cadence™

Analog Design Environment L Frequently Asked Questions

Product Version 6.1.6
November 2014

Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

Analog Design Environment L Frequently Asked Questions

This document contains the frequently asked questions and answers related to Analog Design Environment L.

How to create a device check expression for model or primitive?

You can use the Device Check Specifications form to create simple device check expressions such as `V(d)-V(s)` for a MOS device, or enable checks for devices rather than a specific instance that use a specific model.

See *Device Checking* section in the *Virtuoso Analog Design Environment L User Guide* for more information.

To create the expression for checking the selected device:

1. Select *Model* from the *Type* drop-down list box.

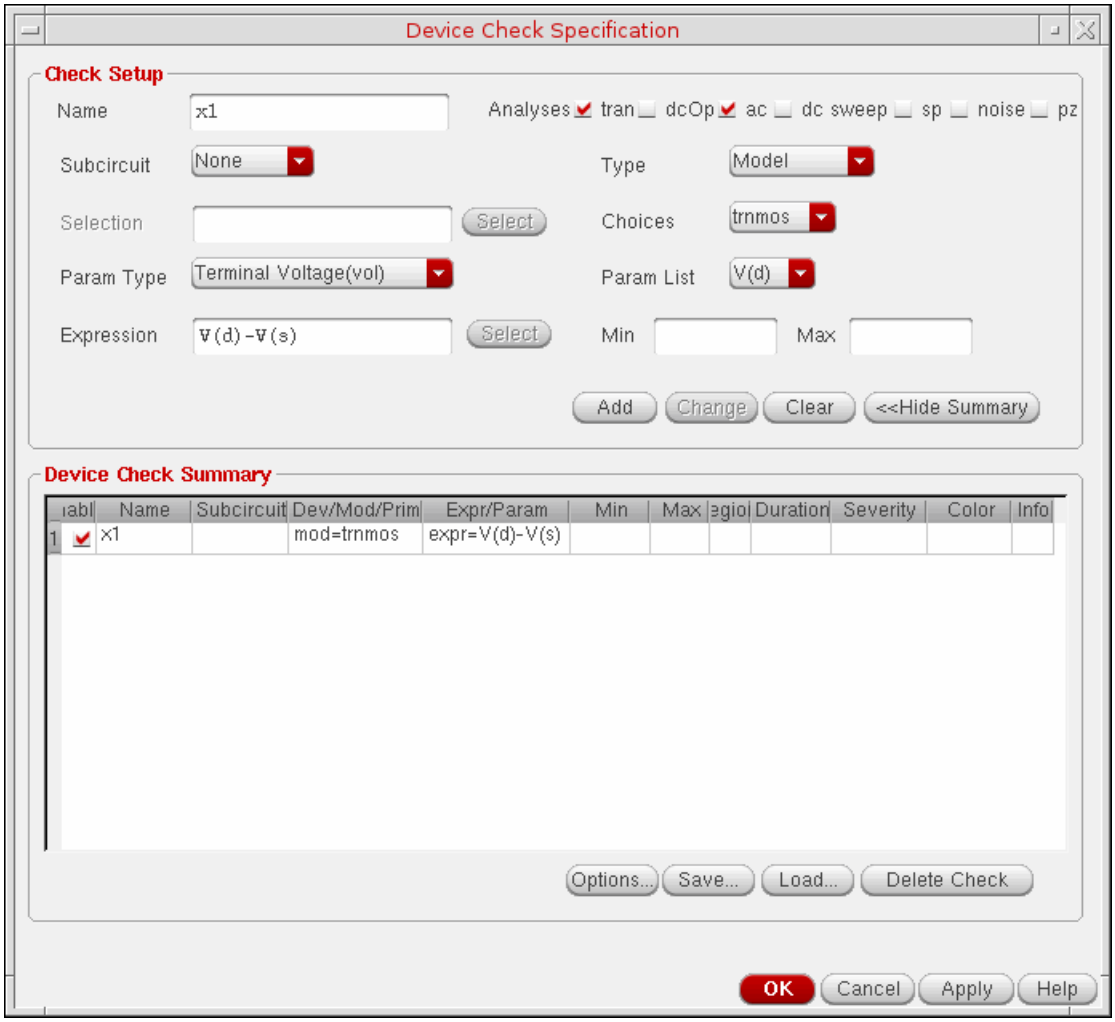
   Note: Select *Primitive* if you want to create the expression for primitive.

2. Select *Terminal Voltage(vol)* from the *Param Type* drop-down list box.

   The *Param List* drop-down list box lists all the available terminals.

3. Type the expression using the terminal voltages available under the *Param List* drop-down list box.

4. Click *Add*.



Note: If you are using a subcircuit model, then select the appropriate subcircuit from the *Subcircuit* drop-down list box and the corresponding model.

The added checks are displayed in the netlist as illustrated below:

```
check1 assert primitive=mos2 expr="V(d)-V(s)" min=0 max=10m
check2 assert sub=mynmos4 mod=trnmos1 expr="V(d)-V(s)" min=0 max=10m
```

Check1 above operates over all MOS instances that have the model of type `mos2`.

Check2 above operates on all NMOS instances that use the subcircuit model `mynmos4`.

How to run more than 10 OCEAN sessions at once?

By default, you can run a maximum of 10 simultaneous OCEAN sessions. If you try running more, the following error is displayed in the CIW:

```
Failed to lock log file: <Logfilepath>/CDS.log.9
*WARNING* file <logfilepath>/CDS.log.9 File is already locked by some other process.
```

This is because the `CDS_LOG_VERSION` environment variable is set to sequential, limiting the maximum simultaneous runs to 10. With the default setting, log files such as `CDS.log.1`, `CDS.log.2`, and so on are created for each OCEAN session. For more information, see the *Log File Environment Variables* section in the *Cadence Application Infrastructure User Guide*.

To run more that 10 OCEAN sessions simultaneously:

1. Specify the log file name with the OCEAN command.

   ```
   ocean -log logfileName file.ocn
   ```

2. Set the environment variable `CDS_LOG_VERSION` to `pid`.

   ```
   setenv CDS_LOG_VERSION pid
   ```

This will create log files with the process ID as the extension. Therefore, there is no restriction on the number of OCEAN sessions that can be run simultaneously.

How to get the list of output signals and expressions set using SKILL?

For an ADE L session, you can get the list of the outputs by using the asiGetOutputList SKILL function as illustrated below:

```
s=asiGetCurrentSession()
=>stdobj@0x1d14060

outs=asiGetOutputList(s)
=>(sevOutputStruct@0x15e71b8 sevOutputStruct@0x15e7208 sevOutputStruct@0x153ef48)

outs~>signal
=>("/out" "/net15" nil)

outs~>expression
=>(nil nil ymax(VT("/net15")))
```

For ADE XL, there is no direct SKILL function to get the list of outputs. However, you can write SKILL code to get a list of outputs defined for each test present in the current ADE XL session and return the signals or expressions listed for all the tests. Currently, the `asiGetOutputList` SKILL function does not list test names of the corresponding outputs.

To get a list of outputs for ADE XL:

1. Load the following procedure in the CIW:

```
procedure(CCSGetOutputList()
let((finalList sessId axlCv tests testId testSes)
sessId = axlGetWindowSession(hiGetCurrentWindow())
axlCv = axlGetMainSetupDB(sessId)
tests = axlGetTests(axlCv)
foreach(test cadr(tests)
testId = axlGetToolSession(sessId test)
testSes = asiGetSession(testId)
finalList = append(finalList asiGetOutputList(testSes))
)
finalList
) ;let
) ;procedure
```

2. Open the ADE XL view for the design you wish to find the output list and enter the following procedure in the CIW:

```
outs= CCSGetOutputList()
```

   The following will return the output list structure:

```
(sevOutputStruct@0x1a0e5098 sevOutputStruct@0x1a0e50a8
sevOutputStruct@0x1b506868)
outs~>signal
=>("/out" "/net15" nil)
outs~>expression
=>(nil nil ymax(VT("/net15")))
```

How to merge design variables from multiple saved ADE L states?

You have two saved states and want to merge the design variables from both states. Currently, when you load a new state, all existing design variables get replaced by those of the newly loaded state.

To accomplish this, you can use the asiGetDesignVarList and the asiAddDesignVarList SKILL functions. For example, to add the design variables from state A to the ones in state B, do the following:

1. Load state A and type the following in the CIW:

```
design_var=asiGetDesignVarList(asiGetCurrentSession( ))
```

   This stores the list of design variables from state A into the `design_var` variable.

2. Load state B, and type the following in the CIW:

```
asiAddDesignVarList( asiGetCurrentSession( )  design_var )
```

This adds the design variables from state A to the design variables list of state B.

Is it possible to use the value function on parametric sweep results?

You can find a value of a signal at a particular point for a normal analysis with no sweeps. However, when you try to access a signal from a parametric sweep results, as shown in the example below, the function always returns `nil`.

```
x=getData("out")
i= sweepValues(time)
value(x i)
```

This is because the value function does not take `variable` as its second argument. Instead, use the sweepVarValues function for parametric sweep results as shown below.

```
selectResult('tran)
sweepNames()
=> ("TEMPDC" "Vsupply" "time")
sweepVarValues("TEMPDC")
=> (0 32)
```

If you want to output the values of a signal at a particular point into a file, you can use the following script as reference:

Note: This script prints the value of the signal `out` for every sweep variable value.

```
openResults('tran)
STATUS = selectResult( 'tran )
if( STATUS !=nil
    then   pun_file = outfile("OUTPUT_FILE" "a")
        j=0
        foreach(i sweepVarValues("time") println(i)
            if( (2*round(j/2) +0.2) >= j
                fprintf(pun_file,"+ %e %15.14e", i, value(getData("out") i))
            )
            j=j+1
        )
        fprintf(pun_file,"\n")
        close(pun_file)
    )
```

How to read and load design variables from a text file to ADE L window?

You can use the following procedure to load the design variables from a text file with `.il` extension, for example, *`loadVarFile.il`*:

```
procedure(CCSAddVarFmFile(fileName)
let((myPort myList myVar)
  myPort = infile(fileName)
  unless(myPort
     error("\n File does not exist or is not accessible. Specify a valid filename")
  ) ;unless
  while(gets(myVar myPort)
       myVar = parseString(myVar)
       myList = append(myList list(list(car(myVar) cadr(myVar))))
  ) ;while
  asiSetDesignVarList(asiGetCurrentSession() myList)
  close(myPort)
t
) ;let
) ;procedure
```

To read and load the design variables:

1. Load the above procedure from `.cdsinit` file or from the CIW using the following command:

   ```
   load "loadVarFile.il"
   ```

   Alternatively, you can directly put the procedure in `.cdsinit` file or CIW to load the design variables.

2. Create a file containing the design variable names and their values defined in the following format:

   ```
   CAP 100p
   Res 310k
   x 4.0
   z 4f
   ```

   Note: There should not be any comments in the design variable file. Also, there should be no space after the last line otherwise a warning message will be displayed in the CIW.

3. Load the design variable file from the CIW using the following command:

   ```
   CCSAddVarFmFile("./designVarList.txt")
   ```

   Note: Before executing the above command, ensure that the ADE session is open in the active window because the procedure uses the asiGetCurrentSession() function.

How to run DC or AC analysis in reliability simulation?

You cannot run DC or AC analysis alone in the reliability simulation. This is because the reliability simulation is dependent on time and therefore, transient analysis results must be available.

While running the simulation for transient analysis, the reliability simulator first runs the *fresh* simulation, followed by *stress* simulation to generate the aged model parameters at a number of stress intervals required for *age* simulations. However, if you have an aged model file, you can use it to run DC or AC analysis. The format of this aged model file is the same as the fresh or non-degraded transistor model except that the parameters are extracted from the degraded transistor in the aged model file. Each aged model file contains the parameter *Age*.

Note: You can generate the aged model file using third party model extractor tools like *BSIMProPlus*.

Suppose that you have two aged model files, `nmos0.mod` and `nmos1.mod`, generated for two different age values for the device *`nchan`* as shown below:

The file `nmos0.mod` contains the following:

```
.model nchan nmos level=49
* This file contains the fresh transistor model parameters
+ vth0=0.7 u0=450 tox=80
* Above line specifies SPICE parameters
*relxpert: +h0=5.0e4 nn0=0.4
*relxpert: +ecrit0=2.0e5 lc0=0.89
*relxpert: +age=0.001
* Above lines specify RELXPERT parameters
```

The file `nmos1.mod` contains the following:

```
.model nchan nmos level=49
* This file contains the transistor model parameters after 20 min stress
+ vth0=0.65 u0=430
* Above line specifies changed SPICE parameters
*relxpert: +age=0.5
```

Now, to include these models in the SPICE netlist file, you need to use `.ageproc` statement as shown below:

```
*relxpert: .ageproc nchan files=nmos0.mod nmos1.mod
```

After including the aged model file into your SPICE netlist, you can run DC or AC analysis on the netlist alone.

Note: Ability to run DC or AC analysis in Spectre-native reliability simulator from ADE XL GUI support was added from IC6.1.5 ISR10 release onwards. The ADE XL flow still requires you to add transient analysis along with DC or AC or both analyses.

How to retain the Schematic toolbar customization after invoking ADE L?

You have done some toolbar customization like showing toolbars such as *File*, *View*, *Create* and removed the remaining ones such as *Go*, *Search* from the schematic editor.

When choose *Launch – ADE L* from the schematic editor menu, all the toolbar customization is lost and the default view (with all toolbars visible) is restored.

This is because when you launch ADE L, the schematic cellview type changes to `analogArtist-schematic`, which re-initializes the menus and toolbars.

You can use the following procedure to retain your toolbar customization in the schematic window:

```
procedure(CCSRetainToolbarCust(arg)
let((toolList )
toolList = hiGetWindowToolbars(hiGetSessionWindow(arg->window))
foreach(item toolList
hiHideToolbar(item)
) ;forecah
hiShowToolbar(deFindToolbar("File" arg->window))
hiShowToolbar(deFindToolbar("View" arg->window))
hiShowToolbar(deFindToolbar("Create" arg->window))
) ;let
) ;procedure
```

Register this procedure for schematic view and `analogArtist-schematic` (view name of schematic after ADE is open) using the following steps:

1. Load the above procedure in the CIW.

   ```
   load "CCSRetainToolbarCust.il"
   ```

2. Register the procedure in the CIW as follows:

   ```
   deRegUserTriggers("schematic" nil nil 'CCSRetainToolbarCust)
   deRegUserTriggers("analogArtist-schematic" nil nil 'CCSRetainToolbarCust)
   ```

Now, all the toolbar customization in the schematic window will be retained even after launching ADE L.