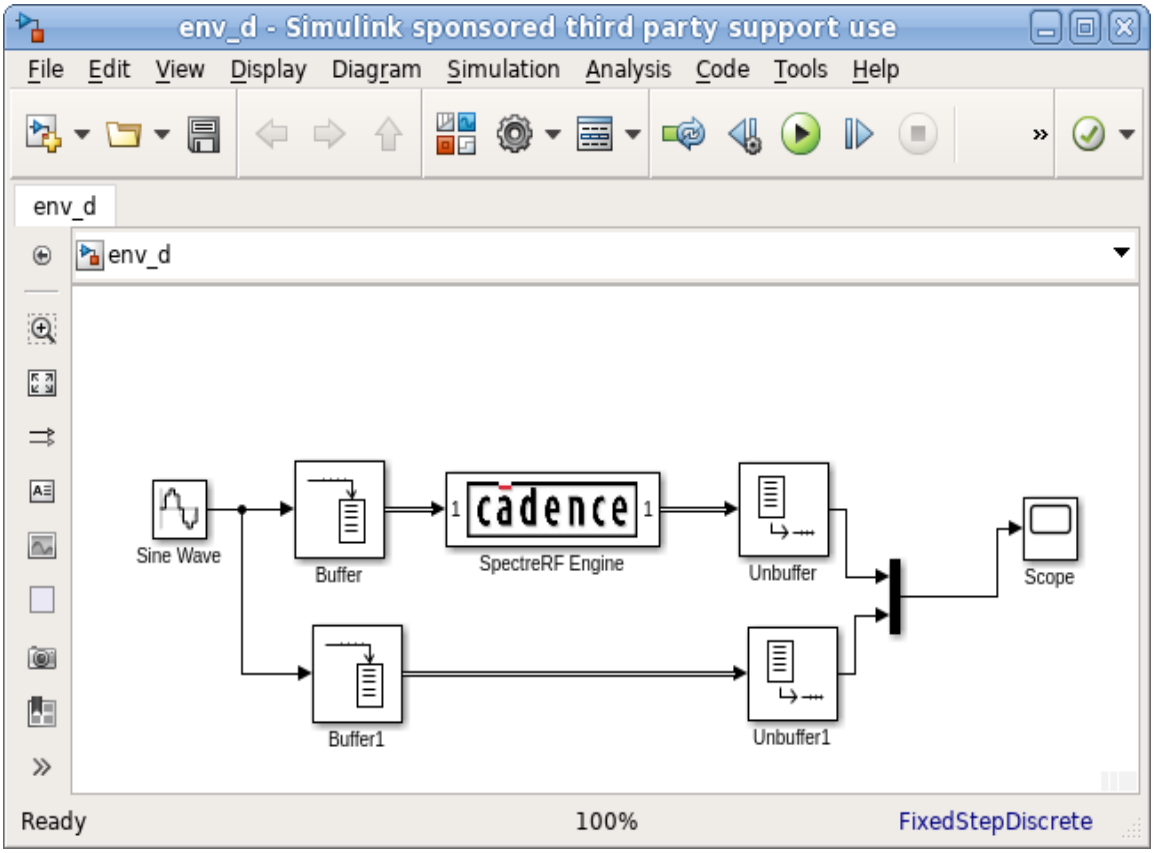


This step completes this section of '`Setting up the cosimulation from ADE`'.

Running the cosimulation from ADE

36. Since we had set the '`Start MATLAB`' option to '`now`' under the '`Cosimulation Options`' form in ADE from the above section, '`Setting up the cosimulation from ADE`', ensure that the `Matlab` '`env_d`' `Tutorial2` Simulink testbench is already opened in the background as shown in Figure 81.

Figure 81: 'env_d' Simulink testbench for cosimulation



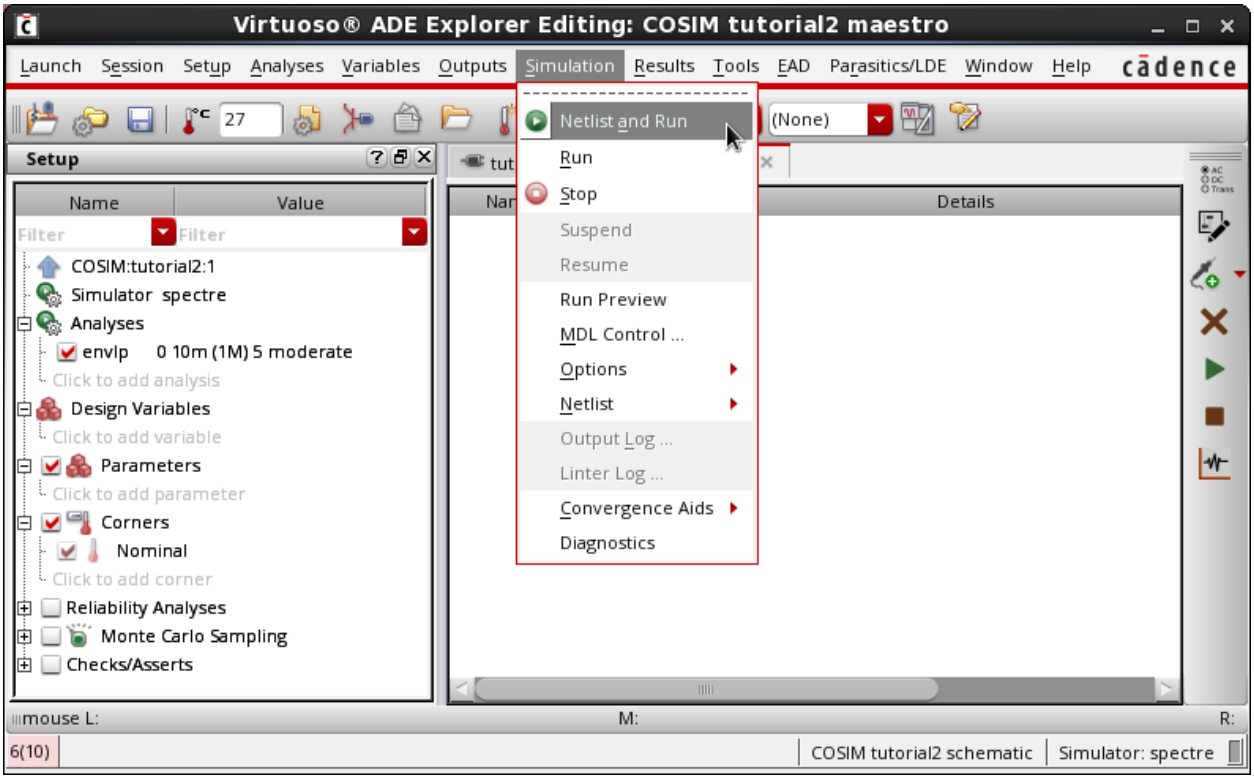
Note: When you use the 'now' `Start MATLAB` option, you must start the simulation in the `Matlab/Simulink` design before starting the `SpectreRF` simulation from `Virtuoso ADE`.

37. To start the `Matlab/Simulink` simulation of `Tutorial2` '`env_d`' testbench, hit the green Run button (or) click `Simulation -> Run` from the Matlab/Simulink window.

    Immediately after pressing the green Run button from `Simulink` testbench, move to `Virtuoso ADE Explorer` window with '`maestro`' view and execute the next step (Step 38).

38. From the `ADE Explorer` '`maestro`' view, hit the green '`Run Simulation`' button from the right edge of `ADE Explorer`, or navigate to `Simulation -> Netlist and Run` option as shown in Figure 82.

Figure 82: Netlist and Run the cosimulation from Virtuoso ADE



The cosimulation automatically runs the MATLAB -> '`env_d`' testbench of `Tutorial2` in the background.

Note: From the earlier section on '`Setting up the cosimulation from ADE`' if you would have set the '`Start MATLAB`' field to '`before Simulation`' option under the '`Cosimulation Options`' form, following message would appear in the `Virtuoso CIW`.

```
Virtuoso® ADE : Opening MATLAB® (this may take a while -
please be patient)...

INFO: executing:"matlab"
```

With '`before Simulation`' mode, `Virtuoso ADE` starts an internal MATLAB session and no MATLAB desktop appears. `ADE` controls the `MATLAB` simulation automatically, by first opening the Simulink design testbench and then triggering to simulate it.

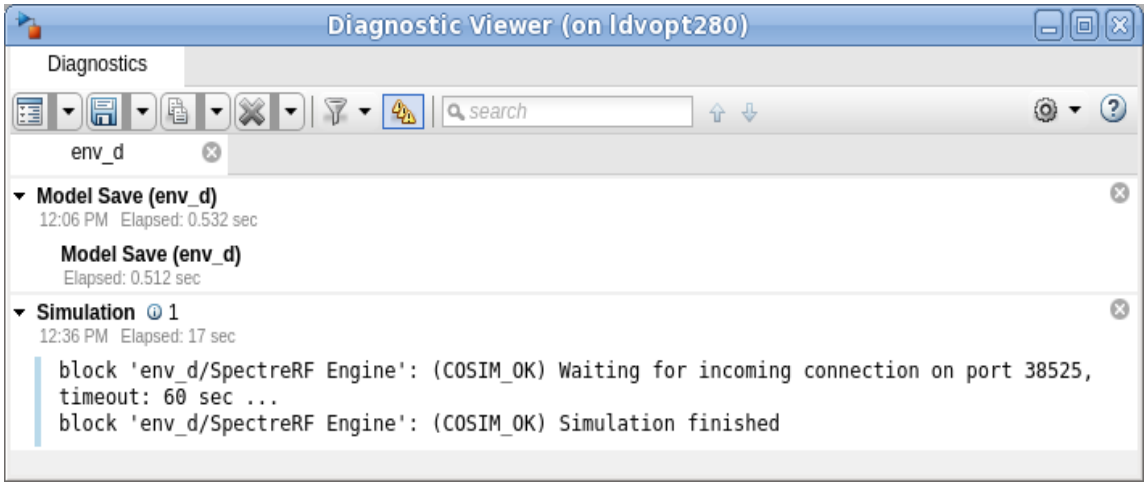
`Virtuoso Spectre` resumes running when MATLAB finishes initialization.

39. When the cosimulation ends, the MATLAB desktop issues the following message:

```
block 'env_d/SpectreRF Engine': (COSIM_OK) Simulation
finished
```

The above message can be seen by navigating from the `Matlab/Simulink` 'env_d' testbench window to `View -> Diagnostic Viewer` option as shown in Figure 83.

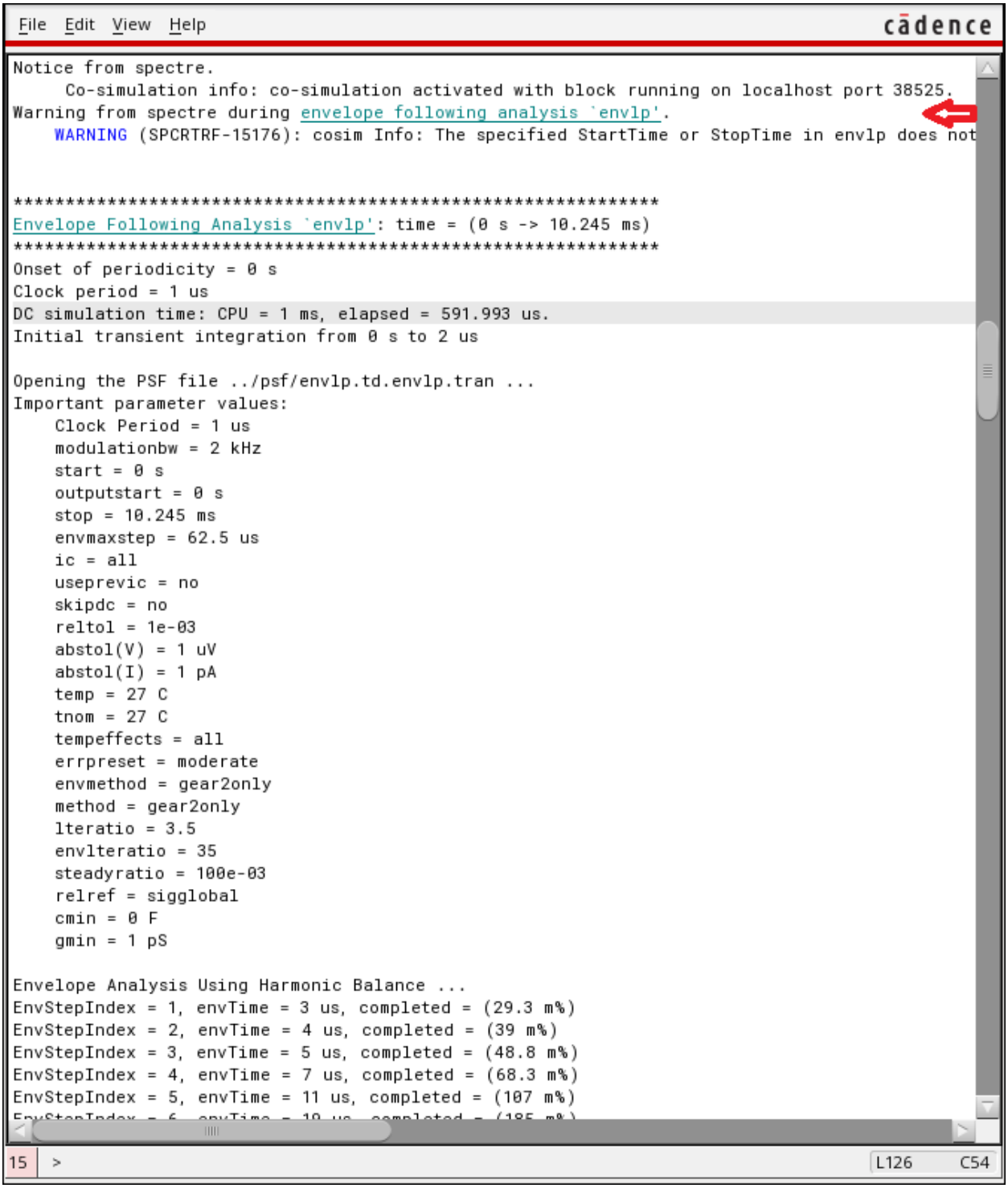
Figure 83: Simulink Diagnostic Viewer window after finishing cosimulation



40. After running the cosimulation from `Virtuoso ADE`, you would see the following message (as shown in Figure 84) inside the generated spectre output log:

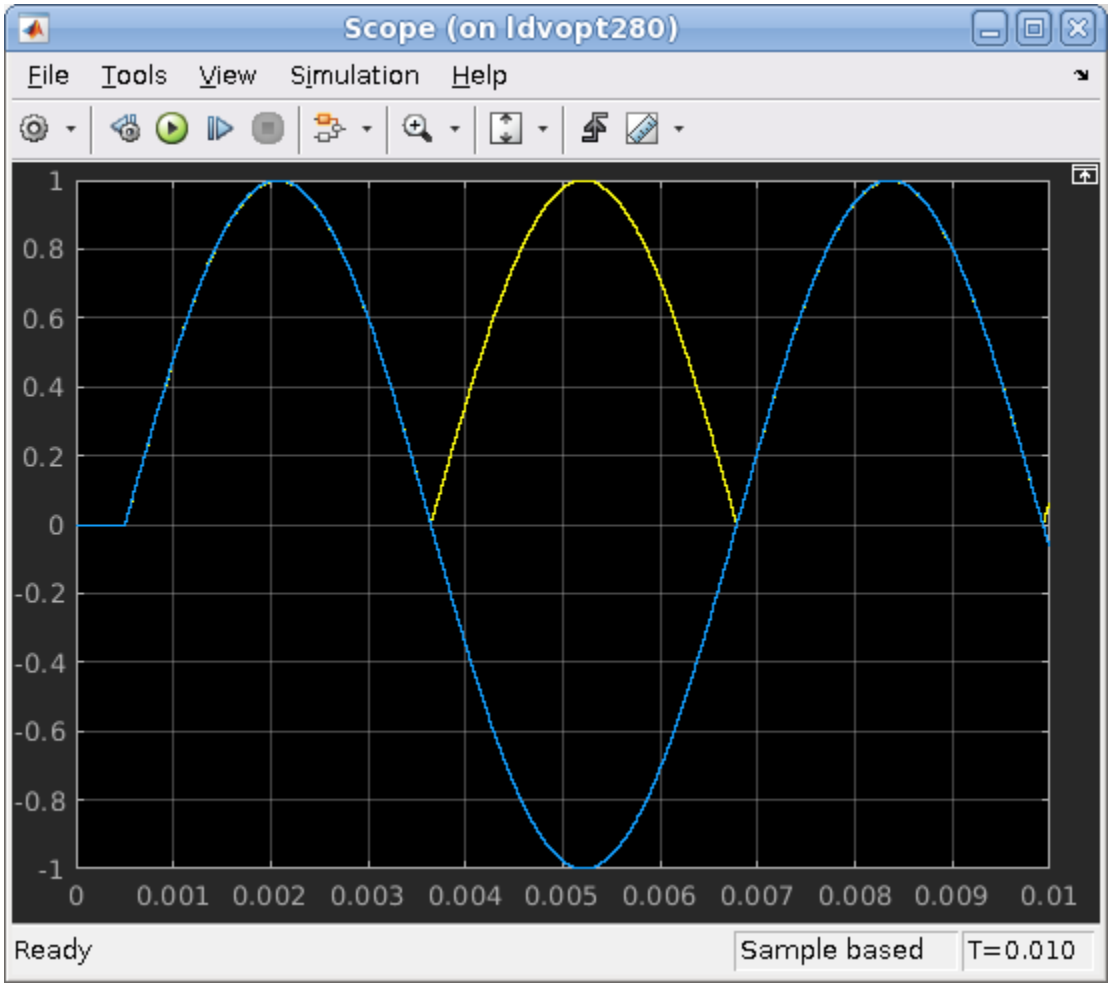
```
Co-simulation info: co-simulation activated with block
running on localhost port 38525.
```

Figure 84: Generated spectre.out after cosimulation of 'envlp' analysis



41. Now, switch to the `Matlab/Simulink` environment of '`env_d`' testbench and double-click on the '`Scope`' block to get the resulting plot as shown in Figure 85.

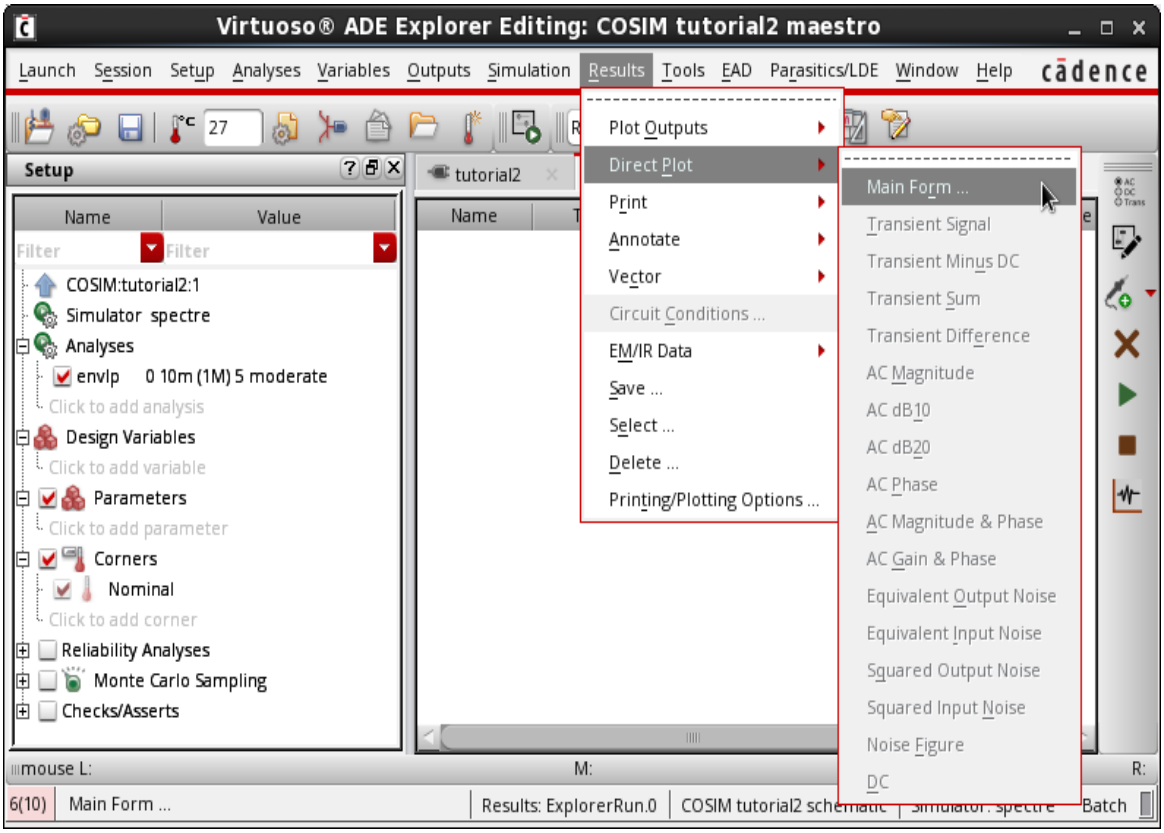
Figure 85: Result of envelope analysis for 'OUT' signal from 1st Harmonic



The "**blue**" curve shown in above Figure 85 is the reference and the "**yellow**" curve is from the `SpectreRF Engine`. The '`yellow`' curve is always greater than zero because of its amplitude output.

42. Switch back to the `Virtuoso ADE Explorer` window with the `COSIM -> tutorial2 -> maestro` view and use the '`Direct Plot Form`' (by navigating from the `maestro` view to `Results -> Direct Plot -> Main Form…`) as shown in Figure 86 to check the "`harmonic time`" output of node '`OUT`'.

Figure 86: Invoking 'Direct Plot Form' from ADE Explorer



43. Set the '`Direct Plot form`', as shown in Figure 87. Under the '`envlp`' Analysis:

    a. Choose '`Voltage`' Function,

    b. '`Select`' type as '`Net`',

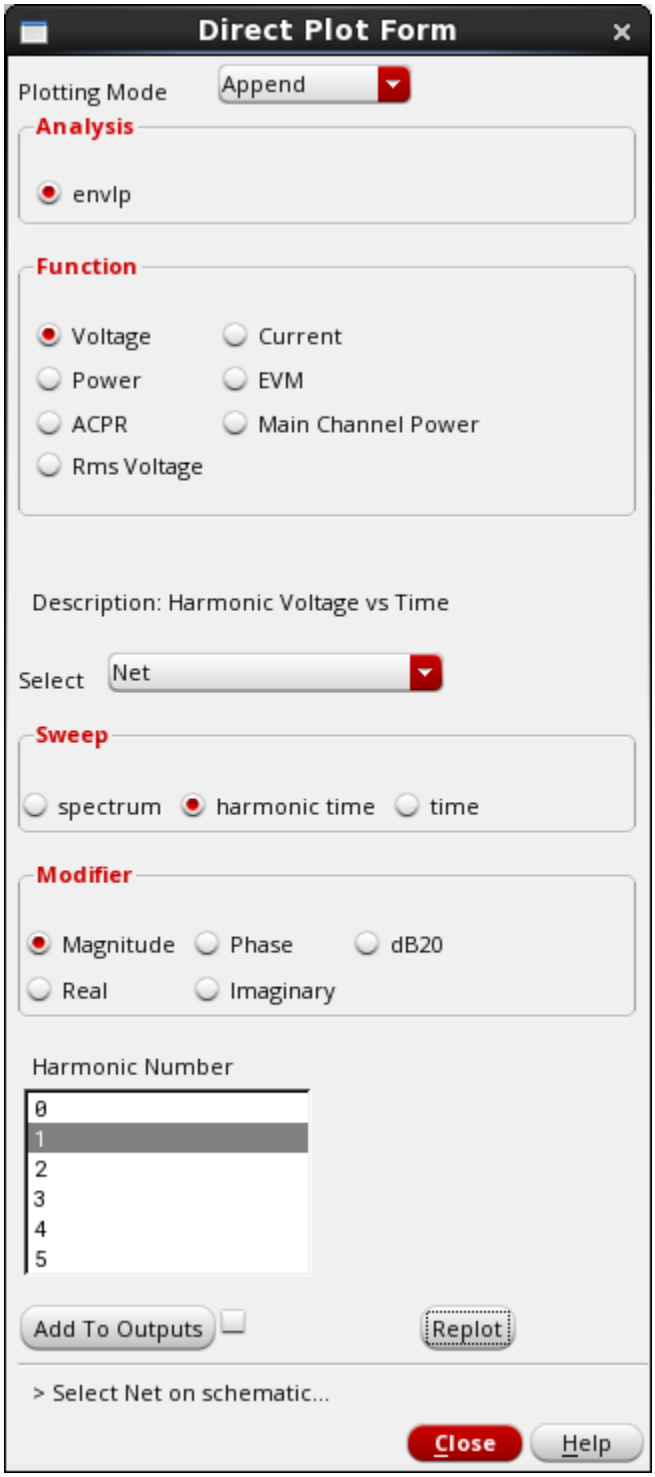
    c. '`Sweep`' field as '`harmonic time`',

    d. Set '`Modifier`' as '`Magnitude`'

    e. Set '`Harmonic Number`' as "`1`"

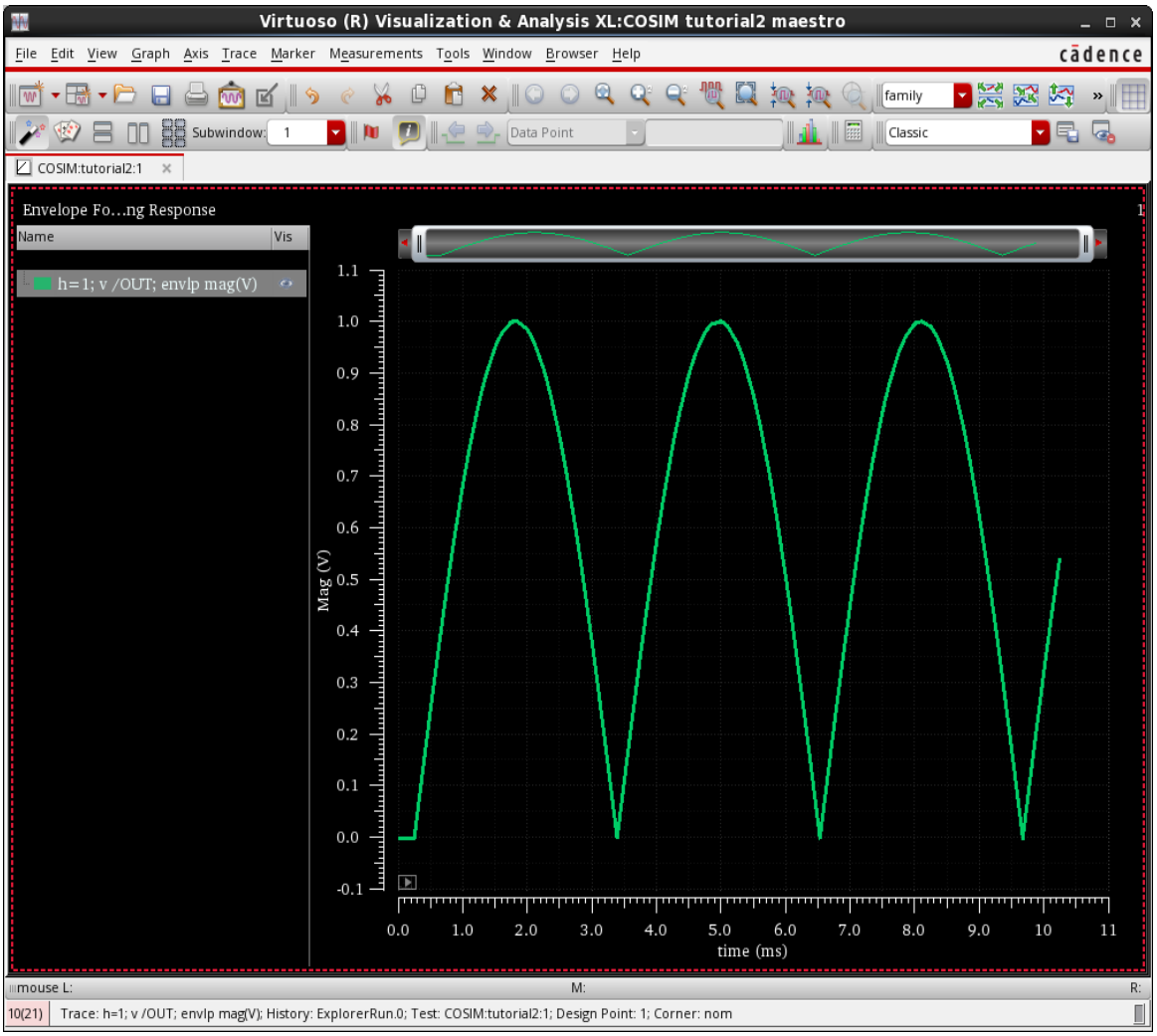
    f. Select the '`OUT`' net from the `tutorial2` '`schematic`' and click the Plot/Replot button.

Figure 87: Direct Plot Form after 'envlp' analysis to plot ‘OUT’ net



You will find that the 'OUT' waveform of first harmonic output is same as the one displayed by the MATLAB Simulink 'Scope' block inside 'env_d' testbench as shown in Figure 88 below of Cadence ViVA XL window.

Figure 88: 'OUT' net plot from 1st Harmonic in ViVA XL



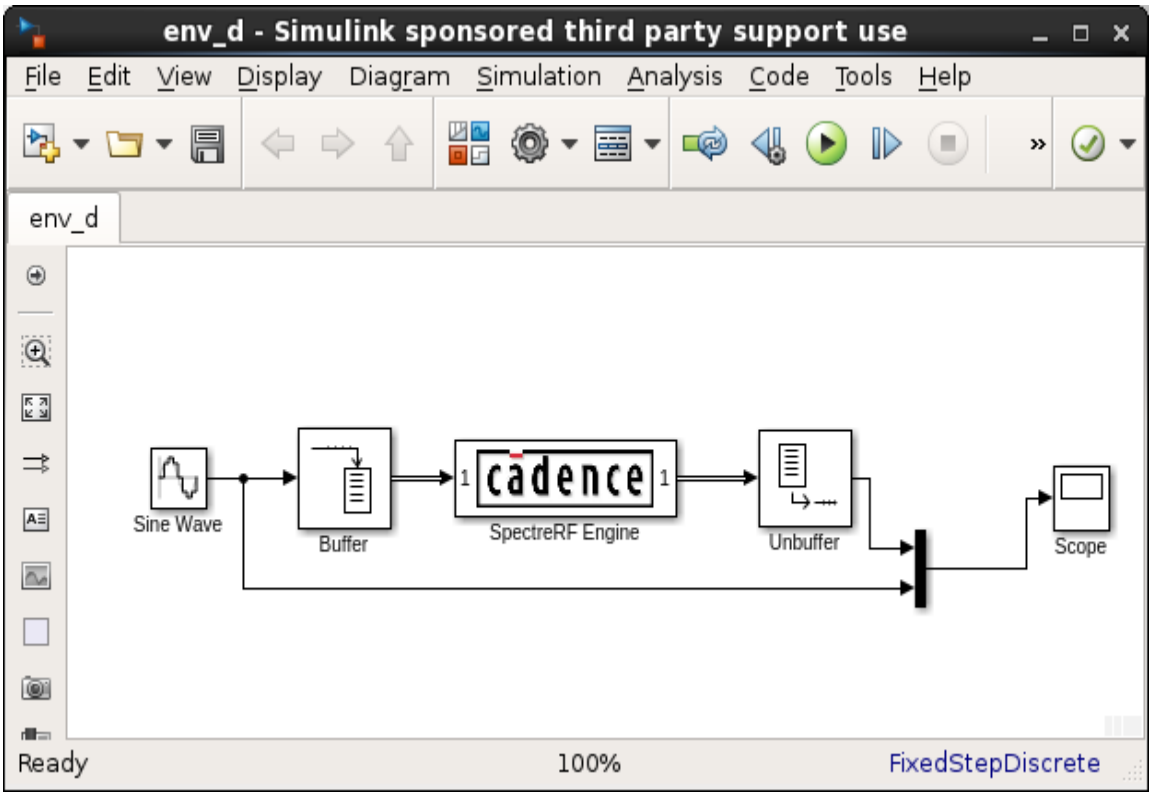
44. Close the ViVA XL window and 'Direct Plot Form' when done.

Following steps in this section describes the behavior of Matlab/Simulink 'env_d' testbench when the buffer components ('Buffer1' and 'Unbuffer1') are NOT used in the bottom reference chain.

45. With the already opened Simulink 'env_d' testbench from above steps, now delete the 'Buffer1' and 'Unbuffer1' blocks from the bottom reference chain and join the remaining connections.

The resulting `Matlab/Simulink` '`env_d`' testbench now looks as shown in Figure 89.

Figure 89: Modified 'env_d' testbench without Buffer1 and Unbuffer1 blocks

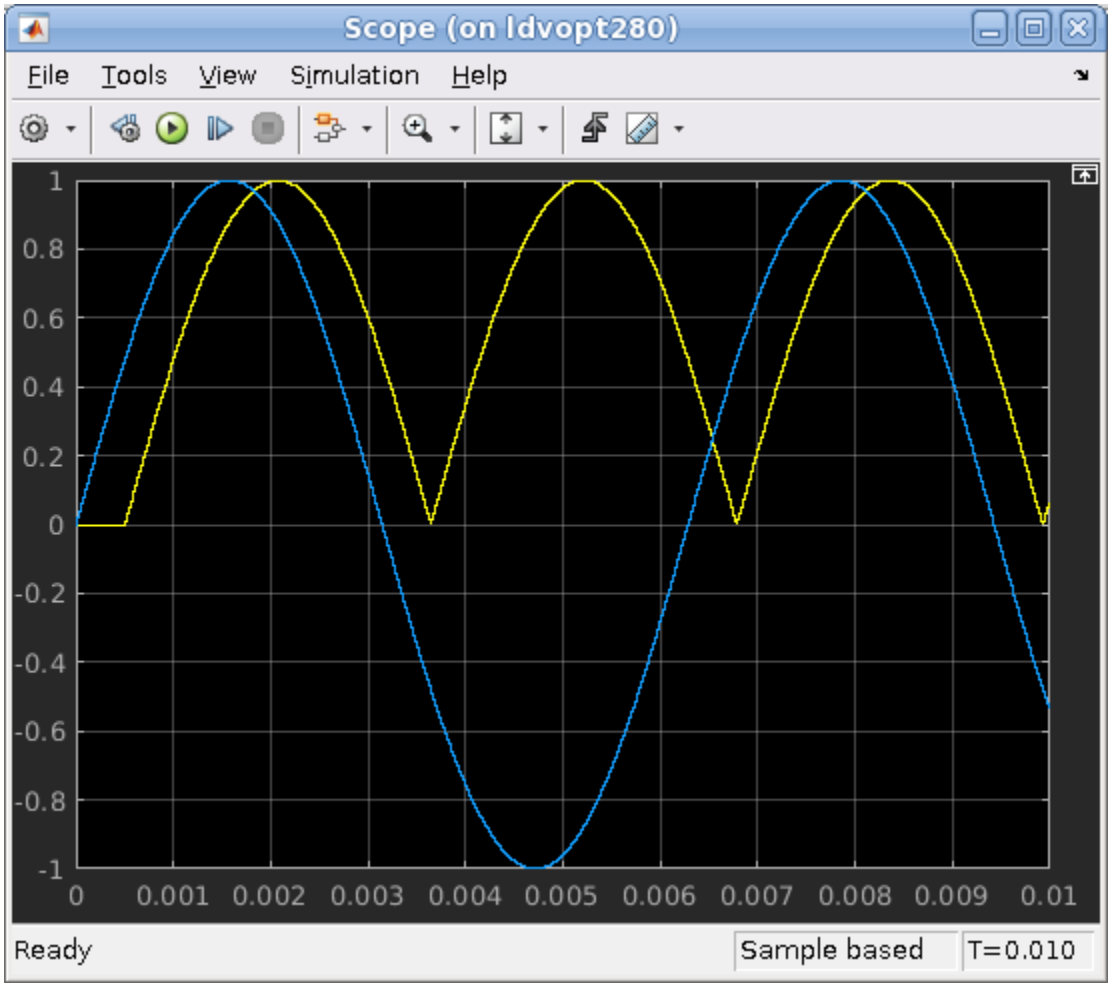


46. Run the cosimulation by hitting the green Run button (or) clicking on the `Simulation -> Run` option from the `Tutorial2` '`env_d`' Simulink testbench, and immediately pressing the green '`Run Simulation`' button from the right edge (or) `Simulation -> Netlist and Run` option of `Virtuoso ADE Explorer` window.

47. Double-click on the '`Scope`' block from the modified `Matlab/Simulink` '`env_d`' testbench. This will open a plot window as shown in Figure 90.

    The simplified '`env_d`' setup now having only the '`Buffer`' and '`Unbuffer`' blocks introduce delay in the circuit.

Figure 90: Result of envelope cosimulation analysis with modified 'env_d' setup of 'OUT' signal from 1st Harmonic

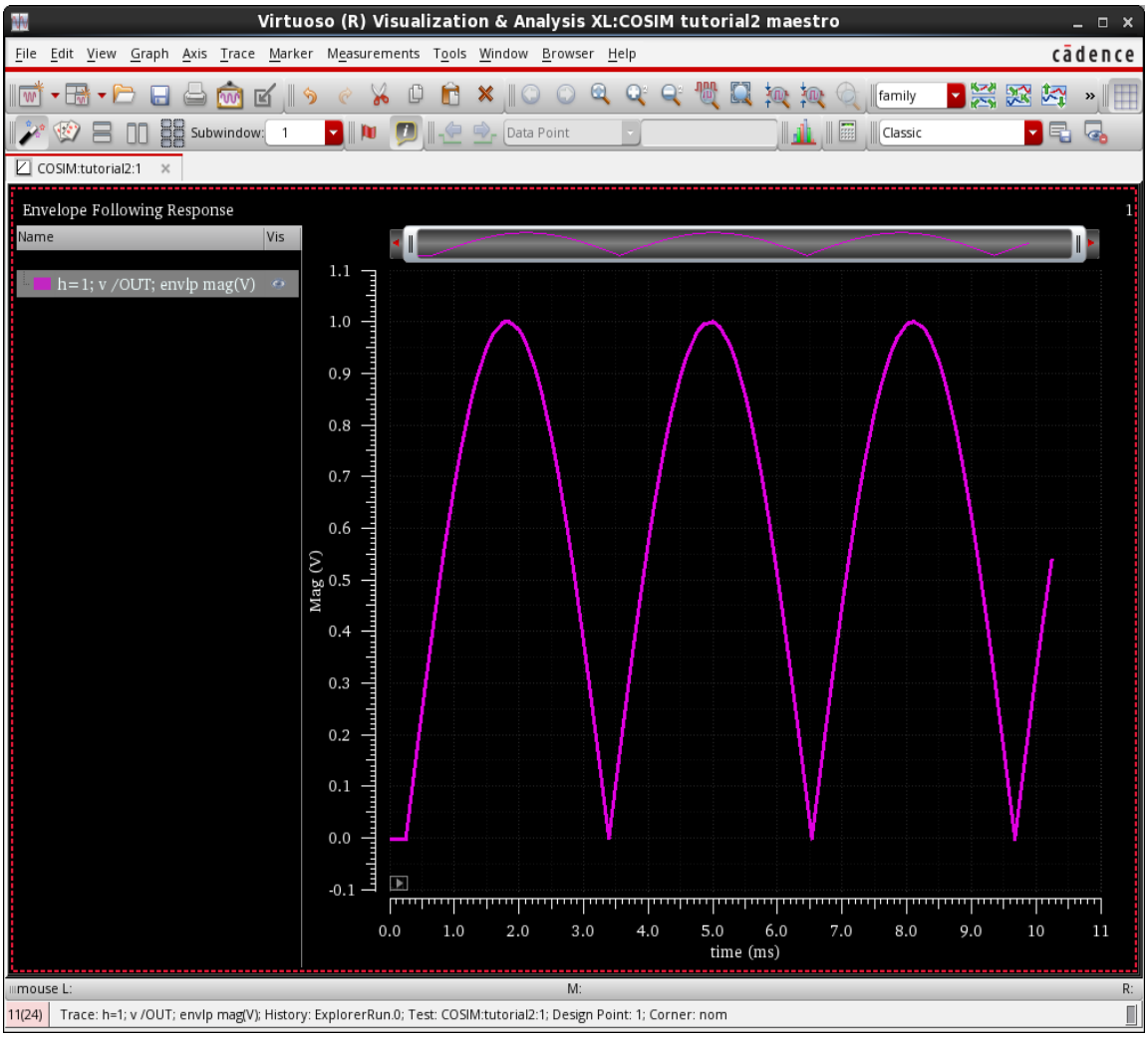


48. Plot the output in `ViVA XL` browser by switching back to the `Virtuoso ADE Explorer` window and launching '`Direct Plot Form`' to check the "`harmonic time`" output of node '`OUT`'.

    Keep the settings for the '`Direct Plot Form`' same as shown in Figure 87 (setup described in Step 43).

    You will find that the '`OUT`' waveform of first harmonic output is same as the one displayed by the MATLAB `Simulink` '`Scope`' block inside '`env_d`' testbench as shown in Figure 91 below of `Cadence ViVA XL` window.

Figure 91: Result of first harmonic at 'OUT' node without delay in 'Reference' chain



49. Close the `ViVA XL` window and '`Direct Plot Form`' when done.

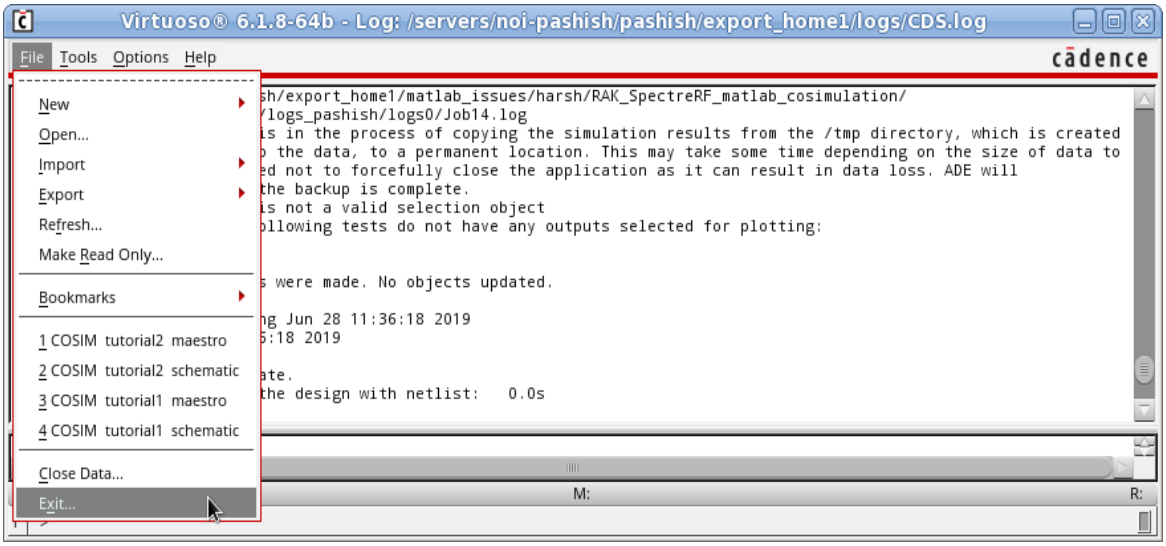
50. Do Not '`Save`' the changes to the modified '`env_d`' `Tutorial2` Simulink testbench.

    If you have already saved the changes, revert to the original `Tutorial2` '`env_d`' testbench by restoring the deleted blocks '`Buffer1`' and '`Unbuffer1`', in order to follow the steps mentioned in the next section.

51. Close all the `Matlab/Simulink` related graphs (if any), its associated `Tutorial2` '`env_d`' testbench, and exit from Matlab environment.

52. Save the `Virtuoso ADE` '`maestro`' view of COSIM '`tutorial2`' by clicking on the '`Save`' icon  and exit the `Virtuoso` session by navigating from the Virtuoso CIW to `File -> Exit…` as shown in Figure 92.

Figure 92: Exit from Virtuoso session



Second use Mode

Setting up the cosimulation from Standalone SpectreRF

If you have already generated a netlist using `Virtuoso ADE` for the COSIM '`tutorial2`' `env_d` testbench, skip this section and go directly to the next section, '`Running the cosimulation with Standalone SpectreRF`'.

53. Ensure that you are in the '`SpectreRF_simulink_`example' directory, as shown.

```
Unix> pwd

/…/home/…/RAK/SpectreRF_simulink_example
```

If not, then navigate to this directory.

54. From this directory, launch `virtuoso`, using the below command.

```
Unix> virtuoso &
```

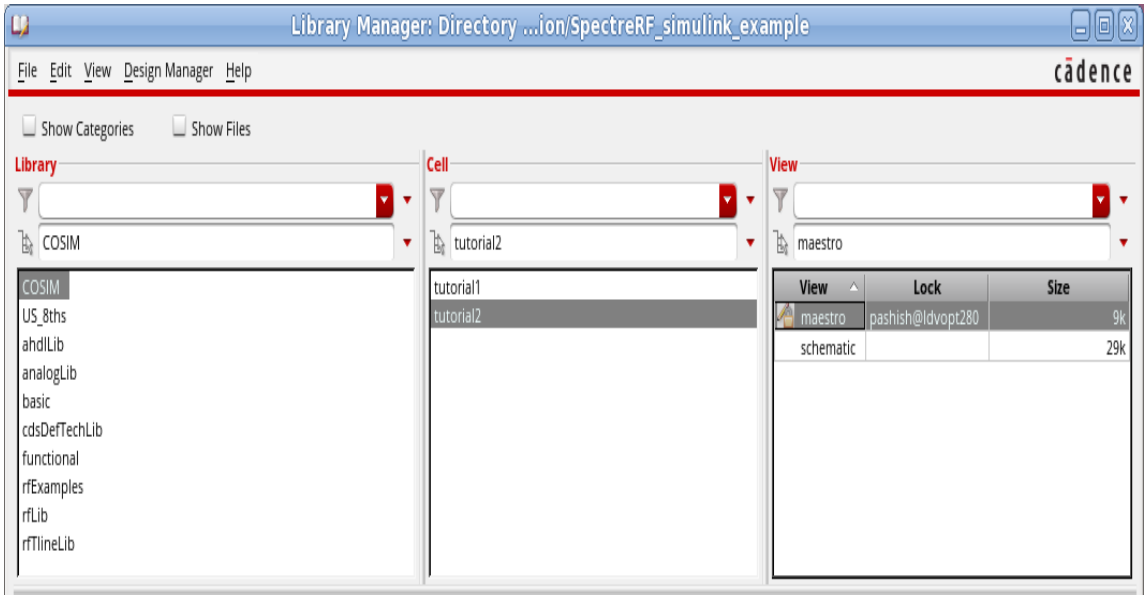
55. From the `Library Manager`, navigate to `COSIM -> tutorial2`, and open its '`maestro`' view, as shown in Figure 93.

    `Library -> COSIM`

    `Cell -> tutorial2`

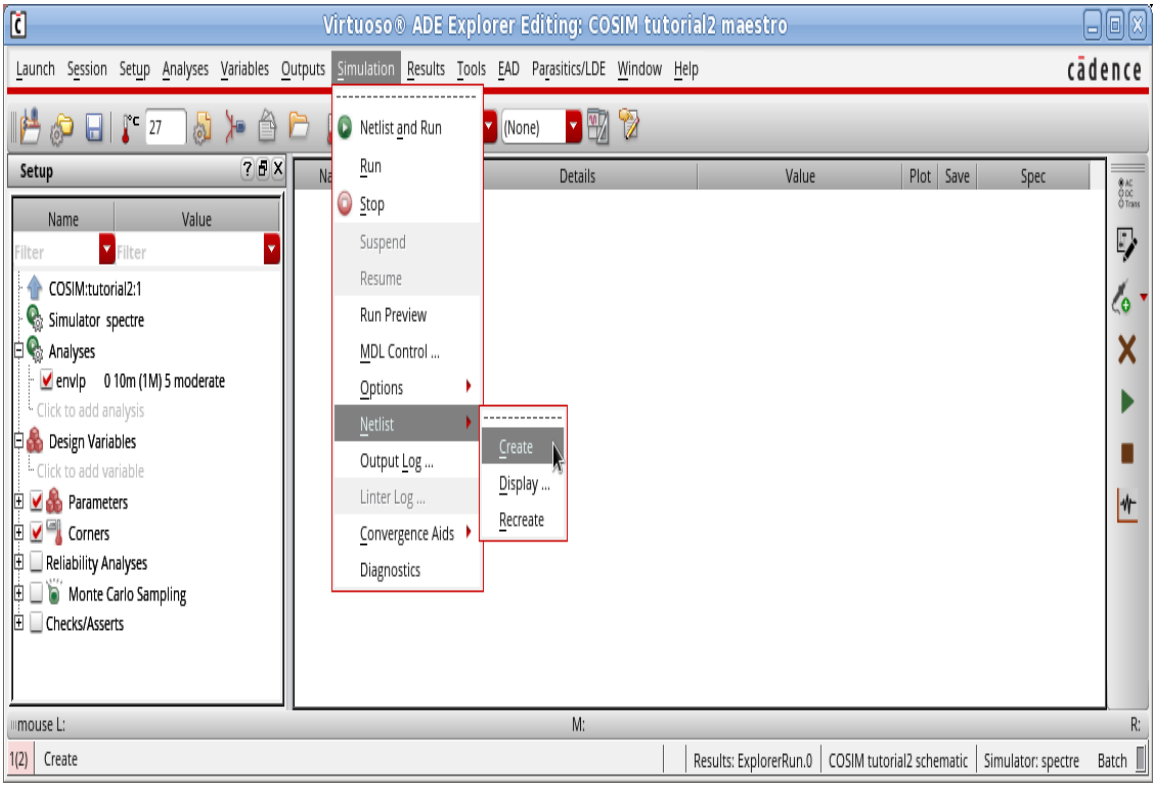
    `View -> maestro`

Figure 93: Opening 'maestro' view of COSIM tutorial2



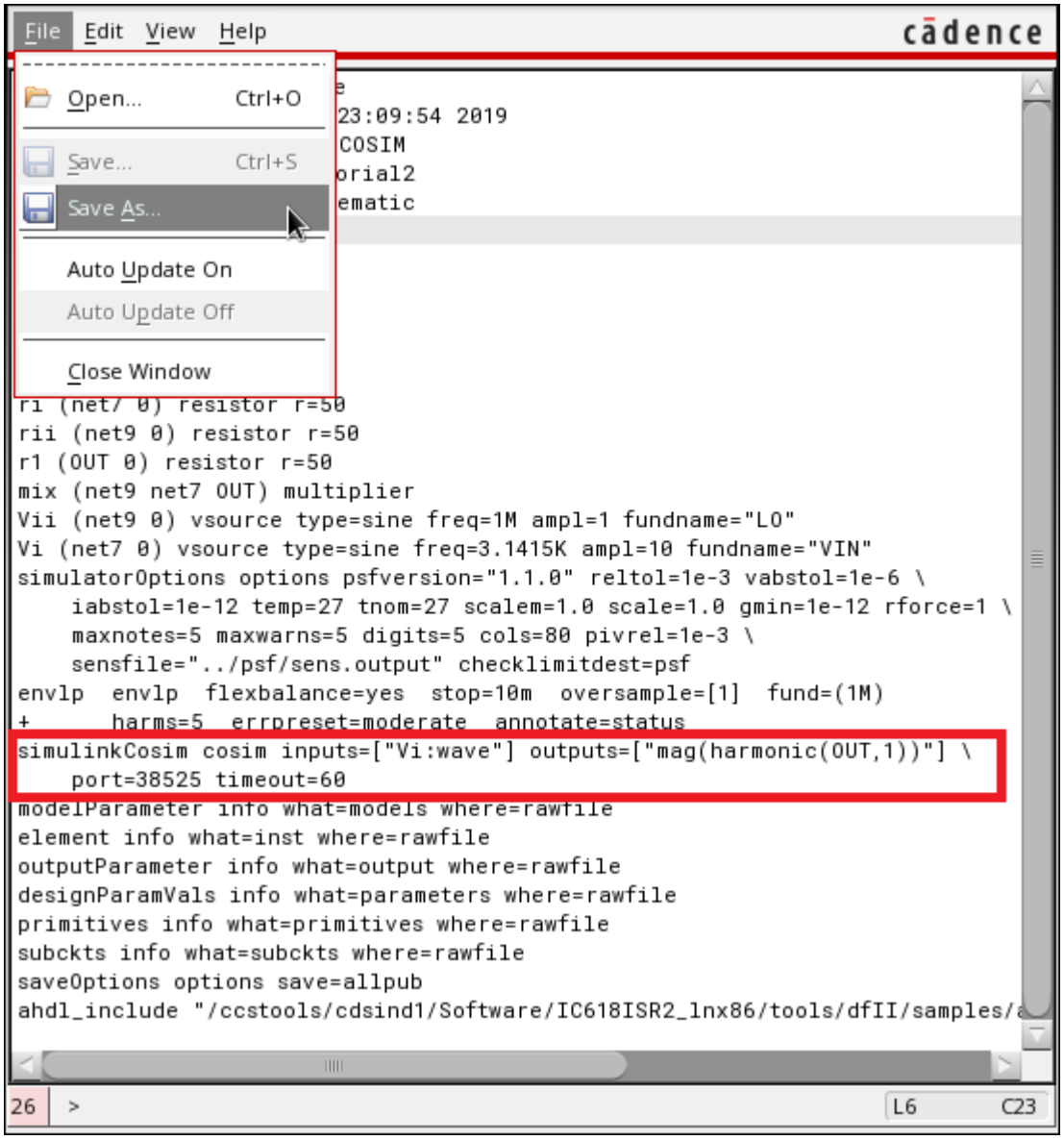
56. From the `ADE` '`maestro`' view generate the netlist for its corresponding design by navigating to `Simulation -> Netlist -> Create…` (or) just perform `Simulation -> Netlist -> Display` as shown in Figure 94 and save the current netlist with the name `tutorial2.scs`, by performing `File -> Save As…` as shown in Figure 95 in your current working directory.

Figure 94: Generate the tutorial2 netlist of maestro view from ADE Explorer window



Note: Ensure that the saved netlist '`tutorial2.scs`' of `COSIM tutorial2` has the '`simulinkCosim cosim`' statement in it.

Figure 95: Saved tutorial2.scs netlist of COSIM tutorial2



57. Once done, close the netlist and the associated `Virtuoso ADE` '`maestro`' view (save the changes if any), and exit the Virtuoso session from CIW by performing `File -> Exit…`

58. Alternatively, you can also use the already available `tutorial2.scs` netlist kept inside the '`SpectreRF_simulink_example/tutorial2`' directory.

    Assuming you are already in the '`SpectreRF_simulink_example`' directory, from the above steps, navigate to the '`tutorial2`' directory inside it, as shown.

```
Unix> pwd

/…/home/…/RAK/SpectreRF_simulink_example

Unix> cd tutorial2
```

Inside ‘tutorial2’ directory, you would see the behavioral VerilogA module for multiplier model, along with the tutorial2.scs netlist, as shown.

```
Unix> ls

multiplier      tutorial2.scs
```

59. Open the netlist file tutorial2/tutorial2.scs with any editor, in edit mode and make sure that the ‘matlab cosim’ statement includes all the parameters, after the envelope ‘envlp’ analysis declaration, inside the netlist as shown.

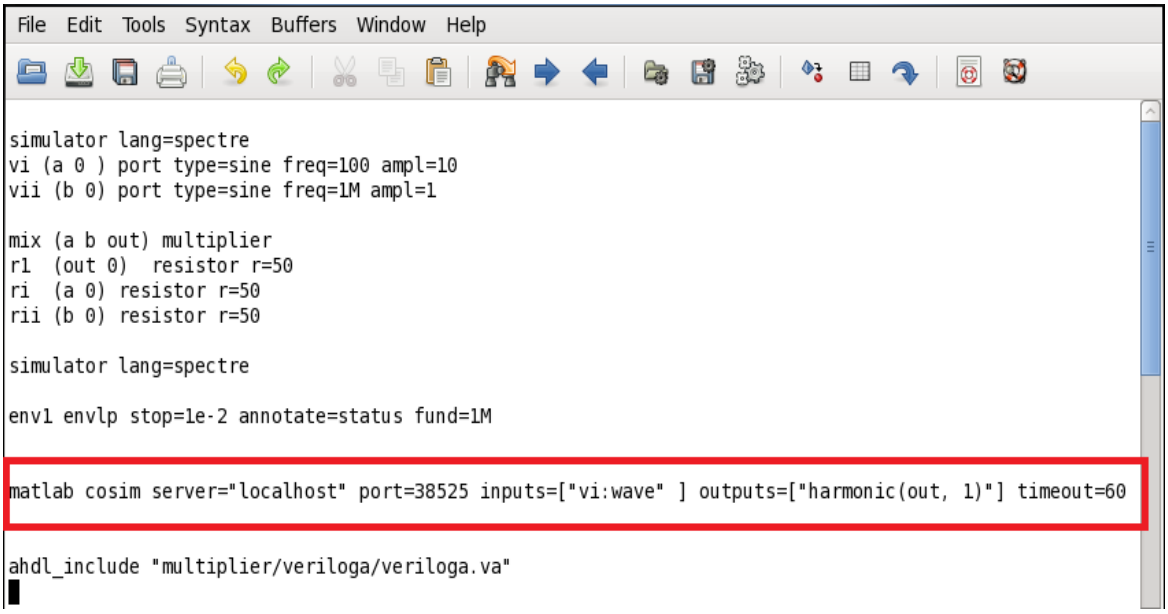
    For example,

```
matlab cosim server="localhost"  port=38525
inputs=["vi:wave" ] outputs=["harmonic(out, 1)"] timeout=60
```

    The tutorial2.scs contents are as shown in Figure 96.

    Here, the outputs=["harmonic(out, 1)"] parameter means the output is the 1st harmonic of the ‘OUT’ voltage.

Figure 96: tutorial2.scs netlist present inside 'tutorial2' directory



60. The other parameters for the 'matlab cosim' statement are already described in detail in earlier sections.

    Note: The netlist 'tutorial2.scs' present inside the 'SpectreRF_simulink_example/tutorial2' directory might be slightly different in netlist parameters, analysis or simulator options, as compared with the already saved netlist 'SpectreRF_simulink_example/tutorial2.scs' from the Virtuoso ADE Explorer window as described in earlier steps, but the functionality would be similar.

61. After performing the above steps, the netlist, 'tutorial2.scs' is now ready for cosimulation.

62. Navigate one level up from the existing 'tutorial2' directory to the 'SpectreRF_simulink_example' directory as shown.

    ```
    Unix> cd ../
    ```

Running the cosimulation from Standalone SpectreRF

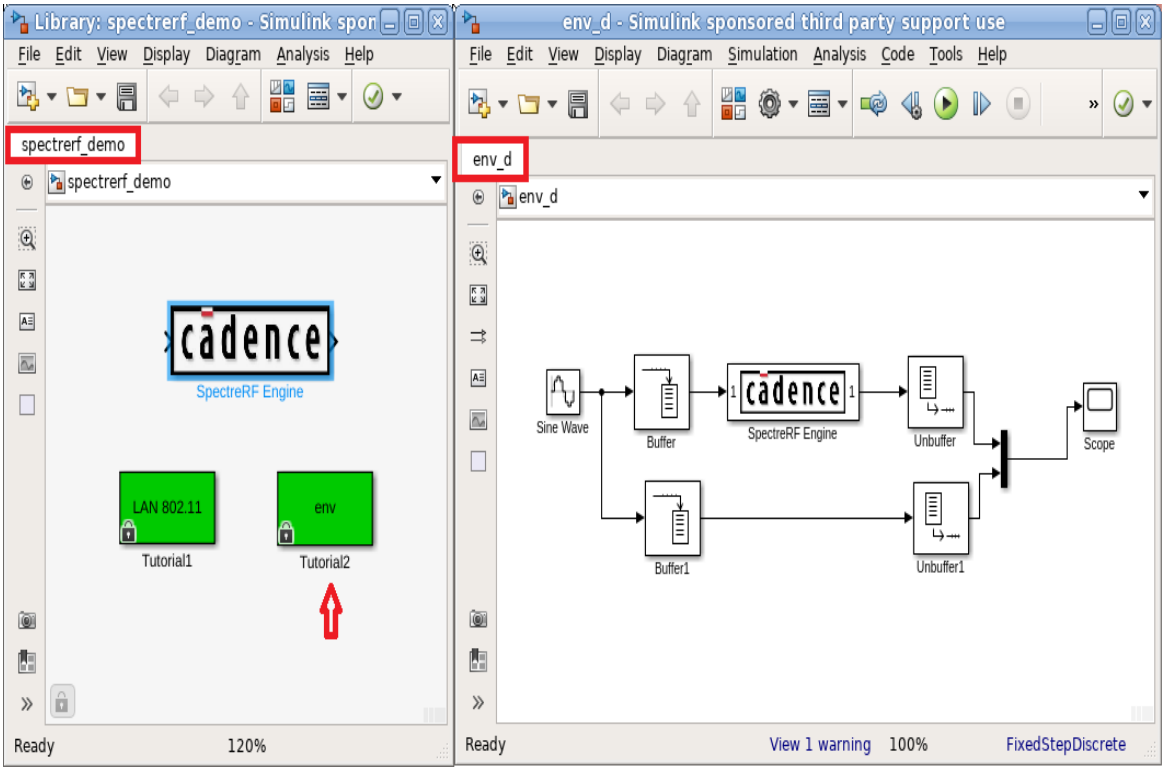
63. Open the Matlab/Simulink 'spectrerf_demo' Library, 'Tutorial2' testbench, for 'env' block having the SpectreRF Engine component instantiated, if you have not closed the already opened Matlab session from previous section 'Running the cosimulation from ADE'.

    If you closed Matlab from previous exercises, ensure that you are in the 'SpectreRF_simulink_example' directory, and launch matlab from there, using the below command.

    ```
    Unix> pwd

    /…/home/…/RAK/SpectreRF_simulink_example

    Unix> matlab&
    ```

    Open the Simulink design env_d by double-clicking the green box labeled 'Tutorial2' in the 'spectrerf_demo' Library having the SpectreRF Engine block as shown in Figure 97. This is the same design used in the 'First use Mode' section described above.

Figure 97: 'env_d' Simulink testbench of 'spectrerf_demo' library



64. Double-click the `SpectreRF Engine` block from '`env_d`' testbench. The `Block Parameters: SpectreRF Engine` form appears.

65. Type the following into the '`Spectre Command`' field of this `Block Parameters` form as shown.

```
spectre tutorial2/tutorial2.scs =log spectre.out
```

Note: In the '`Spectre Command`' field of this `Block Parameters: SpectreRF Engine` form, you can also enter the following command to use the already generated and saved netlist, '`tutorial2.scs`' from the `Virtuoso ADE` '`maestro`' view as shown in earlier '`Setting up the cosimulation from Standalone SpectreRF`' section.

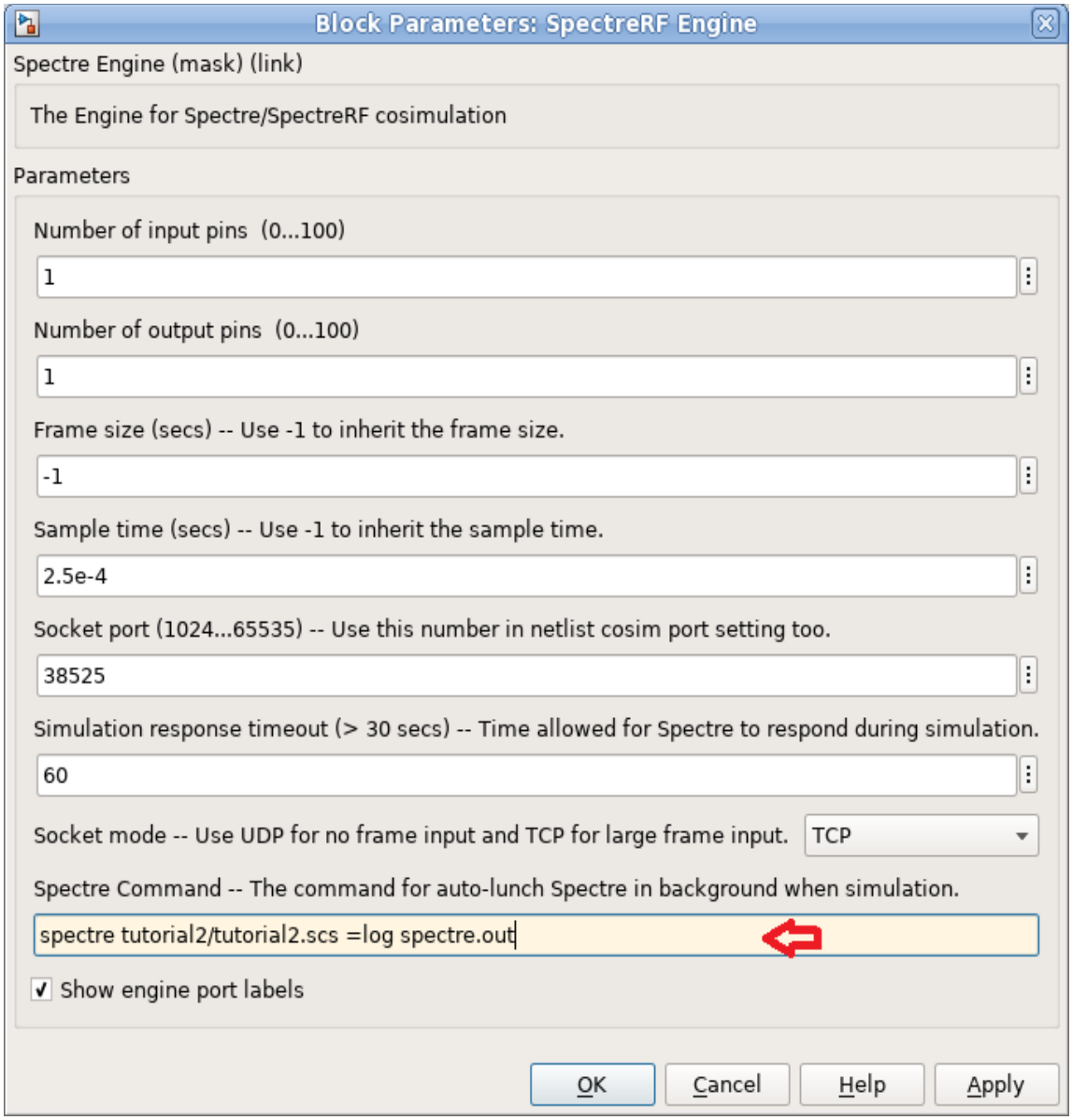
```
spectre tutorial2.scs =log spectre.out
```

66. Ensure that the `Socket port` field is set correctly to '`38525`' for this '`env_d`' testbench.

    Leave all the other `SpectreRF Engine` block properties set to their default.

The resulting '`Block Parameters: SpectreRF Engine`' form looks as shown in Figure 98.

Figure 98: Setting SpectreRF Engine block properties for Standalone cosimulation



67. Click '`Apply`' and then '`OK`' to close the `Block Parameters: SpectreRF Engine` form when done.

68. **(Optional):** You may save the above changes by clicking on the '`Save`' button  from the Simulink '`env_d`' testbench.

69. From the `Matlab/Simulink Tutorial2` '`env_d`' design testbench window, click `Simulation -> Run` or hit the green Run button.

    The MATLAB desktop issues the following message:

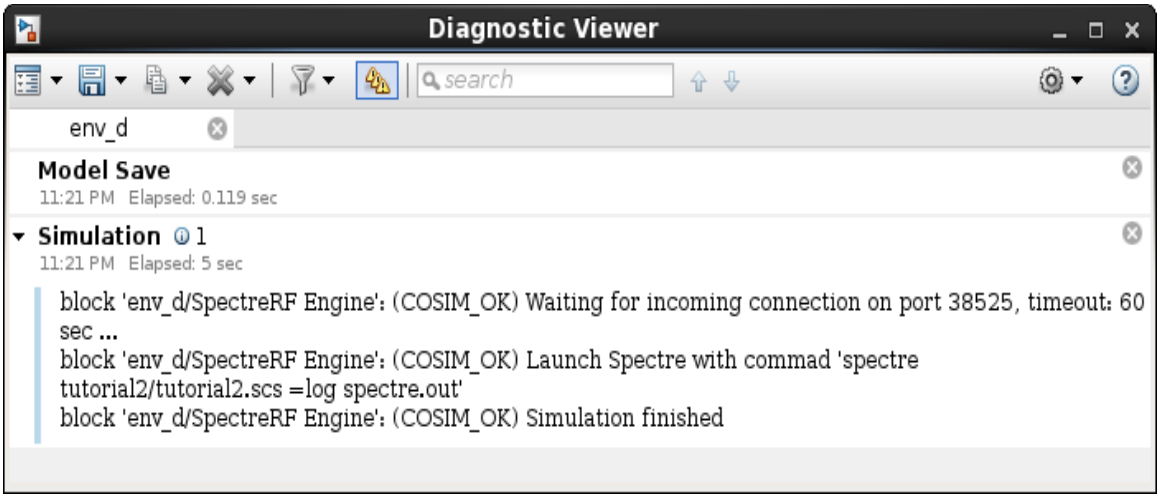
    ```
    block 'env_d/SpectreRF Engine': (COSIM_OK) Waiting for
    incoming connection on port 38525, timeout: 60 sec ...

    block 'env_d/SpectreRF Engine': (COSIM_OK) Launch Spectre
    with commad 'spectre tutorial2/tutorial2.scs =log
    spectre.out'
    ```

    The cosimulation immediately begins. The MATLAB desktop issues the following message (once the simulation runs to completion):

    ```
    block 'env_d/SpectreRF Engine': (COSIM_OK) Simulation
    finished
    ```

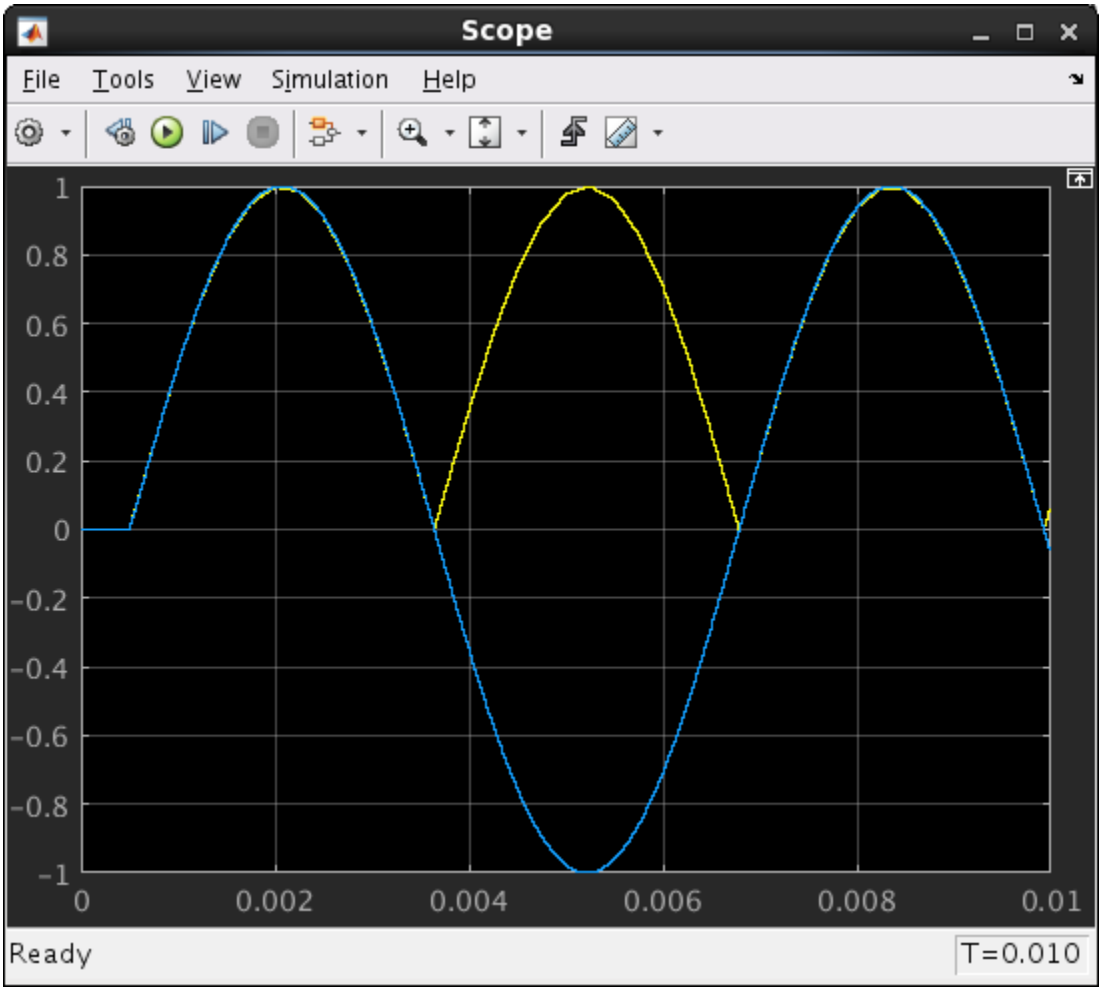
    The above message can be seen by navigating from the `Matlab/Simulink` window to `View -> Diagnostic Viewer` option as shown in Figure 99.

Figure 99: Diagnostic Viewer of Simulink for cosimulation logs



70. Once the cosimulation finishes, click the '`Scope`' output from the '`env_d`' Matlab/Simulink testbench, and ensure that the results are same as that produced from the '`First use Mode`' section discussed above, as shown in Figure 100.

Figure 100: 'env_d' Simulink testbench cosimulation results from Standalone SpectreRF



71. **(Optional):** You can also run the `Matlab/Simulink SpectreRF` cosimulation from Unix **command-line** by following the below approach.

    Assuming you are in the '`SpectreRF_simulink_example`' directory, enter the following '`spectre`' command on the `tutorial2.scs` netlist available inside the '`SpectreRF_simulink_example/tutorial2`' directory (having the '`matlab cosim`' statement) (or) the generated and saved `tutorial2.scs` netlist from the `Virtuoso ADE` '`maestro`' view (having the '`simulinkCosim cosim'` statement) as shown below, immediately after hitting the green Run button or clicking `Simulation -> Run` from `Matlab/Simulink` environment.

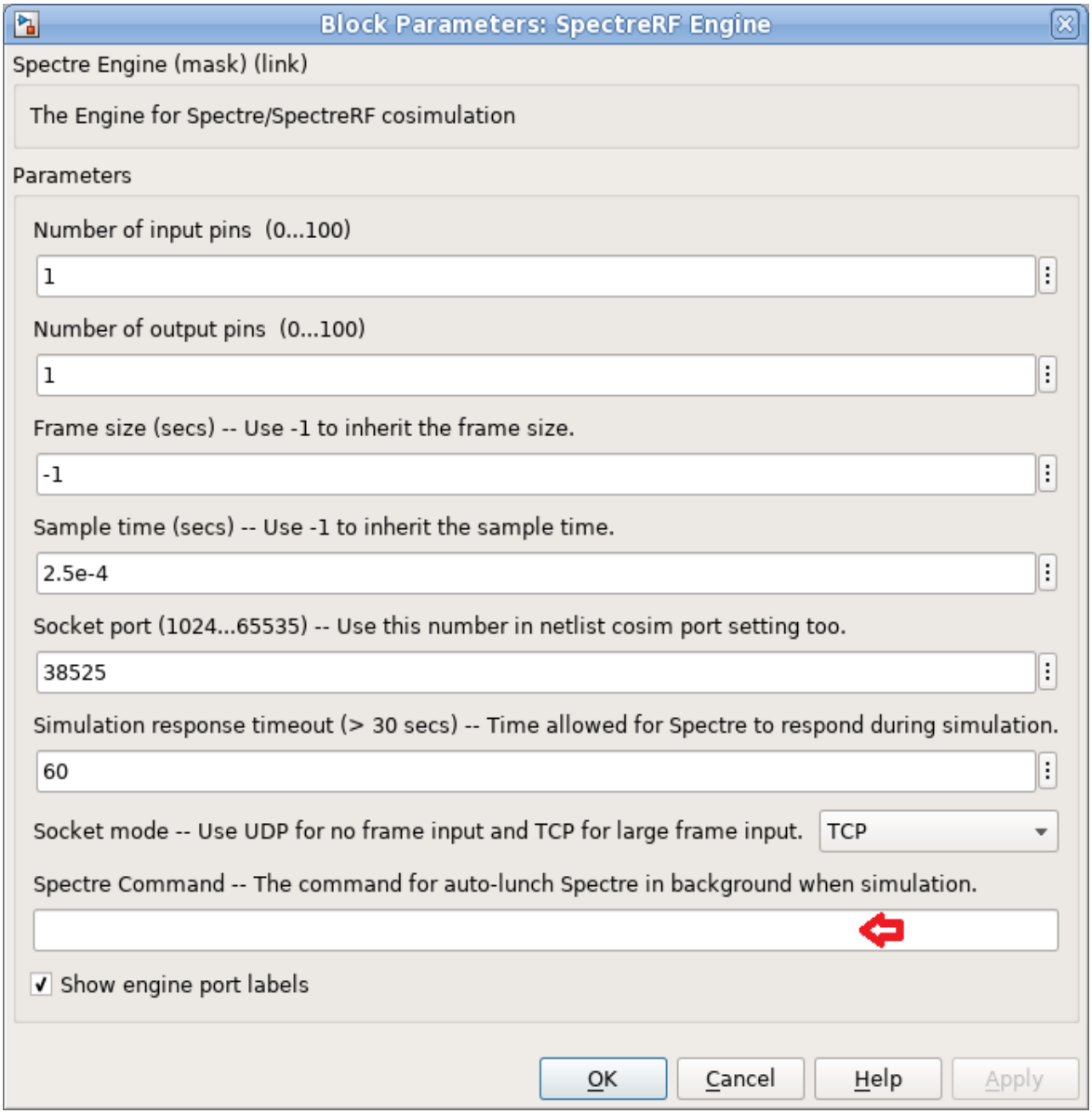
    ```
    Unix> spectre tutorial2/tutorial2.scs =log spectre.out
    ```

    (or)

```
Unix> spectre tutorial2.scs =log spectre.out
```

Note: Before running the cosimulation from Unix `command-line` method using the above '`spectre`' command, ensure that the '`SpectreRF Engine`' component inside the '`env_d`' `Tutorial2` Simulink testbench, has the '`Spectre Command`' field empty, and there is no '`spectre <netlist_file>…`' command entry present, as shown in Figure 101.

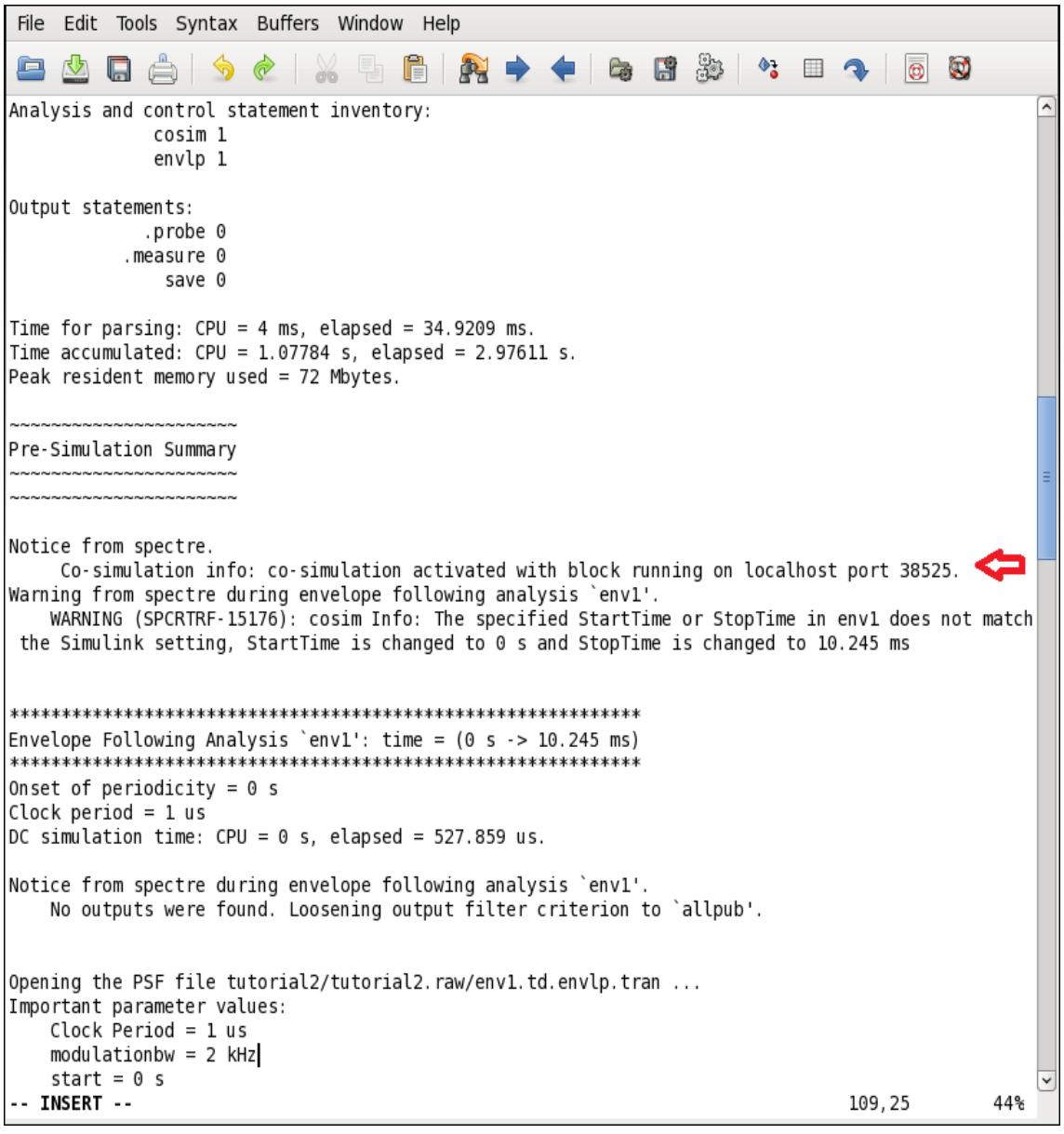
Figure 101: SpectreRF Engine component having 'Spectre Command' field kept empty in Unix command-line mode



72. Once the cosimulation completes, you would see a message as shown in Figure 102 generated in the spectre output log (`spectre.out`):

```
Co-simulation info: co-simulation activated with block
running on localhost port 38525.
```

Figure 102: spectre.out generated after cosimulation



73. Save the '`env_d`' `Tutorial2` Simulink testbench, by clicking on the '`Save`' button  or choose `File -> Save` option from the testbench.

74. Close all the `Matlab/Simulink` graphs, its associated 'env_d' `Tutorial2` testbench, and exit from Matlab environment.

The above **Tutorials** together illustrate the following three use modes:

1. For testing, if you need to modify both the System-level Simulink design and the Analog circuit, the use model described in '`Tutorial1`' is appropriate.

2. If you are an Analog designer who needs to validate a circuit with System-level design input and output, the '`First use Mode`' in Tutorial2 is a good choice.

3. If you are a System level designer, after specifying the `Spectre` command, you can use the `SpectreRF Engine` just like another block in Simulink.

Summary

In this tutorial, we have demonstrated how to setup and use a cosimulation link between MATLAB® and Simulink® system-level simulation environment with Cadence Virtuoso® ADE Explorer and Spectre® circuit simulator RF analysis (SpectreRF).

References

1. Spectre® Circuit Simulator and Accelerated Parallel Simulator RF Analysis in ADE Explorer User Guide

2. Virtuoso ADE Explorer User Guide

Support

Cadence Support Portal provides access to support resources, including an extensive knowledge base, access to software updates for Cadence products, and the ability to interact with Cadence Customer Support. Visit https://support.cadence.com.

Feedback

Email comments, questions, and suggestions to content_feedback@cadence.com.