SOA usage with new formalism

G. Bertrand / A. Kabalan

26/11/2014

life.augmented

Summary

- New SOA formalism
  - Introduction
  - New SOA Formalism Structure
    - Category Out Of Process
    - Category Non Functionnal
    - Category Reliability
    - Category Modeling
  - SOA Filtring

- Check SOA in Framework cadence
  - Allow SOA Check ELDO
  - Post Processing Result Of Eldo
  - Allow SOA Check SPECTRE
  - Post Processing Result Of Spectre
  - Allow SOA Check HSPICE

- AMSRB Tool
  - AMS Results Browser NEW in 13.2
  - Predefined Filter
  - Predifened Filter example
- Conclusion

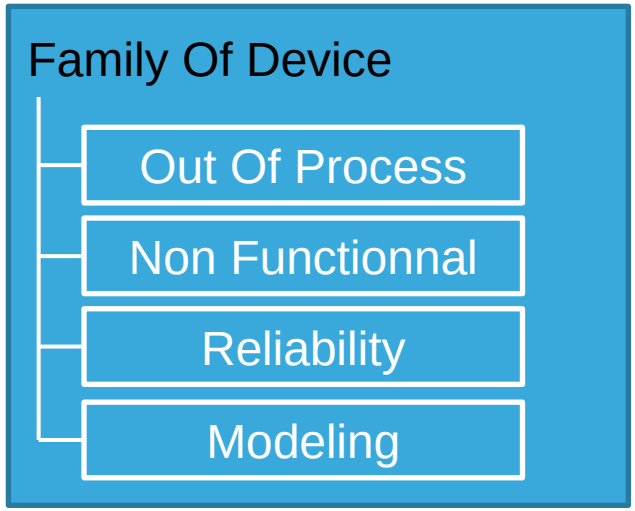
PART I

New SOA Formalism

Introduction

- SOA (Safe Operating Areas) can be defined within the simulator in order to check if the device is properly used.
- These can be viewed as rules that the simulator will evaluate during circuit simulation. If the rule check is false, a message is printed for the related component.
- The used rules are mainly based on device specification, electrical characterization and model limitation.
- Because SOA rules are numerous and concern a lot of device, soas have been classified in four categories.

New SOA formalism Structure



- Created 4 categories of rules for every family of devices
  - **Out Of Process**: Rules related to the standard process specification of the device
  - **Non functional:** They allow to check that the device is functional during all the simulation
  - **Reliability:** We could check that the device operates within the specification and is not broken and could evaluate the circuit reliability
  - **Modeling:** The hot spot issue underlines the fact that a device could operate in a safe domain for itself but could induce damage for other neighbor component

I-Out of Process Category

- This category contains all rules related to the standard process specification of the device.
- If we consider a 1.8V Mos device, we expect that the maximal Vgs and Vds is 1.8V. Such bias range do not rely on reliability or advanced electrical characterization and in consequence can be already defined at the device definition.

II-Non functionnal Category

- The rules present in these category are based on electrical characterization. They allow to check that the device is functionnal during all the simulation.
- The non functionnal limits could come from junction or device breakdown, or snap-back problem for MOS devices for example.

III-Reliability Category

- The rules here are also based on electrical tests, and mainly on the reliability program started at the maturity 10 of the device.
- With the previous categories, we could check that the device operates within the specification and is not broken. With this category, we could evaluate if the circuit is reliable for long time.

IV-Hot spots and modeling Category

- Simulation are based on a device model. The model should be as close as possible to the silicon behaviour of the device. Nevertheless, some limitations, inaccuracies could be present in some operating areas. The rules here will list some known limitations. In the case of HV devices or resistances for example, some models will not take into account the self-heating that is present.

- The hot spot issue underline the fact that a device could operate in a safe domain for itself but could induce damage for other neighbour component. Resistor self-heating for example is important for accurate modeling of the component but could also be crucial when considering that the resistor self-heating could increase the temperature in the back-end line above where electromigration is an important limitation.

SOA filtering

- Three level of filter have been created in order to address specific SOA validation.

- At device level : instance parameter soa
  The device has an instance parameter soa. When set at 0, it switched off all the soa rules for the selected device.

- At familly level : familly_soa switch
  familly_soa parameter can be set within ArtistKit interface or manually in the netlist in order to switch on(1)/off(0) the soas of the related familly.

- Global level : soa outofprocess, soa nonfunctionnal, soa reliability, soa modeling
  Four global parameters (list above) related to the categories could be set in the ArtistKit interface or manually in the netlist. They allow to switch on(1)/off(0) the different rules present in each category.

PART II

Check SOA in Framework Cadence

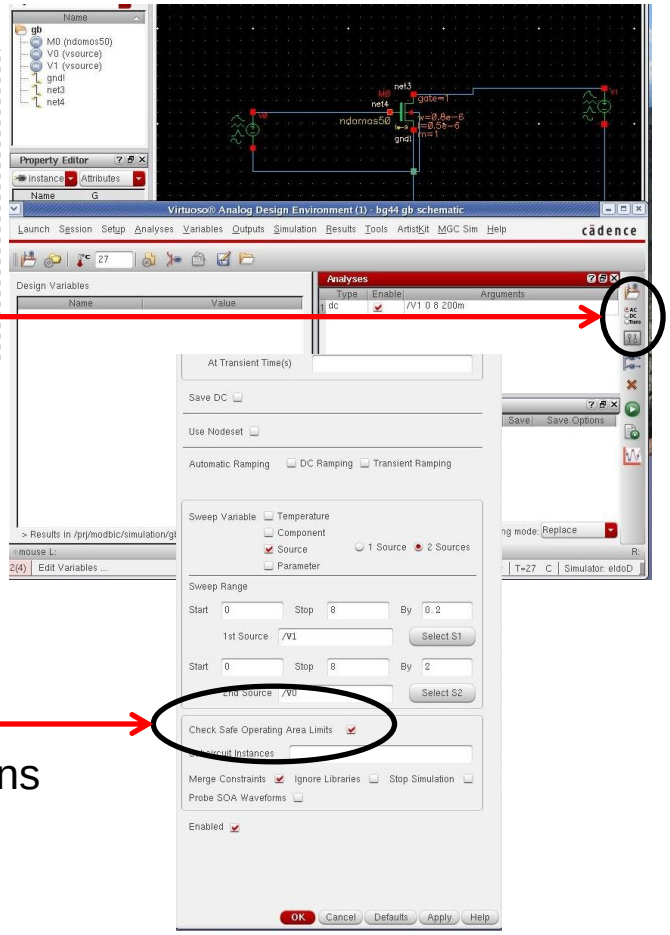
Allow SOA check (ELDO)

CheckSOA Statement **enable/disable** the check soa CheckSOA
➔ Default value is **disable**

How to enable/disable checksoa statment

Step 1: ADE>Choosing Analysis

Step 2: Select from the choosing analysis window «`check Safe Operating area limits`»



After Simulation:

- <netlist>.soa file created Under PSF directory which contains all violations results

Post Processing Result Of Eldo

Mentor Graphics Has developed AMSRB Tool to post processing the Soa Results

Results Display

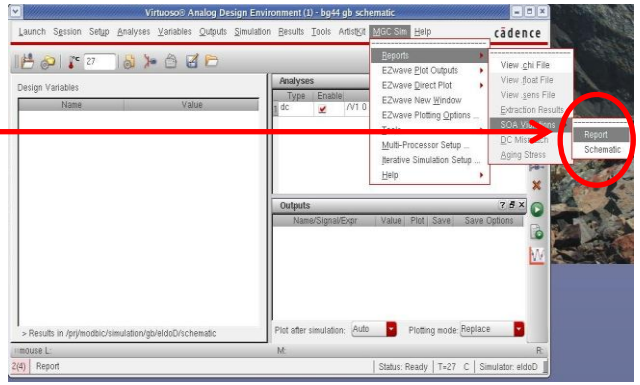
Step 1: Run the simulation

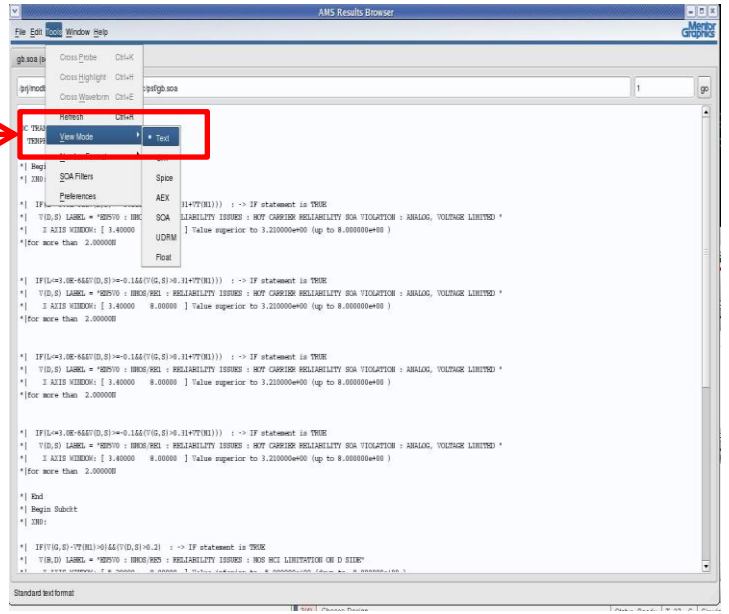
Step 2: ***MGC Sim>Reports>SOA Violations>Report***

`PS: (if you do not see this menu, look at the short cup next Cadence logo at the top rigth corner)`

Step 3: AMSRB Interface **file>open><netlist>.soa**

Step 4: Choose SOA **ModeTools>ViewMode>SOA**





AMSRB is started automatically after the simulation “to display the log file”

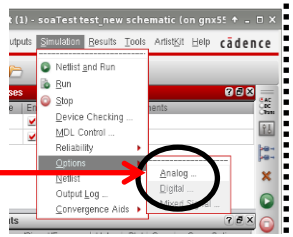
This tool help us to filter our results based on Label, models name, subckt name .....

Allow SOA check (Spectre)

All defined Assert statement in your models are enable by default

How to control the check

Step 1: ADE>Simulation>Options>Analog



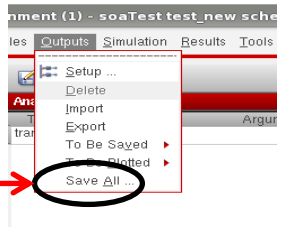
Step 2: Simulation Options>Check tab

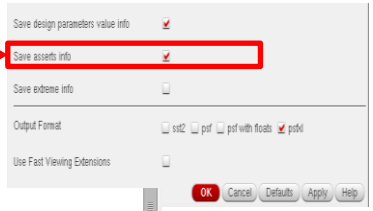
Step 3: `docheckilimit` to yes || no « enable/desable the check »

Step 4: `checklimitfile` To define the output file which contain all violations messages

Step 5: `checklimitdest` define your output destination :

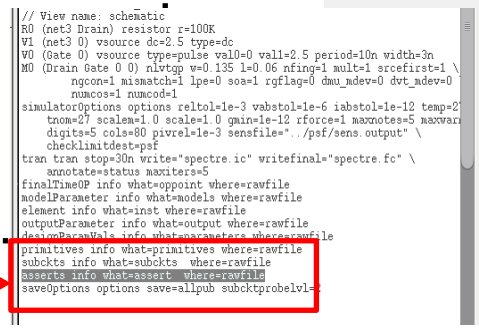
- File
- Psf
- Both





Step 6: Save all assert info:

1. ADE>outputs>Save All
2. Select Save asserts info
3. Verify if assert info is added into your netlist



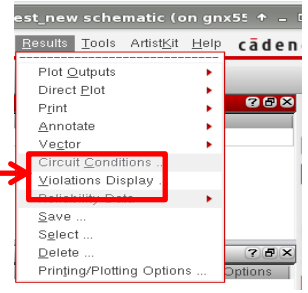
Post Processing Result Of Spectre (1)

Display the assert violations results

Step 1: Run your simulation

Step 2: Launch violations display ***ADE>Results>Violations Display***

Step 3: Violations display Interface



Violations Display (on gnx5591)

Violations Summary

	Name	Status	Subcircu	v/Mod/P	Expr/Param	Min	Max	Regions	Duration	Severity	Color
1	M0.soa_124	FAILED	nlvtgp		expr=((v(g)-v(s))-m1:vth>0&&(v(d)-v(s))>0.2)?((v(b)-v(d))<-1.2||(v(b)-v(d))>0.2):0				0	warning	yellow
2	M0.soa_123	PASSED	nlvtgp		expr=v(b)-v(s)	-2.2			2e-09	warning	yellow
3	M0.soa_121	FAILED	nlvtgp		expr=((v(g)-v(b))-m1:vth<0)?((v(g)-v(b))<-1.1||(v(g)-v(b))>1.1):((v(g)-v(b))<-2.2||(v(g)-v(b))>2.2)				0	warning	yellow
4	M0.soa_122	FAILED	nlvtgp		expr=v(b)-v(d)	-2.2			2e-09	warning	yellow
5	M0.soa_125	PASSED	nlvtgp		expr=((v(g)-v(d))-m1:vth>0&&(v(d)-v(s))<-0.2)?((v(b)-v(s))<-1.2||(v(b)-v(s))>0.2):0				0	warning	yellow
6	M0.soa_119	FAILED	nlvtgp		expr=v(g)-v(s)	-1.1	1.1		2e-09	warning	yellow
7	M0.soa_120	FAILED	nlvtgp		expr=v(d)-v(s)	-1.1	1.1		2e-09	warning	yellow
8	M0.soa_118	FAILED	nlvtgp		expr=v(g)-v(d)	-1.1	1.1		2e-09	warning	yellow
9	M0.soa_127	PASSED	nlvtgp		expr=v(b)-v(s)		0.33		2e-09	warning	yellow
10	M0.soa_126	PASSED	nlvtgp		expr=v(b)-v(d)		0.33		2e-09	warning	yellow

Highlight Clear Details... Print...

Close Help

Post Processing Result Of Spectre(2)

cadence

Options are available : Highlight/Print/details/save

Highlight: To Highlight the Instance

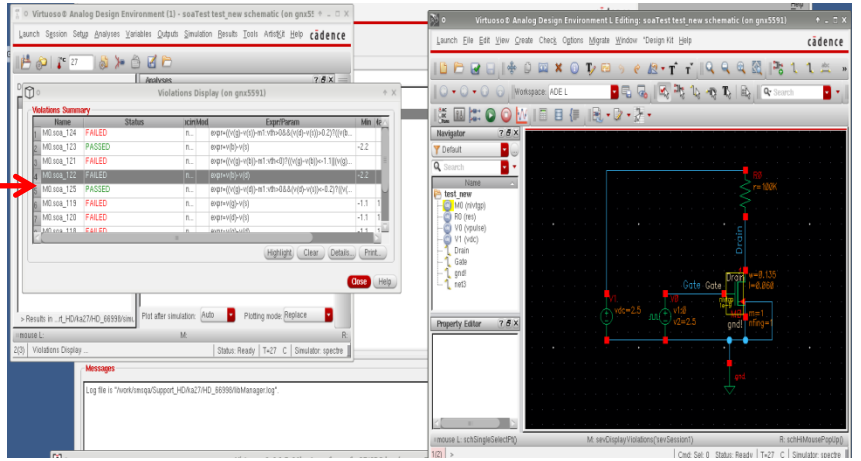
1. Select violation message
2. Click to **highlight**

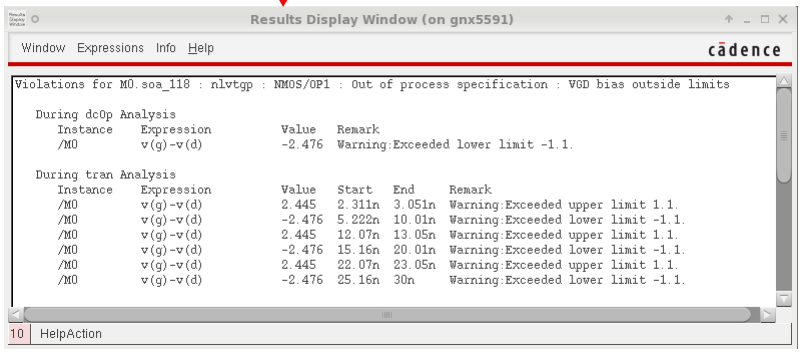
Details: To have more **details** about this violation

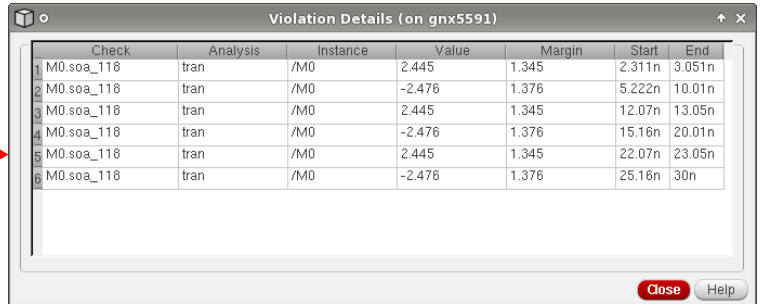
1. Select violation message
2. Click to **details**

Print: To have **all details** about this violation

1. Select violation message
2. Click to **print**







life.augmented

Allow SOA check (Hspice)

SYNOPSYS®

All defined Biaschk statement in your models are enable by default

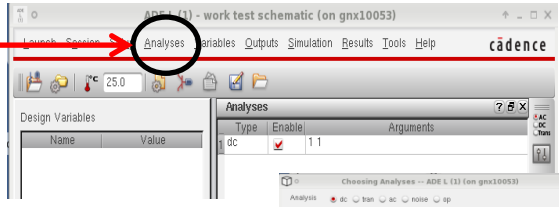
Results Display: Log file contains all biaschk results

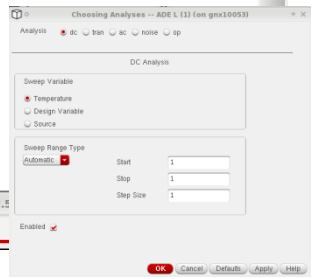
Launch a simulation with biaschk

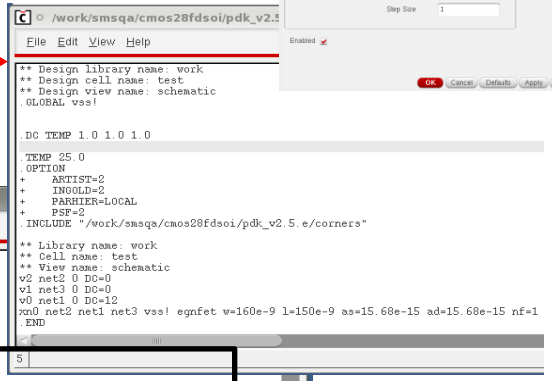
Step1: ADE>Analysis Choose your analysis

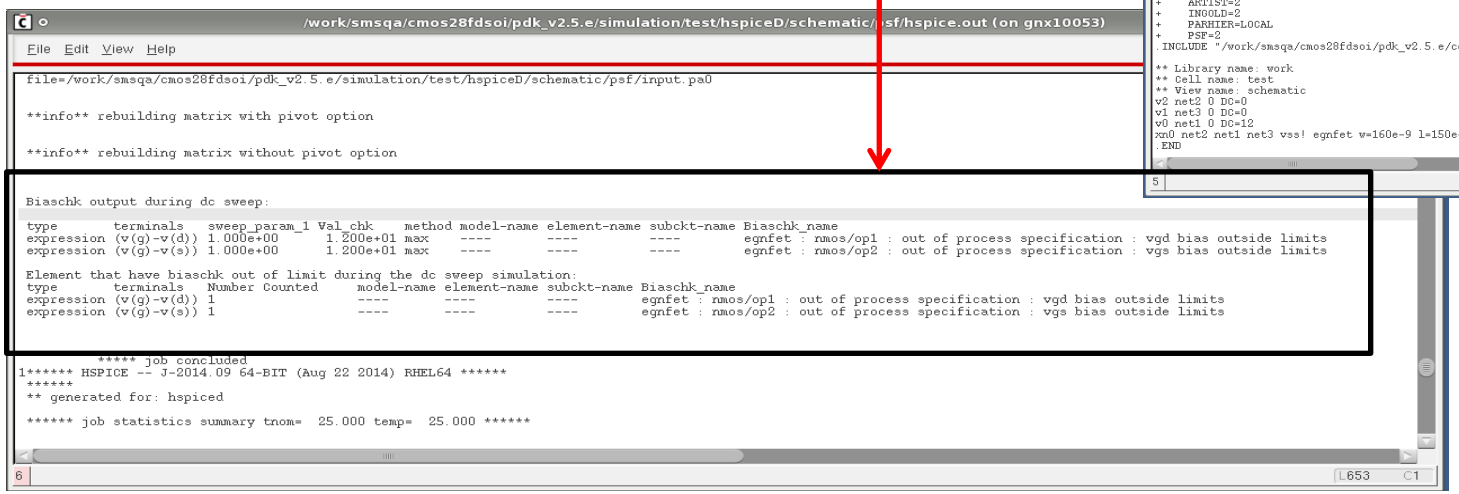
Step2: Create netlist and Run simulation

Step3: All violations appears into log file







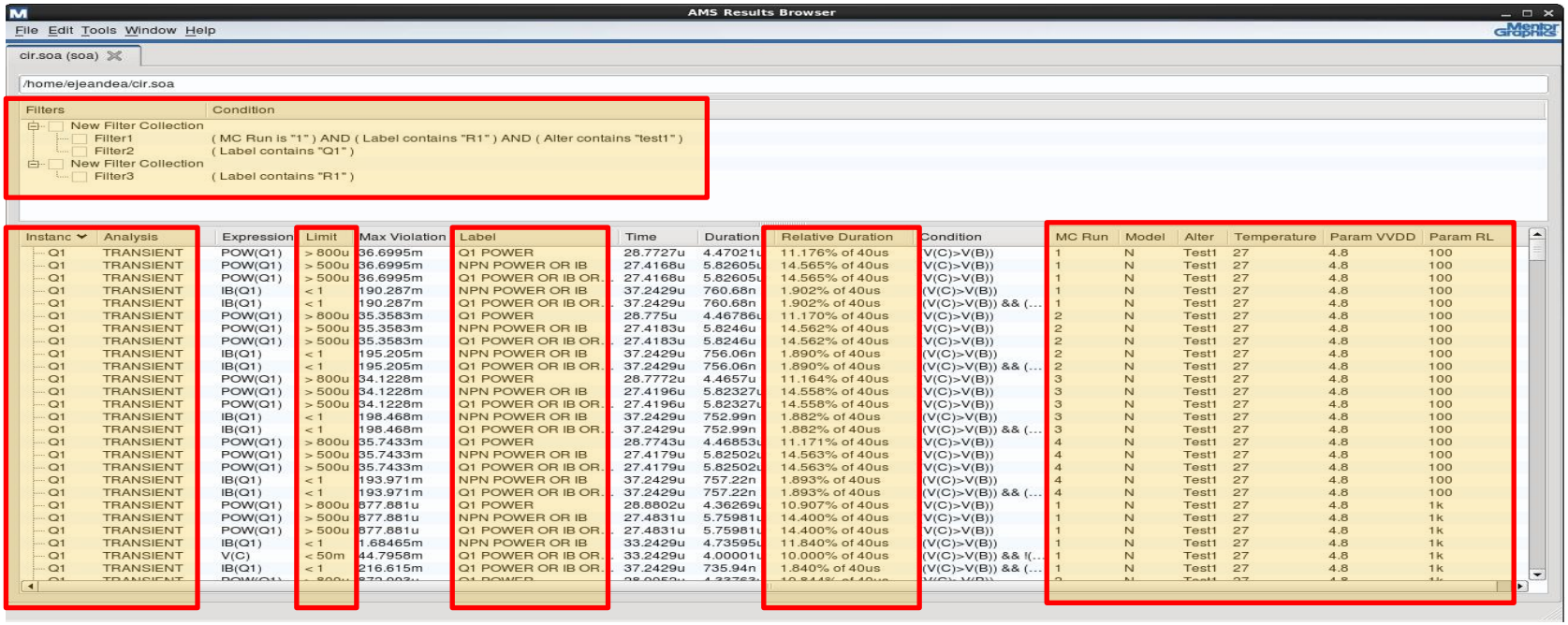


PART III

AMS Result Browser « AMSRB »

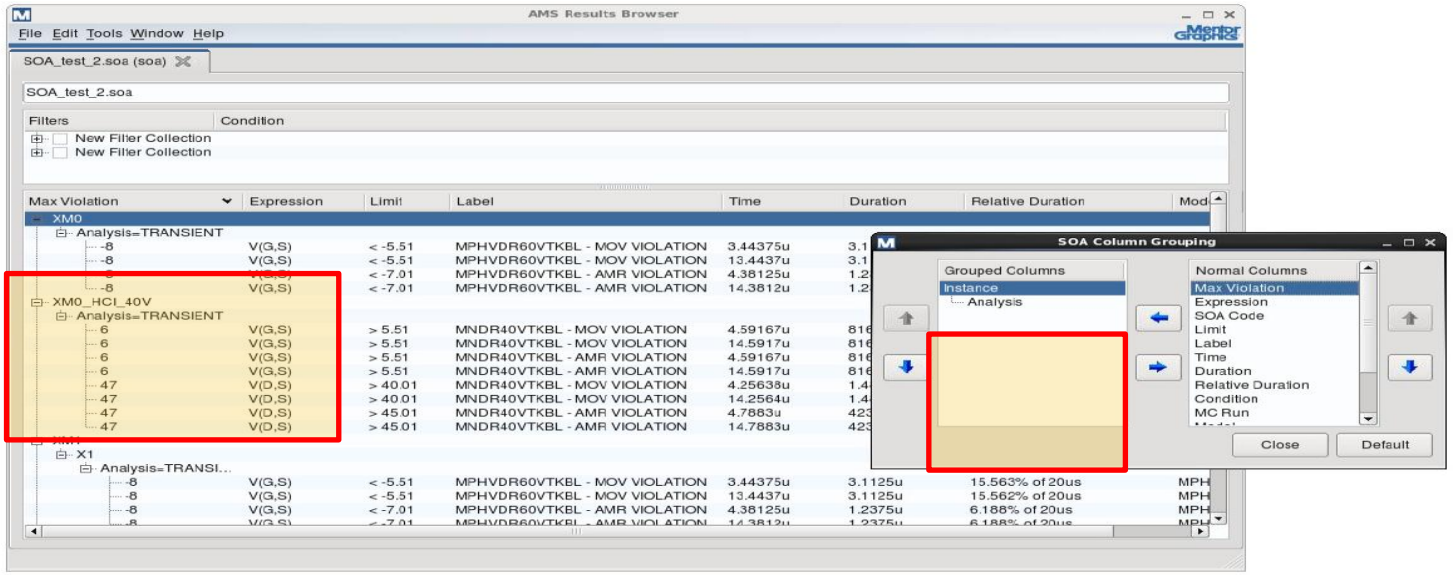
AMS Results Browser : NEW in 13.2

- SOA Mode : New SOA window “look and feel”
  - New columns : **Instance | Model | Analysis | Alter | Limit | Label | Relative duration | MC Run | Alter | Temperature | Parameters**
  - New Filters selection with user-defined expressions



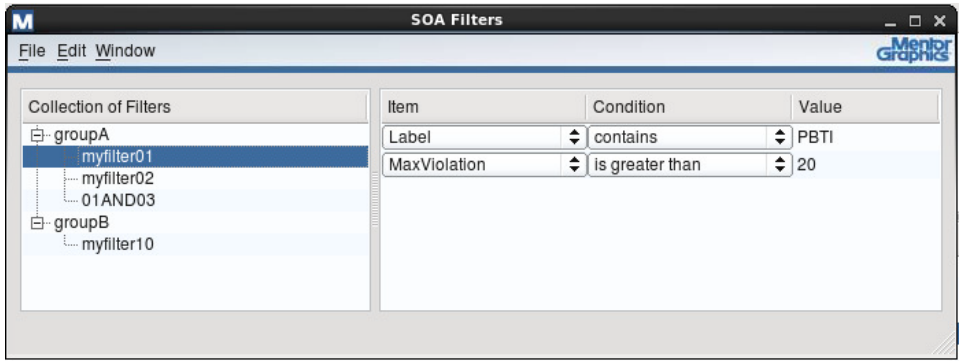
AMS Results Browser : NEW in 13.2

- SOA Mode : New SOA grouping capability
  - User can now configure the display of SOA violation results by grouping several SOA data into the same column.
  - Allow users to classify SOA violations based on their requirements (Hierarchical representation)



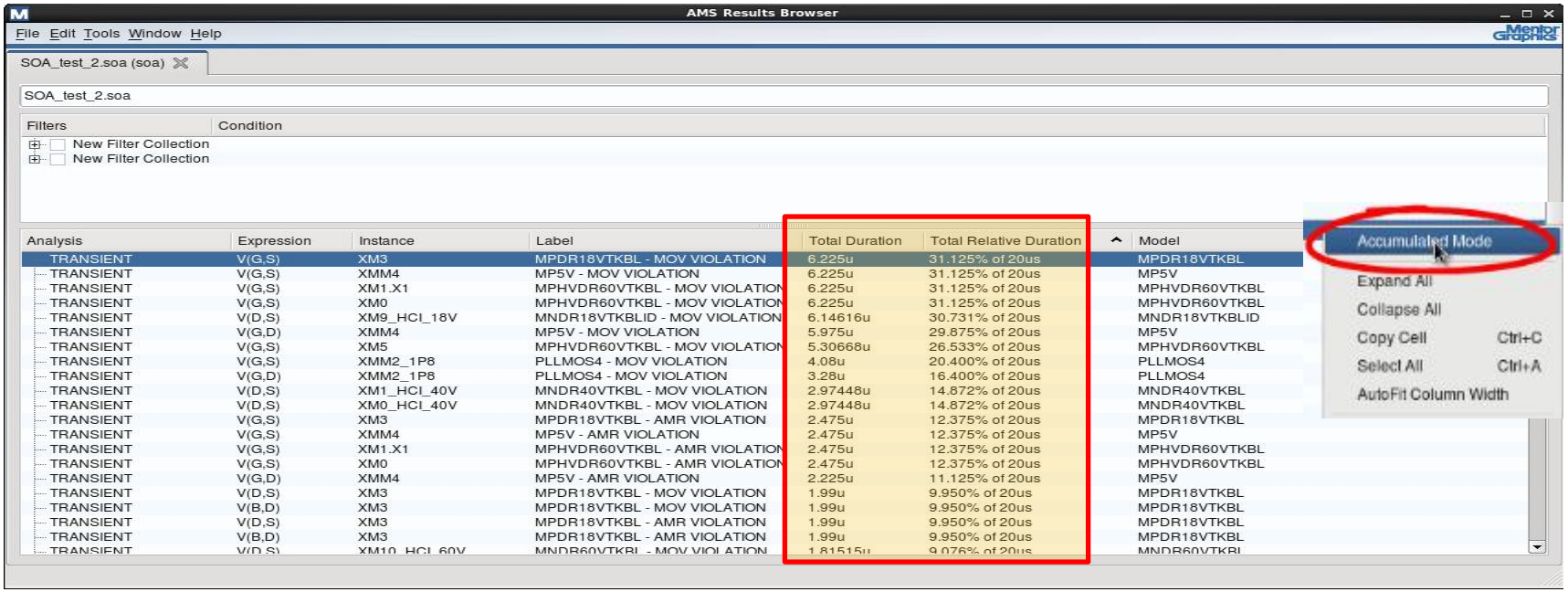
AMS Results Browser : NEW in 13.2

- SOA Mode : New SOA filtering
  - New SOA Filters setup window
  - Filters can now be grouped as a “collection” to allow better classification (PVT conditions / Projects / etc..)
  - New capability to use combination between filters (AND/OR)



AMS Results Browser : NEW in 13.2

- SOA Mode : New “Accumulated mode”
  - SOAs can now be displayed in 2 ways :
    - “Accumulated mode” : all violations on the same device are added to show up the “Total Relative Duration” Vs Total Transient simulation time.
    - “Non-Accumulated View” : Default display just like in previous releases.



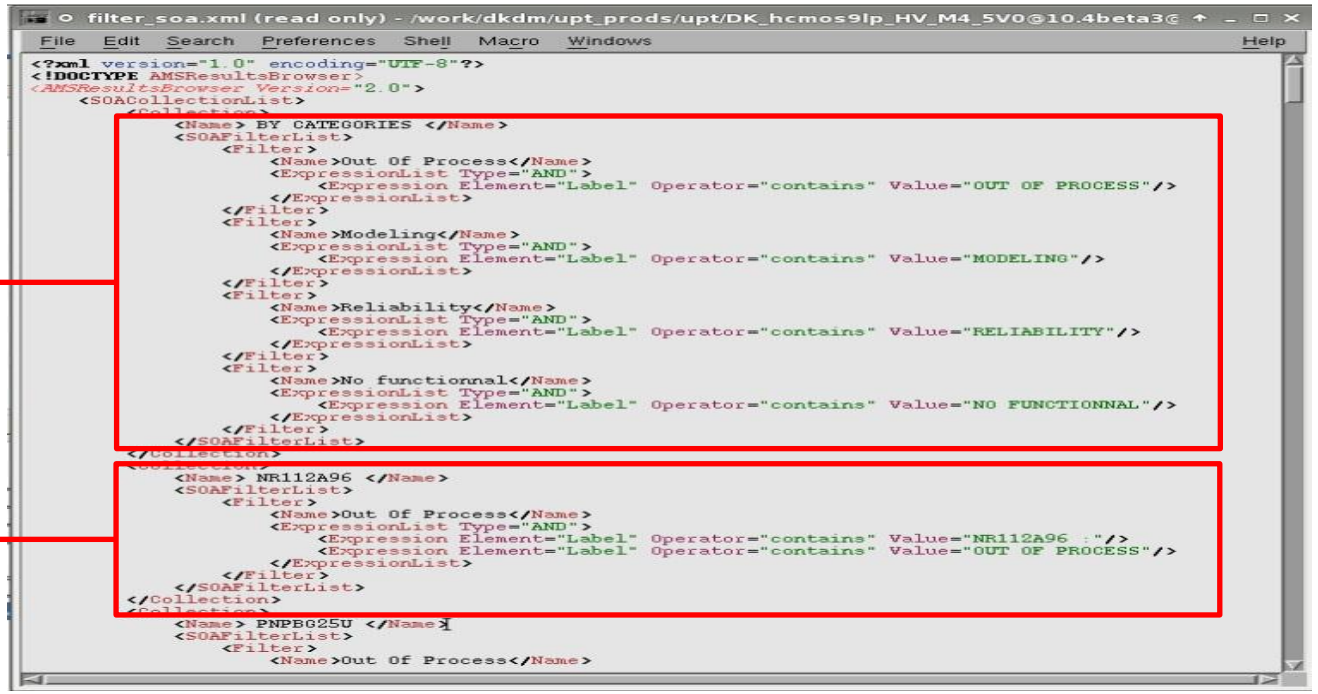
Predefined Filter

In terms to help designers we created a predefined filters :

- Format Xml
- Name `soafilters.xml`
- Since ams13.2a is loaded automaticly with AMSRB
- Possibility to Filter violations message by categories or families

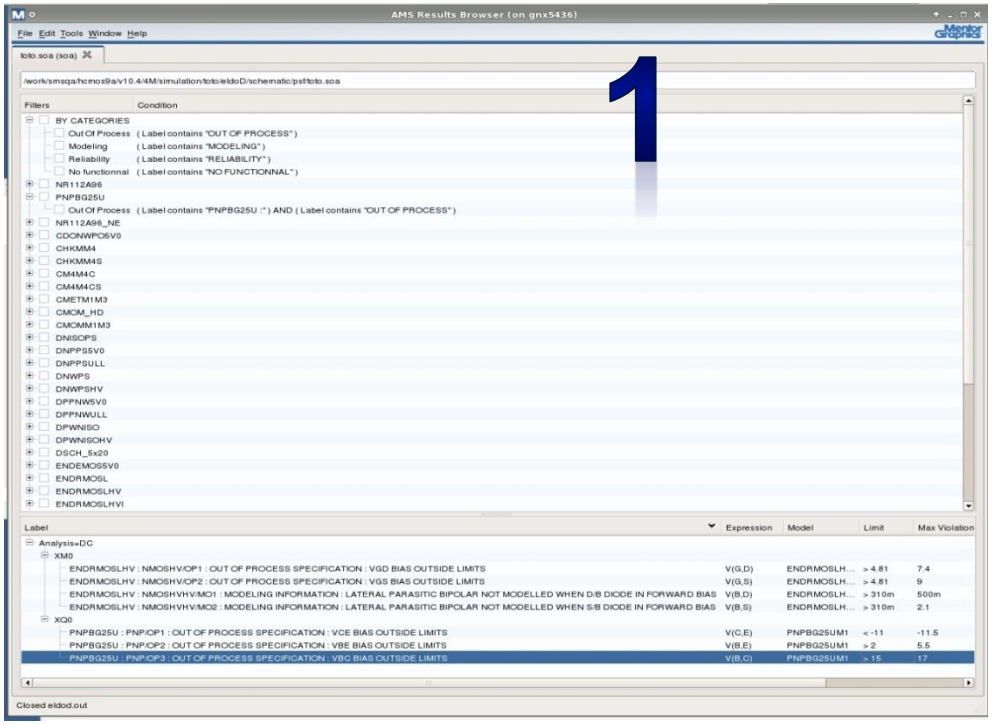
Categories

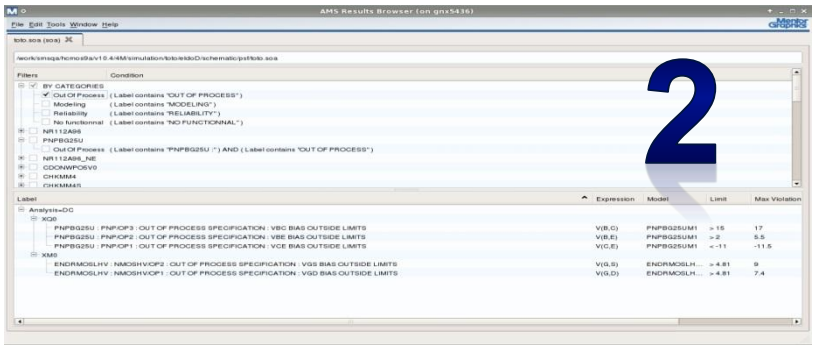
families

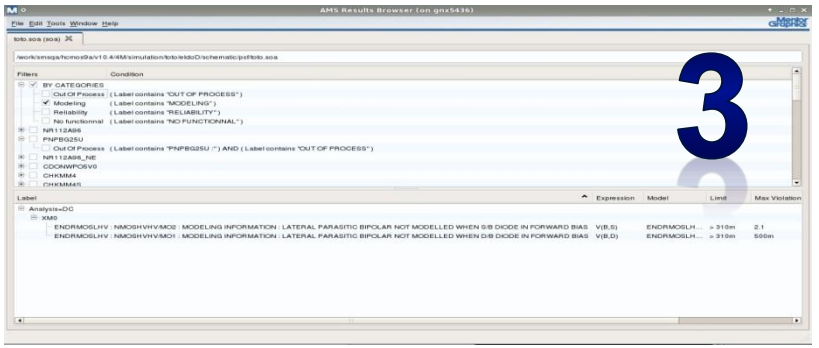


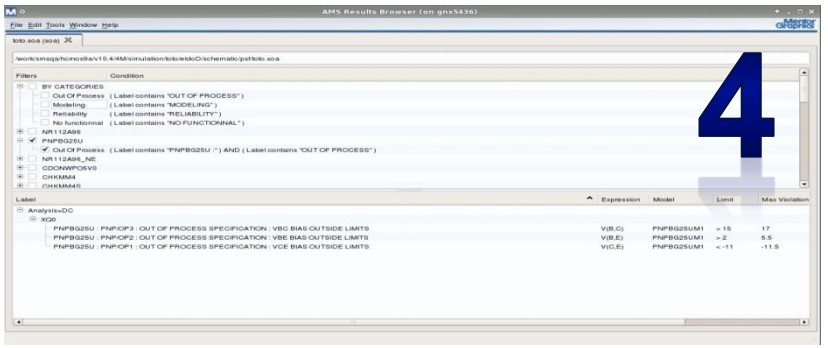
life.augmented

Pre-defined Filter example









Pic1: AMSRb inteface with pre-defined filter
Pic2: Select Out of process Category
Pic3: Select Modeling category
Pic4: Select the PNPBG25U family

Conclusion

- To have more information on the rules implemented, look at the documentation (soa_documentation.pdf) usually present in the DK documentation directory