SIEMENS EDA

AFS Support Kit

Asserts and SOA Violations in Spectre Syntax

SIEMENS

Outline

Objective

The objective of this Support Kit is to cover the assert and SOA statements in Spectre syntax when running DC and transient analysis.

By the end of this Support Kit, you should be able to do the following:

1) Use assert statements with mod, sub, subs, expr, and variable assignment
2) Use assert options such as **checklimitdest**, **checklimitfile**, **assertmode**, and **checktotalviolation**
3) Use **checklimit** to control some assert statements
4) Report built-in SOA using options such as **soa_report** and **warn**
5) Use asserts with AFS (eXTreme) XT

Included Files

You can download the Support Kit data from the Knowledge Base Article (KBA) and extract the tar file as follows:

```
tar xvf afs_asserts_soa_sk.tar

cd afs_asserts_soa_sk
```

Once extracted, you should see the following files and directories:

File	Description
`Answers`	Answers to the lab questions
`README`	Step-by-step Instructions to run the lab
`SPECTREINCLUDE`	Model file directory
`clean`	Script to remove all the unwanted files
`input1.scs`	Spectre Netlist
`input2.scs`	Spectre Netlist

Description

The Safe Operating Area (SOA) is one of the fundamental specifications for the designers working on circuits like power supplies and amplifiers. The SOA ensures that all desired devices and conditions fall under the chosen region of operation. In addition, it helps to assess the design of appropriate protection or the limitation of device performance.

The assert statement uses a user-defined range for a certain parameter or expression and generates messages when the value of that parameter/expression falls out of the defined SOA range and when it returns back to it. Assert statements are only supported for DC and Transient analyses.

AFS supports assert statements in Spectre syntax to monitor the variation of an expression or a parameter of a subcircuit(s), device(s), model(s), or primitive(s). Below, are examples of each:

assert1 assert sub=inv1 expr=”v(in,out)” min=0.4 max=1 level=error message=”violation 1”

assert2 assert dev=idiv.m* expr=”vgs” min=0.4 level=notice message=”violation 2”

assert3 assert mod=nmos expr=”vds<=2.2” level=warning message=”violation 3”

assert4 assert expr=”i(idiv.m1:2)>1u” level=warning message=”violation 4”

The destination and the file name to which the assert violations will be written are determined by using the simulator options **checklimitdest** and **checklimitfile** respectively. If **checklimitdest** is set to file (default) or “both”, then the violations will be written to the AFS log file.

simulatorOptions options checklimitdest=file | psf | both | sqldb   checklimitfile=”filename”

In this example, since the checklimitdest=both, all the assert violations are written to a file called MyChecks and to the violations files in the PSF directory:

simulatorOptions options checklimitdest=both checklimitfile=”MyChecks”

AFS also supports **checklimit** statements used to enable/disable assert statements for a certain duration during transient or DC. If one assert is enabled, the rest of the assert statements will be disabled by default.

All the assert statements can be enabled/disabled at once using the checklimit option **checkallasserts=yes|no** which can be overridden using the simulator option **dochecklimit=0**.

In this example, the Assert violations are written to the simulation log file and all the checklimit statements in the netlists are disabled:

simulatorOptions options checklimitdest=both dochecklimit=0

My_checklimit checklimit checkallasserts=no

It is important to know that assert warnings are different from built-in SOA (safe operating area) warnings that come from the SOA device model parameters such as **bv_max** or **vbe_max**.

To save only the built-in SOA violations to a separate file, use the options below. Please note, the SOA violations are saved to a file with the following default name: ***.soarp0**

simulatorOptions options soa_report=1 warn=1

To save both the built-in SOA violations and the Assert violations use the following options:

simulatorOptions options soa_report=2 warn=1 checklimitdest=both checklimitfile="filename"

A detailed summary of the common options used with the SOA rules is listed below:

Options		Description
Built-in SOA	soa_report	Write SOA warnings to a separate .soarpt file
	warn=1	Write all SOA warnings
Assert	checklimitdest = file|psf|both|sqldb	Destination for the SOA file
	checklimitfile = filename	SOA file name
	assertflag=0	Disable processing of all asserts and checklimit statement
	assertmode=1	Provide more info to the soa file such as peak, average. etc.
	checktotalviolation=1	Reports the total violation duration for the enabled assert statements.
checklimit	checkallasserts	Analysis option that switches off all the assert statements. Can be overridden by the simulator option dochecklimit.
	dochecklimit=0	Disable all checklimit statements in the netlist, overriding the checkallasserts=yes option for the checklimit statement in the netlist.

Please note, by default, AFS XT disables all the assert statements. To enable asserts, use the simulator option **assertflag=1** or set this option in the afs configuration file, **.afscfg**:

afs.option.assertflag=1

To set this flag in ADE, select:
Simulation → Options → Analog → Check → Enable asserts in eXTreme mode

Directions

In this Support Kit, you will get hands-on experience in using assert statements and reporting the Safe Operating Area (SOA) on the command line using Spectre syntax.

Exercise 1

In this exercise, assert statements are set on model parameters and subcircuits.

1. Open the netlist **“input1.scs”** using your preferred editor and get familiar with the assert statements.

```
//Assert Statements
assert_mod assert mod=nfet90 sub=nd21 param=region values=["subth" "triode" "sat"] \
level=warning message="VIOLATION: device is OFF"

assert_subs assert subs=["nd*" "amp"] expr="i(mp1:2)<=200u" \
level=warning message="VIOLATION: Id is lower than 200u!!"
```

   - What is the difference between the 2 assert statements?
   - What simulator options were used for the assert statements and the built-in SOA?

2. Run the simulation:

   afs input1.scs -f psfbin -o out1

3. Open the file **“out1.soarpt0”** using your preferred editor.

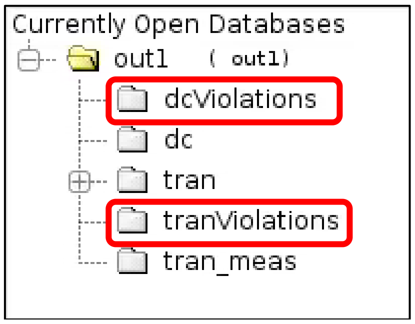
   - What instances and parameters were violated at 34.8ps?

4. Open the file **“assert_file1”** using your preferred editor.

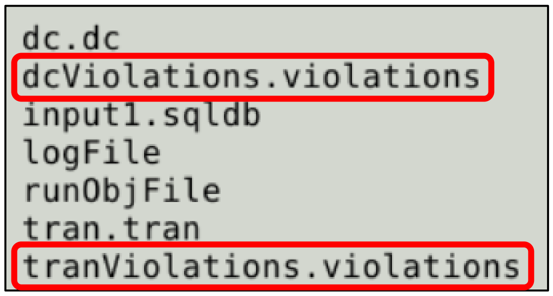
   - Are there assert violations in DC and Transient?
   - What instance(s) generated a violation warning at time=8e-15?

5. There are two violations files (**dcViolations.violations** and **tranViolations.violations**) that get generated in the output directory and report all the assert warnings for transient and DC. These files can also be viewed, sorted, and saved as a .csv file from EZwave.

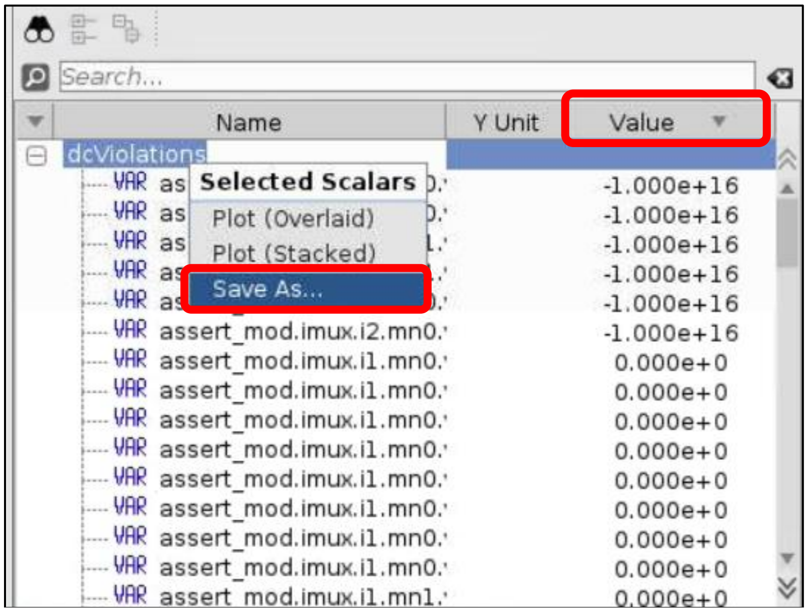
   ezwave out1 &



```
Currently Open Databases
outl    ( out1)
    dcViolations
    dc
    tran
    tranViolations
    tran_meas
```



```
dc.dc
dcViolations.violations
input1.sqldb
logFile
runObjFile
tran.tran
tranViolations.violations
```



Name	Y Unit	Value
dcViolations		
VAR as		-1.000e+16
VAR as		-1.000e+16
VAR as		-1.000e+16
VAR as		-1.000e+16
VAR as		-1.000e+16
VAR assert_mod.imux.i2.mn0.'		-1.000e+16
VAR assert_mod.imux.i1.mn0.'		0.000e+0
VAR assert_mod.imux.i1.mn0.'		0.000e+0
VAR assert_mod.imux.i1.mn0.'		0.000e+0
VAR assert_mod.imux.i1.mn0.'		0.000e+0
VAR assert_mod.imux.i1.mn0.'		0.000e+0
VAR assert_mod.imux.i1.mn0.'		0.000e+0
VAR assert_mod.imux.i1.mn0.'		0.000e+0
VAR assert_mod.imux.i1.mn0.'		0.000e+0
VAR assert_mod.imux.i1.mn1.'		0.000e+0

6. Close EZwave and all the open files

7. Remove the results files using the command below:

```
./clean
```

Exercise 2

In this exercise, circuit expressions and variable assignments asserts are examined. In addition, Checklimit statement is used to disable one assert statement.

1. Open the netlist **“input2.scs”** using your preferred editor and get familiar with the assert statements.

```
//Assert Statements
assert_cir_expr assert expr="v(out,ck_out)" min=0.5 max=1.2 \
level=error message="VIOLATION: V(out) - V(ck_out) is less than 0.5V or greater than 1.2V!!"

assert_var_ass assert expr="val1=i(iamp.mn0:d); val2=i(iamp.mn1:d); val1+val2" min=-550u max=550u \
level=warning message="VIOLATION:Out of Range"
```

2. Run the simulation:

   afs input2.scs -f psfbin -o out2

   - Did the simulation run successfully? Why?

3. Modify the netlist, **input2.scs**, and change the first assert statement, level=error to **level=warning**.

4. Save the changes and re-run the simulation.

5. Open the file **“out2.soarpt0”** using your preferred editor.

   - Does the file include SOA Violations and Assert Violations? Why?

6. Modify the netlist, **input2.scs** and uncomment the simulator option **“checktotalviolation=1”** on line number 193 and save the changes.

7. Re-run the simulation and review the file **“out2.soarpt0”**.

   - What are the total times for all the assert violations?

     Hint: look at the bottom of the file.

8. In the netlist **“input2.scs”,** uncomment line number 207 and save the changes.

   assert_control checklimit disable=assert_cir_expr

9. Re-run the simulation and reload the file **“out2.soarpt0”**.

   - What has changed in the assert violations section?

10. Close all the open files and remove all the results files using the command below:

    ./clean

Exercise 3

In this exercise, an assert statement is used with AFS XT (e**XT**reme).

1. Open the netlist **"input2.scs"** using your preferred editor and get familiar with the assert statements.

```
//Assert Statements
assert_cir_expr assert expr="v(out,ck_out)" min=0.5 max=1.2 \
level=error message="VIOLATION: V(out) - V(ck_out) is less than 0.5V or greater than 1.2V!!"

assert_var_ass assert expr="val1=i(iamp.mn0:d); val2=i(iamp.mn1:d); val1+val2" min=-550u max=550u \
level=warning message="VIOLATION:Out of Range"
```

2. Run the simulation.

   afs input2.scs -f psfbin -o out3 --xt green

   - Is the file **"assert_file2"** created in the working directory? Why?

3. In the netlist **"input2.scs",** uncomment the simulator option **"assertflag=1"** in line number 193.

4. Save the changes and re-run the simulation

   - Are the assert statements used? Why?

5. Remove all the results files using the command below:

   ./clean

Conclusion

In this Support Kit, we covered the following:

1) Using assert statements with mod, sub, subs, expr, and variable assignment
2) Using different assert options such as **checklimitdest**, **checklimitfile**, **assertmode**, and **checktotalviolation**
3) Using checklimit to control some assert statements
4) Reporting built-in SOA using options such as **soa_report** and **warn**
5) Using asserts with AFS (eXTreme) XT