cadence®

Legato Reliability Solution

Product Version ICADVM20.1
October 2020

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

Contents

Preface . . . 5
Scope . . . 5
Licensing Requirements . . . 6
Related Documentation . . . 7
What's New and KPNS . . . 7
Installation, Environment, and Infrastructure . . . 7
Technology Information . . . 8
Virtuoso Tools . . . 8
Additional Learning Resources . . . 8
Video Library . . . 8
Virtuoso Videos Book . . . 9
Rapid Adoption Kits . . . 9
Workshops . . . 9
Help and Support Facilities . . . 9
Customer Support . . . 10
Feedback about Documentation . . . 10
Typographic and Syntax Conventions . . . 11
Introduction to Legato™ Reliability Solution . . . 13
Related Learning Resources . . . 14
Analog Defect Simulation . . . 15
Simulating Analog Defects in ADE Assembler . . . 16
Injecting Faults in Design . . . 17
Simulating Tests with Defects . . . 19
Analyzing Faults and Test Coverage . . . 20
Functional Safety Checks for Design . . . 23
Simulating Analog Defects using Spectre at Command-Line . . . 26
Electrothermal Analysis . . . 33
Electrothermal Analysis Methods . . . 34
Steady-State Thermal Analysis . . . 34
Dynamic Thermal Analysis . . . 35

Performing Electrothermal Analysis in Spectre Command-Line Mode 36
Viewing Electrothermal Analysis Output in Spectre Results 37
Performing Electrothermal Analysis in Virtuoso ADE Assembler 39
Pre-requisites for Electrothermal Analysis 39
Viewing Results of Electrothermal Analysis 44
Advanced Aging Analysis 47
Self-Heating and Aging Flows 47
Aging Monte Carlo Analysis 48
Single Stress (1+N) flow 48
Multi-Stress (N+N) flow 50

Preface

The Cadence ® Legato™ Reliability Solution is a complete design-for-reliability solution for analog designers. This user guide describes the details of this solution.

This preface describes the following:

- Scope
- Licensing Requirements
- Related Documentation
- Additional Learning Resources
- Customer Support
- Feedback about Documentation
- Typographic and Syntax Conventions

Scope

Unless otherwise noted, the functionality described in this guide can be used in both mature node (for example, IC6.1.8) and advanced node (for example, ICADVM18.1) releases.

Label	Meaning
(ICADVM18.1 Only)	Exclusive set of advanced node features that require the Virtuoso Advanced Node Option for Layout (95511) license.
(IC6.1.8 Only)	Features supported only in mature node releases.

Licensing Requirements

The Legato Reliability Solution is available in IC6.1.8, ICADVM18.1, and SPECTRE 18.1 releases. In Spectre 18.1, this solution is available only with Spectre® Accelerated Parallel Simulator (Spectre APS) high performance multi-core SPICE engine.

The licenses required for the Legato Reliability Solution are listed below.

- 95260, Virtuoso ADE Assembler license
- A new Legato reliability solution license, which is a pack of 6 tokens, is required to run the following reliability features:

Features	No of tokens	Functionality
Analog Fault Simulation	3	Spectre transient fault analysis and Spectre direct fault analysis
Fault Generation	0	info what=bridges|opens Does not check out the license; however, checks for the availability of the license.
spectre_ddmrpt	0	spectre_ddmrpt Does not check out the license; however, checks for the availability of the license.
spectre_fsrpt	0	Does not check out the license; however, checks for the availability of the license.
Electrothermal Simulation	2	Spectre electrothermal simulation **Note:** Electrothermal simulation requires a Sigtherm_Signoff license during Spectre thermal extraction, and 2 Legato tokens during the remaining electrothermal simulation. The Sigtherm_Signoff license is released after thermal extraction is completed, and then the 2 legato tokens are checked out.

RelXpert	4	RelXpert simulation
Spectre self-heating	1	Spectre self-heating analysis
Spectre self-heating + aging analysis	1	Spectre self-heating and aging analysis
Aging analysis + Monte Carlo	1	Aging Monte Carlo analysis
RF reliability analysis	1	Spectre RF reliability analysis
Spectre reliability with XPS S mode	1	Spectre reliability analysis with XPS SPICE mode

For information on licensing for Virtuoso products, see *Virtuoso Software Licensing and Configuration Guide*.

Related Documentation

The following documents provide more information about the topics discussed in this guide.

What's New and KPNS

- *Virtuoso ADE Assembler What's New*
- *Virtuoso ADE Assembler Known Problems and Solutions*
- *Spectre Circuit Simulator and Accelerated Parallel Simulator What's New*

Installation, Environment, and Infrastructure

- *Cadence Installation Guide*
- *Virtuoso Design Environment User Guide*
- *Virtuoso Design Environment SKILL Reference*
- *Virtuoso ADE Explorer and ADE Assembler SKILL Reference*
- *Cadence Application Infrastructure User Guide*

Technology Information

- *Virtuoso Technology Data User Guide*
- *Virtuoso Technology Data ASCII Files Reference*
- *Virtuoso Technology Data SKILL Reference*

Virtuoso Tools

- *Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*
- *Virtuoso ADE Assembler User Guide*
- *Virtuoso ADE Explorer User Guide*
- *Virtuoso Schematic Editor User Guide*
- *Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide*
- *Spectre Circuit Simulator Reference*
- *Spectre Circuit Simulator and Accelerated Parallel Simulator RF Analysis in ADE Explorer User Guide*
- *Virtuoso UltraSim Simulator User Guide*
- *Virtuoso AMS Designer Simulator User Guide*
- *Virtuoso Parasitic Estimation and Analysis User Guide*
- *Virtuoso Visualization and Analysis Tool User Guide*
- *Component Description Format User Guide*
- *Analog Expression Language Reference*

Additional Learning Resources

Video Library

The following videos related to the Legato Reliability solution are available on the Cadence Online Support website:

- Performing Fault Simulation in Virtuoso ADE Assembler using Legato™ Reliability Solution

- Performing Fault Analysis in Spectre using Legato™ Reliability Solution

In addition, you can view the Video Library on the Cadence Online Support website. This library provides a comprehensive list of videos on various Cadence products.

To view a list of videos related to a specific product, you can use the *Filter Results* feature available in the pane on the left. For example, click the *Virtuoso Layout Suite* product link to view a list of videos available for the product.

You can also save your product preferences in the Product Selection form, which opens when you click the *Edit* icon located next to *My Products*.

Virtuoso Videos Book

You can access certain videos directly from Cadence Help. To learn more about this feature and to access the list of available videos, see Virtuoso Videos.

Rapid Adoption Kits

Cadence provides a number of Rapid Adoption Kits that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

Workshops

For Spectre, you can run the following workshops that are available in the installation directory:

- `<Spectre-install-path>/tools/spectre/examples/SpectreFaultAnalysis`
- `<Spectre-install-path>/tools/spectre/examples/SpectreThermalAnalysis`
- `<Spectre-install-path>/tools/spectre/examples/reliability_workshop`

Help and Support Facilities

Virtuoso offers several built-in features to let you access help and support directly from the software.

- The Virtuoso *Help* menu provides consistent help system access across Virtuoso tools and applications. The standard Virtuoso *Help* menu lets you access the most useful help and support resources from the Cadence support and corporate websites directly from the CIW or any Virtuoso application.

- The Virtuoso Welcome Page is a self-help launch pad offering access to a host of useful knowledge resources, including quick links to content available within the Virtuoso installation as well as to other popular online content.

  The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the *Home* button on the Cadence Help toolbar (provided you have not set a custom home page).

For more information, see Getting Help in *Virtuoso Design Environment User Guide.*

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

  Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit https://www.cadence.com/support.

- Log on to Cadence Online Support

  Customers with a maintenance contract with Cadence can obtain the latest information about various tools at https://support.cadence.com.

Feedback about Documentation

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for
- Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support Product Manuals page, select the required product and submit your feedback by using the *Provide Feedback* box.

Typographic and Syntax Conventions

The following typographic and syntax conventions are used in this manual.

text	Indicates names of manuals, menu commands, buttons, and fields.
`text`	Indicates text that you must type exactly as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
`z_argument`	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, *`z_`*) indicates the data type the argument can accept and must not be typed.
|	Separates a choice of options.
`{ }`	Encloses a list of choices, separated by vertical bars, from which you **must** choose one.
`[ ]`	Encloses an optional argument or a list of choices separated by vertical bars, from which you **may** choose one.
`[ ?argName` *`t_arg`* `]`	Denotes a *key argument*. The question mark and argument name must be typed as they appear in the syntax and must be followed by the required value for that argument.
`...`	Indicates that you can repeat the previous argument. Used with brackets to indicate that you can specify zero or more arguments. Used without brackets to indicate that you must specify at least one argument.
`,...`	Indicates that multiple arguments must be separated by commas.
`=>`	Indicates the values returned by a Cadence SKILL language function.
`/`	Separates the values that can be returned by a Cadence SKILL language function.

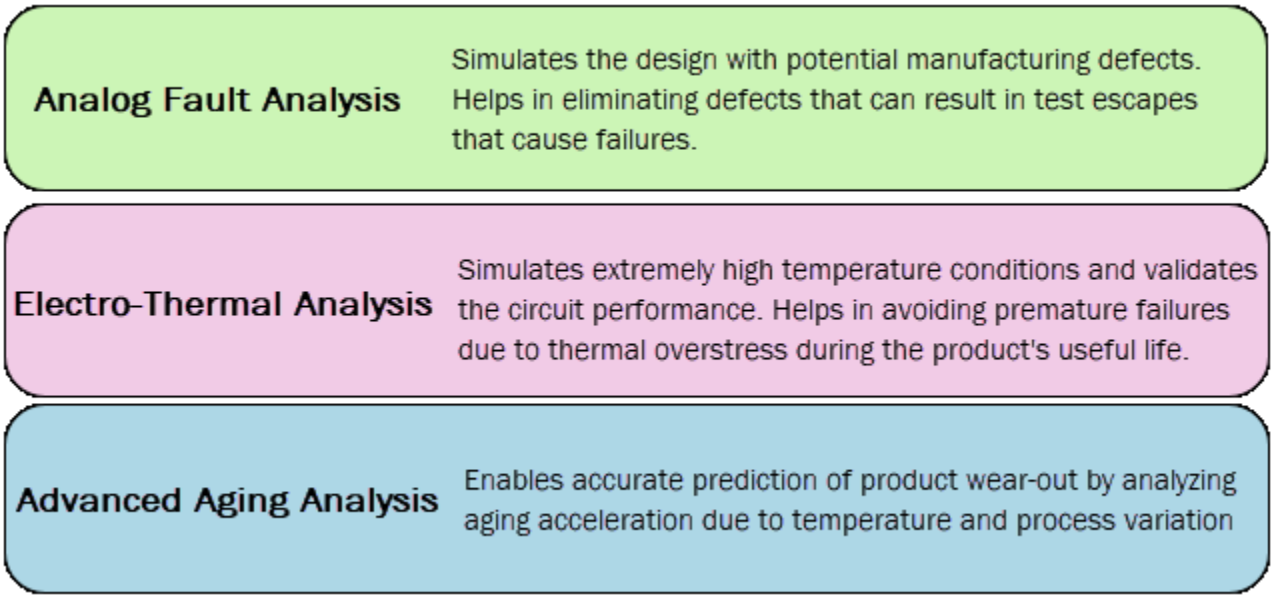
If a command-line or SKILL expression is too long to fit within the paragraph margins of this document, the remainder of the expression is moved to the next line and indented. In code excerpts, a backslash (\ ) indicates that the current line continues on to the next line.

1

Introduction to Legato™ Reliability Solution

Reliability throughout the product's life cycle is the top-most challenge faced by the analog designers involved in designing the advanced and complex devices used in mission-critical systems for aerospace, automative and medical equipments. Not only it is important to produce reliable devices by eliminating all possible manufacturing defects, but it is also essential to ensure high performance in extreme operating conditions throughout their useful life. This requires a deep analysis of the effects of high temperature and other stress conditions that can lead to an early end of life.

The Legato™ Reliability Solution is the industry's first complete analog IC design-for-reliability solution that empowers the analog designers with a set of three new analyses that they can include in design simulations to address these challenges.



The three analyses in this solution are based on the Virtuoso® IC custom design platform and Spectre® Accelerated Parallel Simulator.

The following sections explain how to use these analyses to improve the reliability of your analog designs:

- Analog Defect Simulation
- Electrothermal Analysis
- Advanced Aging Analysis

Related Learning Resources

- Legato Reliability Solution product page

1

Analog Defect Simulation

An important concern of IC manufacturers is the processing defects that get introduced into the devices due to various issues in the manufacturing process. Some examples of such issues are open circuits caused by weak wire bonds or misaligned layers that lead to improper connections. Some of these issues can be observed in newly fabricated devices, but some are difficult to trace. Therefore, manufacturers look at performing a thorough inspection to reduce the potential causes of defects.

The Fault Simulation component of the Legato Reliability solution provides a capability to run defect-oriented tests on analog IC designs. These tests allow you to evaluate the ability to eliminate a die with manufacturing defects and resulting test escapes that cause field failures. It can also be used to optimize wafer test, reducing the number of tests required to achieve the target defect coverage by eliminating over-testing and potentially reducing the number of tests.

As a part of this solution, the following features have been provided in Cadence products:

- Fault Analysis in Spectre Accelerated Parallel Simulator

  This analysis in Spectre provides a transistor-level simulation capability that can be enabled in an analog test methodology to improve test coverage by identifying critical test patterns. It is very fast in handling large numbers of faults, and also very accurate when full fault analysis precision is required.

- Fault Simulation run mode in Virtuoso ADE Assembler

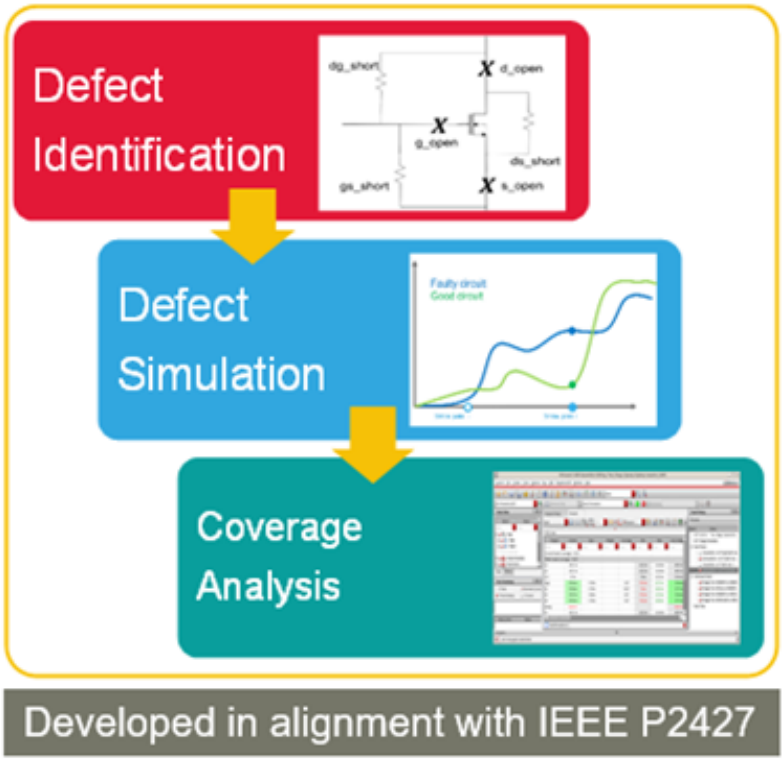
  ADE Assembler provides an interactive simulation environment to create test setups, inject faults in the test designs, run the Fault Analysis using Spectre, and present the results for post-simulation analysis. Since analog designers are already familiar with ADE Assembler, running fault analysis is easy and quick.

  Note: To run fault simulation in ADE Assembler, you require Spectre 18.1 ISR1 or a newer version of Spectre simulator.

Simulating Analog Defects in ADE Assembler

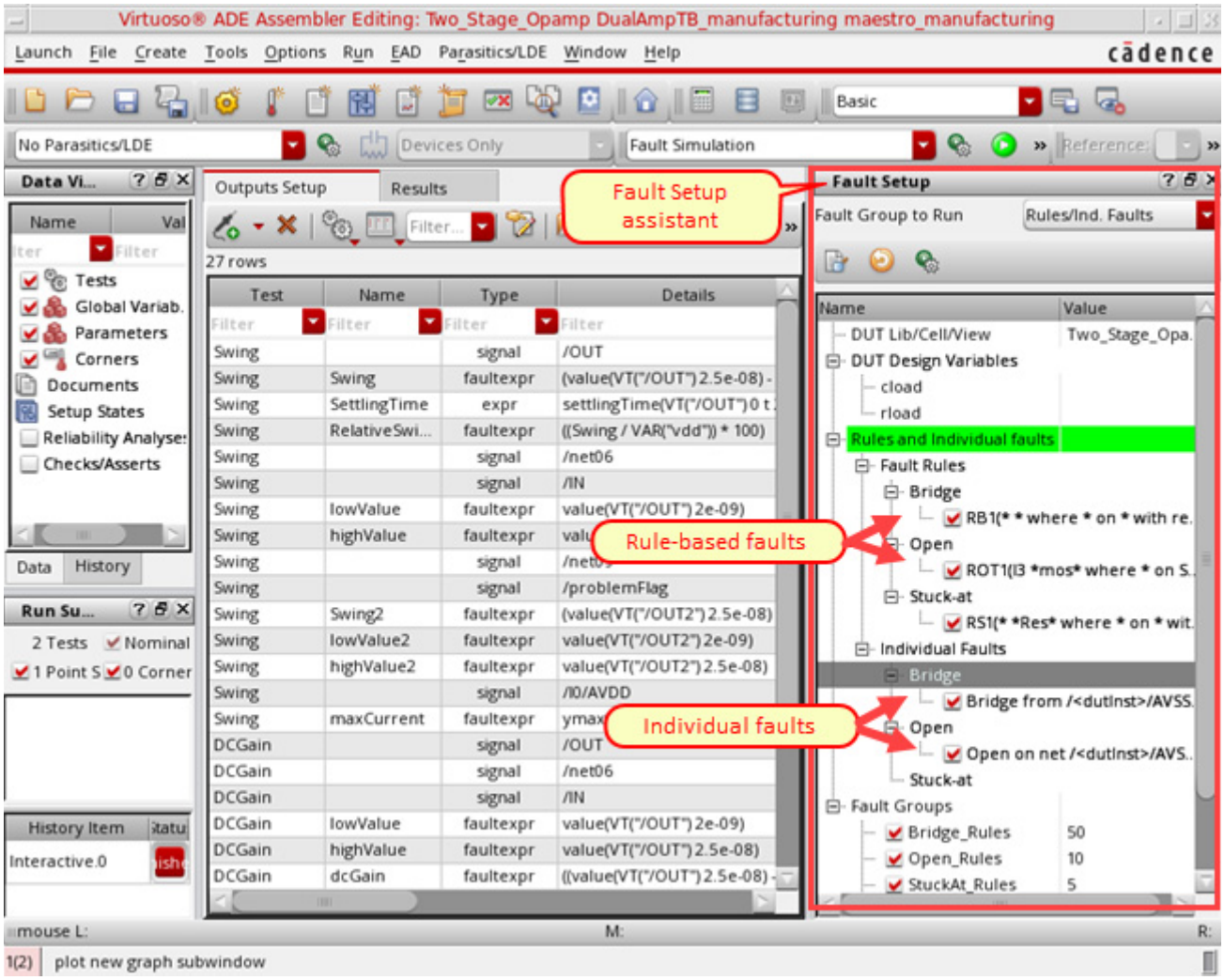
A typical fault analysis that simulates analog defects consists of the following three steps:

- Injecting Faults in Design
- Simulating Tests with Defects
- Analyzing Faults and Test Coverage



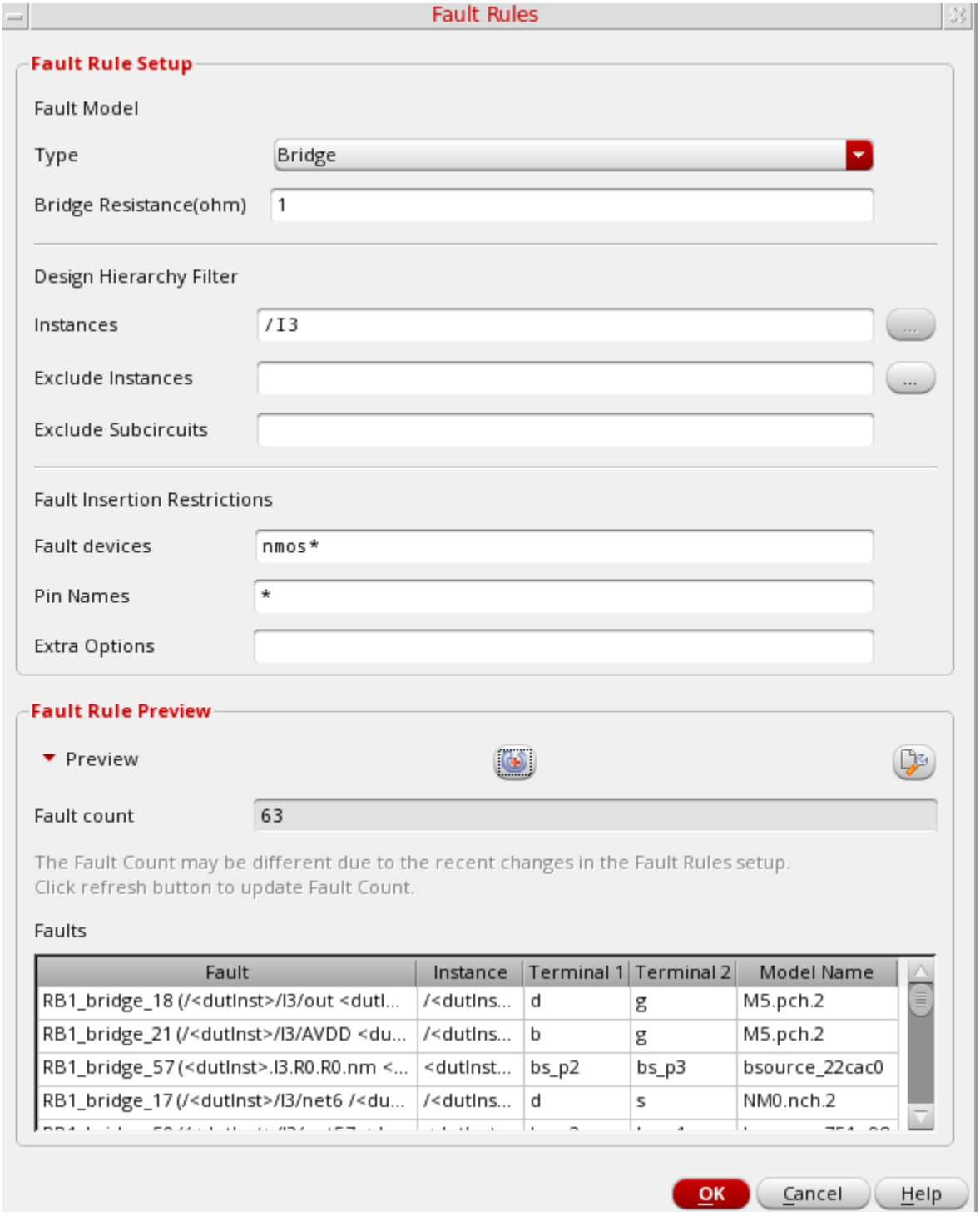
Injecting Faults in Design

Use the new *Fault Setup* assistant in ADE Assembler to specify the faults to be injected into designs. You can either specify individual faults to be injected in specific devices, or create rules for faults to be injected in multiple devices based on the specific criteria.



To insert individual faults, you can directly select objects on the schematic, and specify a resistance value for the fault.

To create faults using rules, use the Fault Rules form to specify the design hierarchy filter and fault insertion restrictions. Based on the settings in this form, a defect list is automatically created and displayed in the *Fault Rule Preview* section of this form.

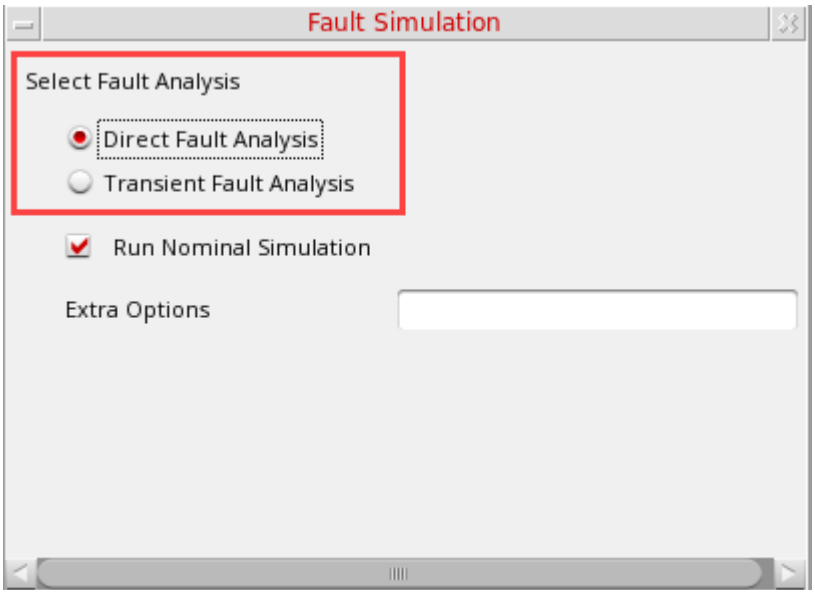


For more details about the Fault Setup assistant and the Fault Rules form, refer to Injecting Faults in the Design in *Virtuoso ADE Assembler User Guide*.

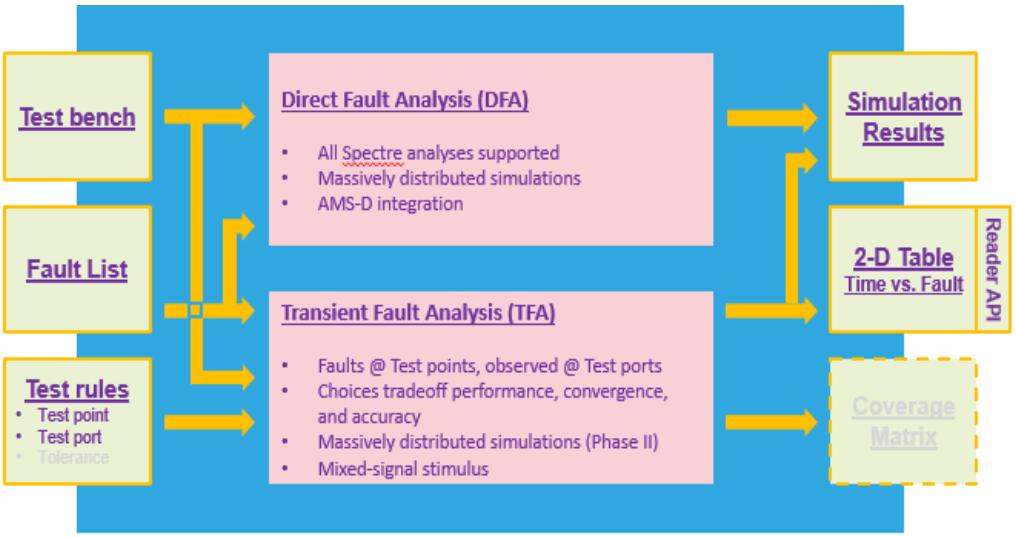
Simulating Tests with Defects

When the setup is complete, choose the `Fault Simulation` run mode, specify run options and run simulations using Spectre APS or AMS Designer.

You can perform fault analysis by using one of these two approaches: direct fault analysis and transient fault analysis.



The following figure explains the difference between the two approaches:



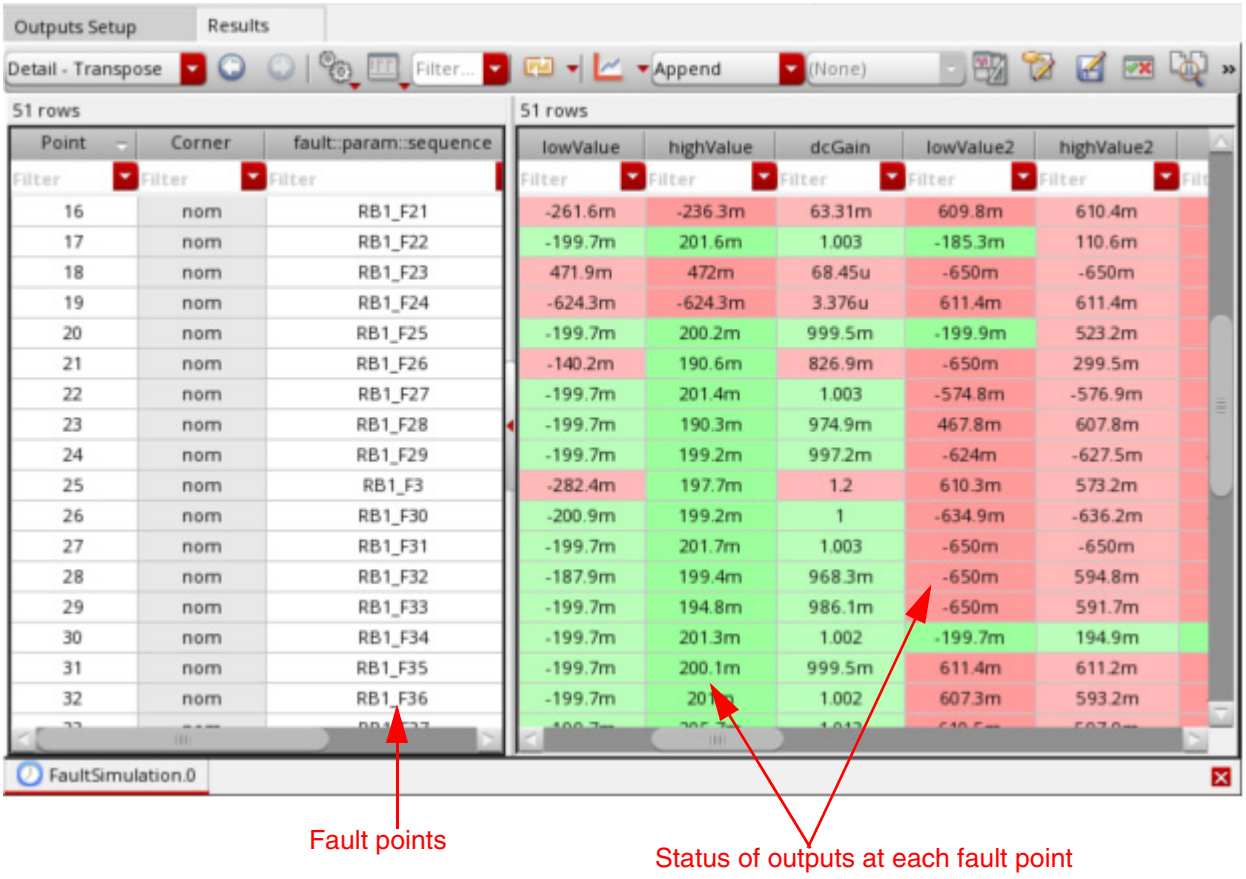
Depending on your requirements for speed and accuracy, you can choose an appropriate approach, and specify settings for the simulation run.

For details about the settings for these methods, refer to Setting up Run Options in *Virtuoso ADE Assembler User Guide*.

Analyzing Faults and Test Coverage

Analog fault analysis provides the capability to estimate fault coverage by generating a fault detection matrix. This matrix is obtained by comparing the faulty simulation results with

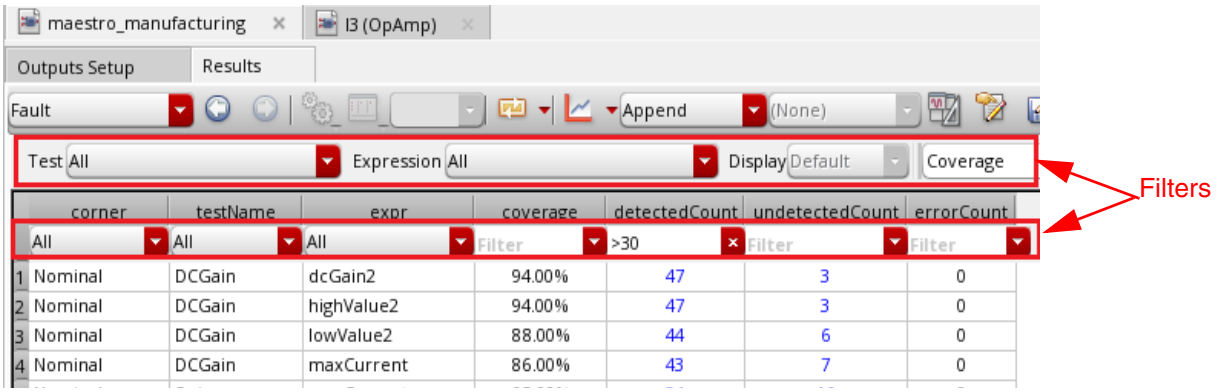
nominal (faultless) data, as shown in the snapshot of the `Detail-Transpose` results view given below.



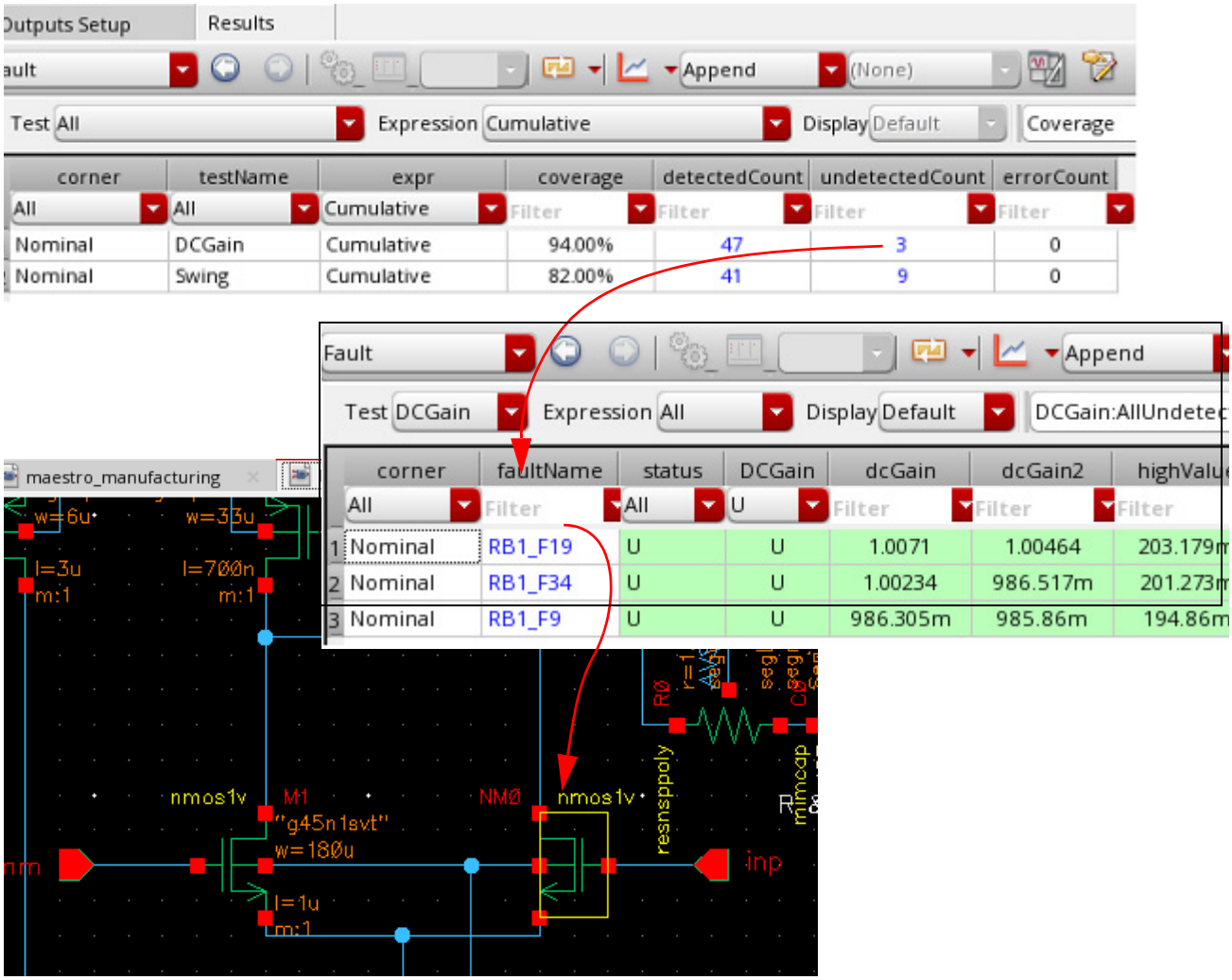
At each fault point, the simulator evaluates the outputs and calculates the count of detected and undetected faults.

You can also view the summary of these results in the new `Fault` results view. By default, this view shows the count of detected or undetected faults found while evaluating the expressions for each test.

You can use various filters or hyperlinks to focus on specific test, expression, or output, or to view faults in the design schematic.



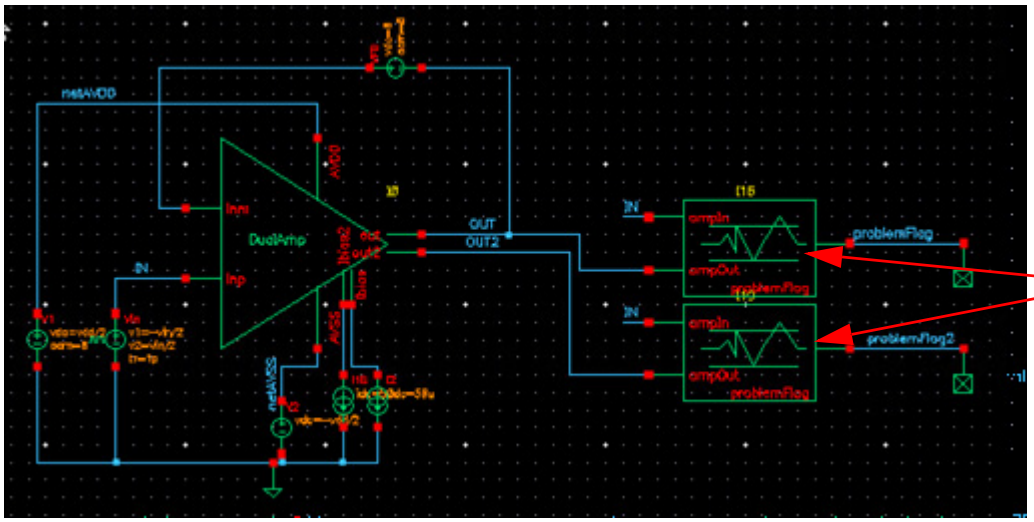
You can check the fault coverage for each test, and further optimize the tests to improve the fault coverage based on the fault detection matrix.



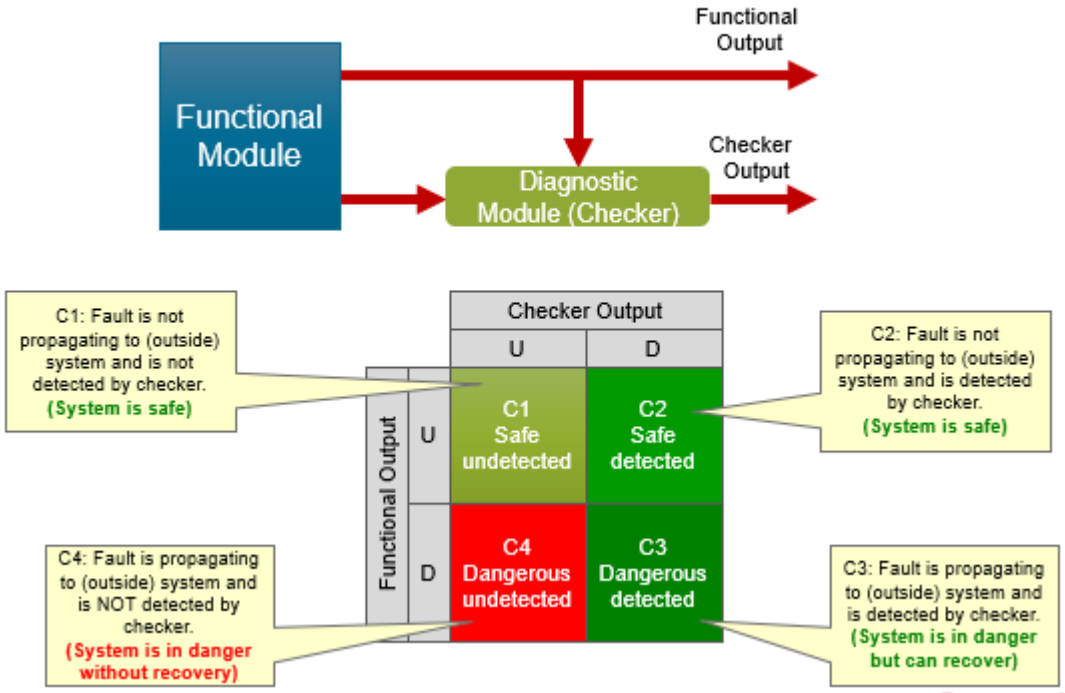
Functional Safety Checks for Design

In addition to the fault detection matrix, the results of fault simulation can be used as part of functional safety analysis. This analysis mainly focuses on the effect of a failure during the lifetime of a device.

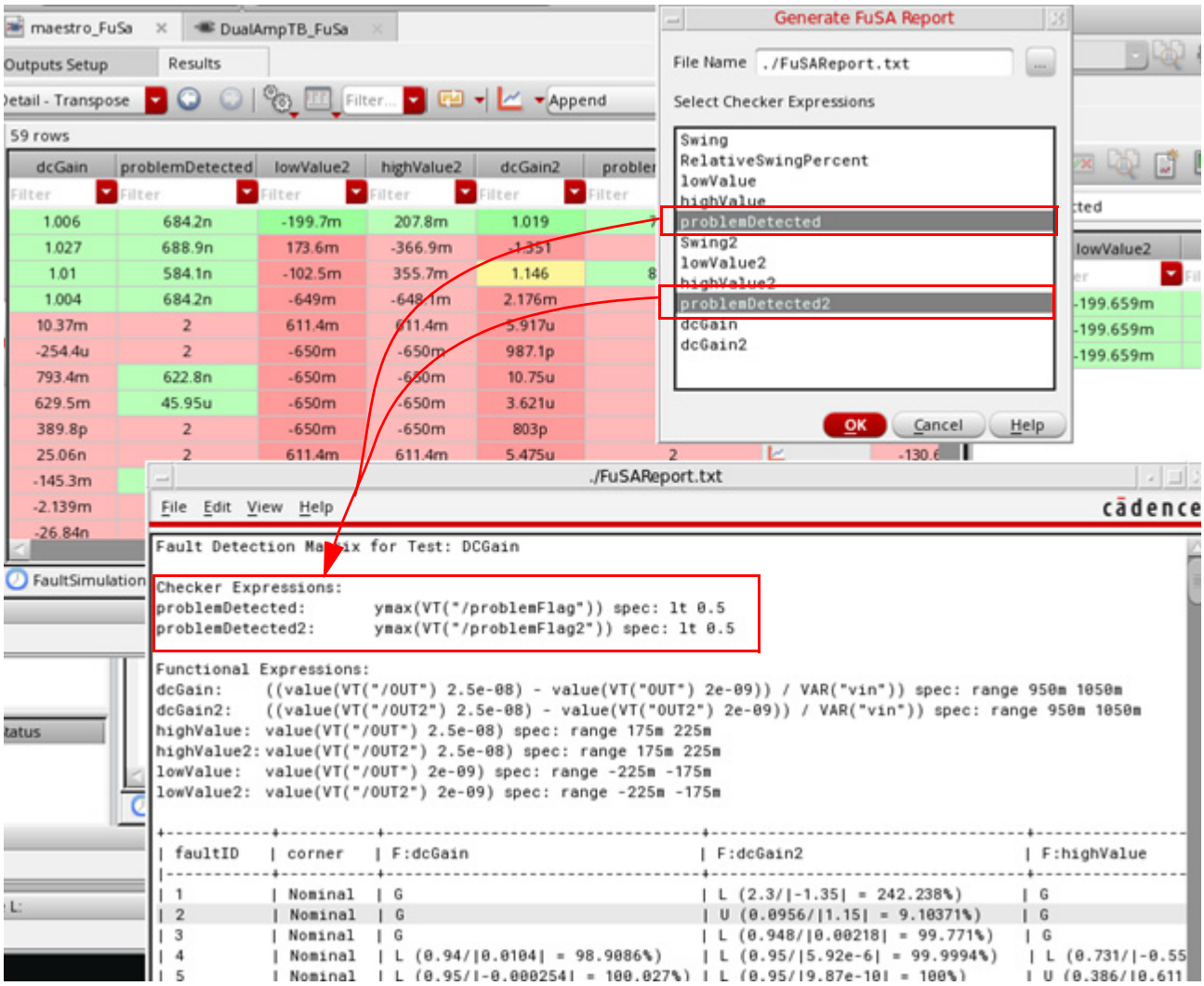
To generate this report, you need to have diagnostic or checker blocks in your design to validate the outputs of the functional block.



In functional safety applications, fault detection is tracked for both functional and diagnostic (checker) modules but reported separately. Using these results, you can generate a functional safety report that is aligned with the safety classification defined according to the ISO 26262 standard which classifies safe or dangerous, and detected or undetected defects.



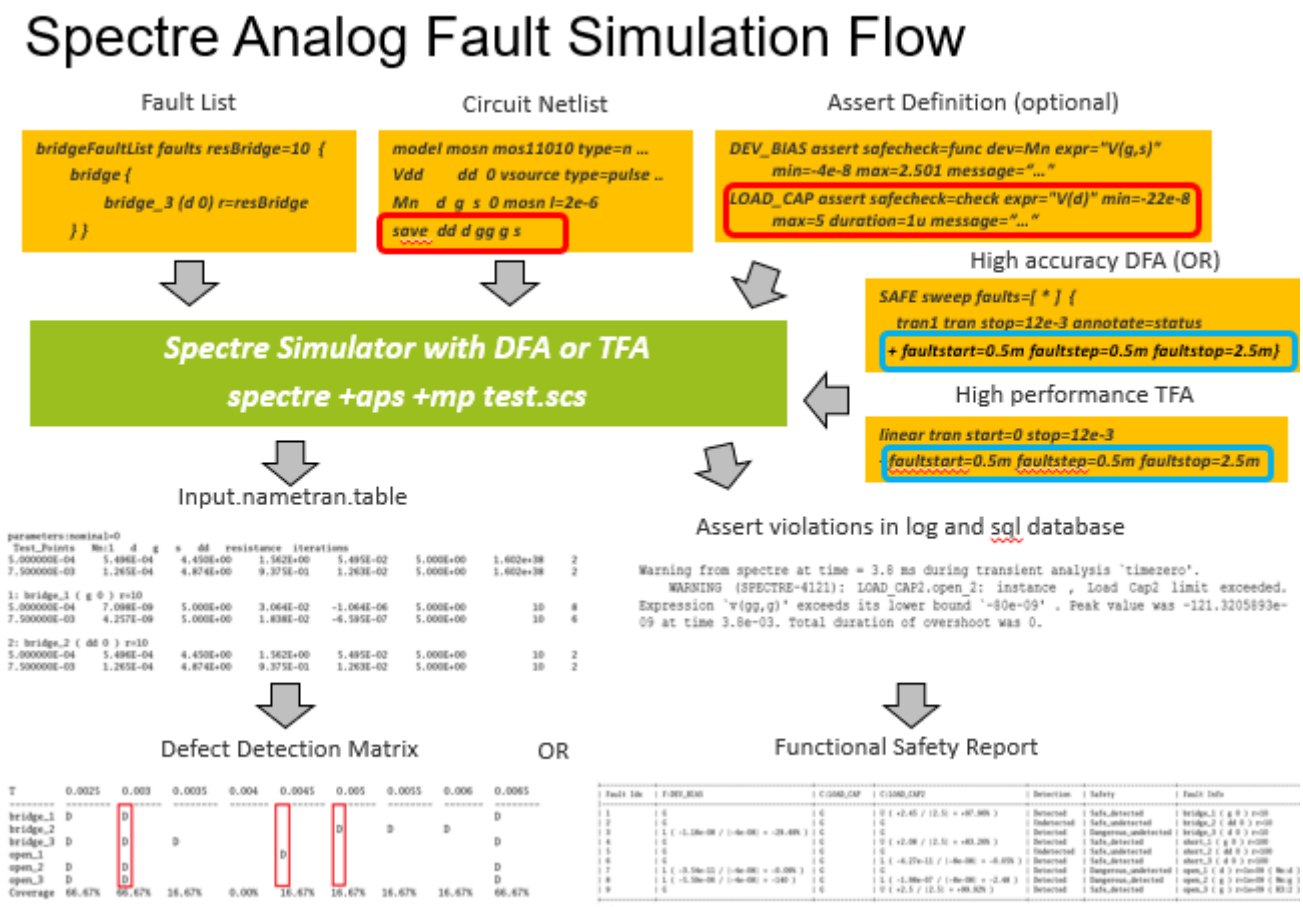
The following figure shows an example of the functional safety report generated in ADE Assembler.



For more details, refer to Viewing Results of Fault Simulation in *Virtuoso ADE Assembler User Guide*.

Simulating Analog Defects using Spectre at Command-Line

The flow of an analog defect simulation in Spectre is summarized in the following figure.



These steps are described below:

1. **Specify faults to be checked.** You can specify the faults in two ways:

   - By creating the `faults` block to specify a fault list that defines the type and location of faults
   - By automatically generating faults from the design.

   To create your own fault list, define the `faults` blocks. Spectre supports two types of faults - bridges (shorts) and opens, as shown in the example below.

```
name faults <parameter=value> {

bridge {

<name> ( <node1> <node2> ) r = <expression>

}

short {

<name> ( <node1> <node2> ) r = <expression>

}

open {

<name> ( <node> ) r = <expression> [ c = <expression> ]
   { instance1[:terminal] [instance2[:terminal2]]… }

}

}
```

Alternatively, you can run Spectre `info` analysis to generate the fault list based on the primitives, models, or subcircuits of a given design.

Syntax for Fault Creation

```
nameList info what=bridges|opens where=file file=file_faultlist_1.scs [
faultblock=nameBlock [ faultdev=[...] [ faultres=resValue [
faultterminals=[...] ] ] ] ]
```

Here:

- `faultblock`: specifies the names of the fault blocks to be generated. Default is the name of info analysis.
- `faultdev=[dev1 dev2...]`: specifies the fault devices by primitive name, subcircuit name, or model name. Default value is `all`.
- `faultterminals`: specifies the terminals of fault devices for open or short handling. The default value is `all terminals`.
- `faultres`: specifies the resistance value for bridges or opens. The default value is `10 Ohms` for bridges and `1 GOhms` for opens.
- `faultcap`: specifies the capacitance value for open faults. The default value is `0 F`.
- `faultduplicate`: If set to yes, duplicate faults are included in all fault lists when multiple info analyses are specified for fault generation. Possible values are `yes` and `no`.
- `faultdeviter`: If set to yes, a separate faults list is generated for each iterated instance, that is, for instances whose name contains `<number:[number]>`. Possible values are `yes`, and `no`.

2. **Use an assert violation as a fault detection technique by specifying the boundaries**. A fault is detected when an assert violation is triggered. To distinguish the checking of fault analysis from the regular assert check, you can add the `safecheck` parameter in the `assert` statement which is used in post-processing of assert violations to generate the functional safety report.

```
safecheck=[ none | func | check ]
```

3. **Specify the test points** using any of the following three methods:

   - Specify the sweeping points by using the `faultstart`, `faultstop`, and `faultstep` parameters
   - Specify the discrete time points using the `faulttimes` parameter
   - List the test points in a separate file to be included in simulation.

   Example 1

```
faultstart=0.5m faultstop=1.0m faultstep=0.5m
```

   Example 2

```
faultimes = [0.5m 0.75m 1.0m]
```

   Example 3

```
faultfile = "./file_test_points.txt"
```

   Test points in the `./file_test_points.txt` file can be defined as follows:

```
//fault time point
0.5m
0.75m
1.0m
```

4. **Run simulation**.

   Running Direct Fault Analysis

   Direct fault analysis uses sweep to iterate over the fault list and run the nested child analyses, such as tran, dc, ac, and so on.

   The following is the use model for Spectre direct fault analysis:

```
DirectName sweep faults=[ * | faultblock1...] nominal=[yes|no]
[faultsid=[...]] [faultsname=[...]][faultsinst=[...]] {
       TranName tran stop=10u step=1p
       DCName dc
       ACName ac start=1e3 stop=1e9
}
```

   Here:

   - `faults=[*]` specifies the fault analysis sweeps for all faults.

- `faults=[faultsblock..]` specifies the fault analysis sweeps for the defined blocks.
- `nominal=yes` performs nominal fault free simulation along with fault simulation.
- `faultsid=[...]` specifies the indexes of faults from the list to be considered during simulation.
- `faultnames=[...]` specifies the names of the faults from the list to be considered during simulation.
- `faultsinst=[...]` specifies the list of instances to be considered during fault simulation.

The following is an example of running the transient simulation for all faults blocks and the fault-free transient before the direct fault analysis starts.

```
Directfault sweep faults=[*] nominal=yes {

tran1 tran start=0 stop=3e-3 annotate=status

}
```

Running Transient Fault Analysis

Spectre transient fault analysis requires a complete simulation test bench with fault list, test point(s), test signal(s)/port(s), and the fault method defined using the tran analysis statement. Fault analysis is ignored when this critical information is missing.

To perform transient fault analysis, you need to perform the following:

- Specify the test signals/ports to be checked using the save statement, as shown below.

  ```
  save dd d gg g s
  ```

- Specify the test points and fault method in the transient statement, as shown below.

  ```
  leadtime tran start=0 stop=12e-3 errpreset=conservative maxiters=5
  + faultstart=0.5m faultstep=0.5m faultstop=7.5m + faultmethod=leadtime
  faultleadtime=0.1m
  ```

  Note: The faultstop time must not be beyond the transient stop time.

- Run the simulation, as shown below.

  ```
  %spectre +aps test_fault.scs +log test_fault.out -outdir out_faults
  ```

  Note: The command-line option to run the fault simulation is the same as transient simulation. All the commands to adjust accuracy and performance for transient simulation are also applied to transient fault simulation.

- To run a full fault simulation with the faults injected at time zero and enable auto-stop, use the examples of the transient statement and the Spectre command, as shown below.

```
timezero tran start=0 stop=12e-3 errpreset=conservative maxiters=5
+annotate=status faulttimes=[7.5m] faultmethod=timezero
+faultautostop=all
```

```
%spectre +aps test_fullfault.scs +log test_fullfault.out -outdir out_full/
```

5. **View the fault simulation results** saved in a table file where solutions at the test points are grouped for each fault. The name of the table is *netlistname.tranname.table*.

   The following is a sample table file:

```
title: Fault analysis
analysis: transient
undefined: 1.60000E+38


parameters:nominal=0
 Test_Points    dd    d    gg    g    s    resistance  iterations
5.000000E-04      5.000E+00      4.450E+00      3.125E+00      1.562E+00      5.495E-02      1.6
7.500000E-03      5.000E+00      4.874E+00      1.875E+00      9.375E-01      1.263E-02      1.6

1: bridge_1 ( g 0 ) r=10
5.000000E-04      5.000E+00      5.000E+00      3.125E+00      3.064E-02      1.072E-09
7.500000E-03      5.000E+00      5.000E+00      1.875E+00      1.838E-02      7.194E-10

2: bridge_2 ( dd 0 ) r=10
5.000000E-04      5.000E+00      4.450E+00      3.125E+00      1.562E+00      5.495E-02
7.500000E-03      5.000E+00      4.874E+00      1.875E+00      9.375E-01      1.263E-02

3: bridge_3 ( d 0 ) r=10
5.000000E-04      5.000E+00      4.901E-02      3.125E+00      1.562E+00      4.953E-03
7.500000E-03      5.000E+00      4.926E-02      1.875E+00      9.375E-01      2.498E-03

4: short_1 ( g 0 ) r=100
5.000000E-04      5.000E+00      5.000E+00      3.125E+00      2.604E-01      9.802E-07
7.500000E-03      5.000E+00      5.000E+00      1.875E+00      1.563E-01      4.275E-08

5: short_2 ( dd 0 ) r=100
5.000000E-04      5.000E+00      4.450E+00      3.125E+00      1.562E+00      5.495E-02
7.500000E-03      5.000E+00      4.874E+00      1.875E+00      9.375E-01      1.263E-02

6: short_3 ( d 0 ) r=100
5.000000E-04      5.000E+00      4.227E-01      3.125E+00      1.562E+00      3.500E-02
7.500000E-03      5.000E+00      4.445E-01      1.875E+00      9.375E-01      1.105E-02

7: open_1 ( d ) r=1e+09 { Mn:1 }
5.000000E-04      5.000E+00      5.000E+00      3.125E+00      1.562E+00      5.589E-07
```

   The first group with parameter nominal=0 shows the faultless result and serves as golden reference data. The signals specified in the save statement are the fault results

that are printed in rows and are followed by fault resistance for bridges or opens and iteration number taken by the fault simulation to complete.

6. Post-process the results data saved by the Spectre simulation

After the fault analysis is run, the list of faults is reported in circuit inventory and design checks inventory saved in the log file.

```
Circuit inventory:
        nodes 6
        assert 5
        mos11010 1
        resistor 6
        vsource 2
Design checks inventory:
        bridge 10
        assert 3
        open 12
```

Post-processing the results of a direct fault analysis

For direct fault analysis, the result data is dumped in the raw directory `out_dfa/direct_fault.raw/`.

You can use the Spectre binary `spectre_fsarpt` to generate the functional safety report based on the assert violation for Direct Fault Analysis on screen, or output to a file. The script reads the log file and looks for the path of sqldb file for violations.

```
%.../bin/spectre_fsarpt out_dfa/direct_fault.log
%.../bin/spectre_fsarpt out_dfa/direct_fault.log -o report_file.txt
```

Post-processing the results of a transient fault analysis

For transient fault analysis, the Spectre log file also reports the transient fault simulation by printing the fault numbers which failed to converge at every test point(s).

Depending on the input statements given in the netlist file, a few report tables are also generated in the `out_faults` directory. For example:

```
tfa_test_fault.leadtime.table
tfa_test_fault.linear.table
```

You can use the Spectre binary `spectre_ddmrpt`, to evaluate the fault table file and generate the defect detection matrix. An example report is shown below.

```
Detection Matrix:
+----------+------------+------------+------------+---------+¶
|          | 0.0005/d   | 0.0045/d   | 0.0075/d   | Total   |¶
|----------+------------+------------+------------+---------|¶
| bridge_1 | D          | G          | G          | D       |¶
| bridge_2 | G          | G          | G          | G       |¶
| bridge_3 | D          | G          | D          | D       |¶
| open_1   | G          | D          | G          | D       |¶
| open_2   | D          | G          | G          | D       |¶
| open_3   | D          | G          | G          | D       |¶
| Coverage | 66.67%     | 16.67%     | 16.67%     | 83.33%  |¶
+----------+------------+------------+------------+---------+¶
Solution Matrix:
+----------+------------+------------+------------+¶
|          | 0.0005/d   | 0.0045/d   | 0.0075/d   |¶
|----------+------------+------------+------------|¶
| nominal  | 4.45       | 1.6e-07    | 4.874      |¶
| bridge_1 | 4.907      | -3.857e-07 | 5.058      |¶
| bridge_2 | 4.45       | 1.6e-07    | 4.874      |¶
| bridge_3 | 3.85       | 3.186e-09  | 4.274      |¶
| open_1   | 4.456      | -1.391e-06 | 4.876      |¶
| open_2   | 4.908      | -3.874e-07 | 5.063      |¶
| open_3   | 4.907      | -3.856e-07 | 5.059      |¶
+----------+------------+------------+------------+¶
Legend:

   G - Good, measurement consistent with fault-free circuit within error bound

   D - Detected with violations of error bound

   N - Not testable (error occurred during analysis, e.g. non-convergence)
```

For more details, refer to the following topics: in *Spectre® Classic Simulator, Spectre Accelerated Parallel Simulator (APS), and Spectre Extensive Partitioning Simulator (XPS) User Guide*:

- Fault Analysis
- Viewing Data Output of Transient Fault Analysis
- Post-processing for Functional Safety Report

2

Electrothermal Analysis

Many analog circuits for complex technologies operate at high power. The combination of high-power dissipation in a high-temperature environment can result in thermal over-stress. The stress of operating at high temperature can cause devices that have been designed for a long operating lifetime to fail sooner. The electrothermal analysis feature of the Legato Reliability solution enables designers to prevent thermal over-stress and avoid such premature failures.

In electrothermal analysis, a simulation is performed using Cadence Thermal Technology that delivers true electrothermal co-simulation. In this flow, thermal technology is used to create a thermal model of the die, based on the chip structure and thermal properties of the die stack. This model is a thermal equivalent circuit of the die. Electrothermal analysis uses the electrical and thermal descriptions of the circuit, the netlist, and the thermal model to perform co-simulation. It considers the thermal interactions between devices. In addition, it considers all the power sources in the chip, including transistors, resistors, and other devices.

The electrothermal analysis consists of the following components:

- Spectre APS for electrical simulation

  Spectre APS is used for electrical simulation, which computes the power of each power dissipating instances in the circuit: MOS transistors, resistors, diodes, and bipolar transistors.

- Cadence Thermal Technology

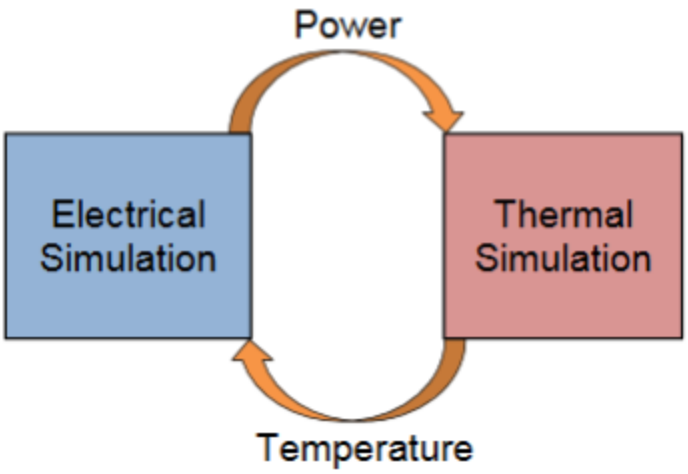
  The thermal technology is called internally by Spectre APS to perform thermal extraction of the die and to create the thermal model used in simulation.

- Spectre APS for thermal simulation

  Spectre APS is used for thermal simulation. It calculates the instance temperatures based on the power consumption of the instances. The thermal solver is built into the electrothermal simulation and is not available as a standalone offering.

Spectre APS automatically maps the information from the electrical simulator, power, to the thermal solver and maps the information from the thermal solver, temperature, back to the electrical simulator.

Electrothermal analysis starts with an electrical simulation and uses the instance power to drive a thermal solution to calculate the temperature of the instances on the die. If the temperatures converge, the simulation is complete. If not, the thermal and electrical simulation process continues until the results converge to a stable solution that satisfies both the circuit electrical power and the circuit temperature condition. This process is called electrothermal co-simulation or electrothermal simulation.



Electrothermal Analysis Methods

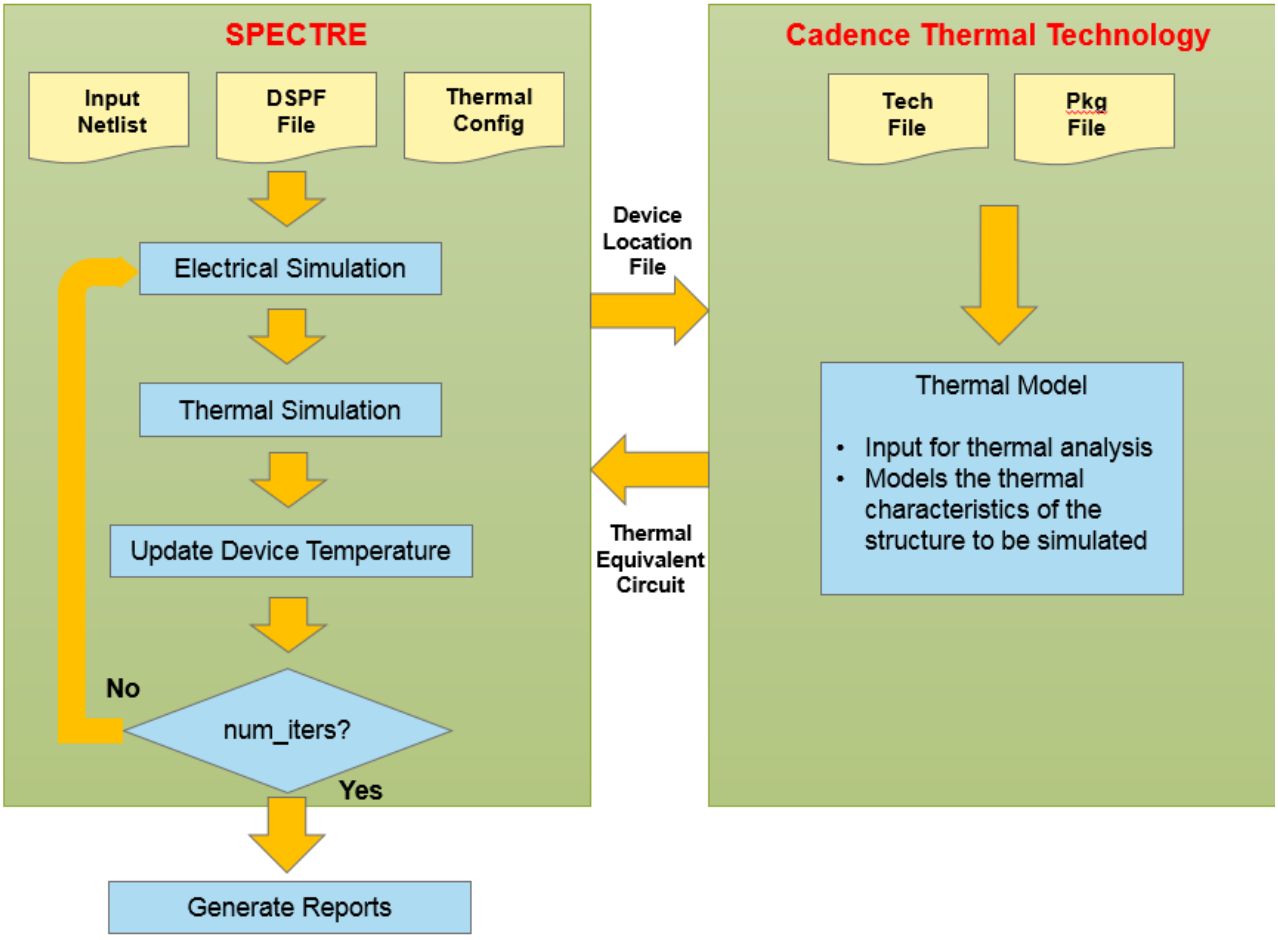
Electrothermal analysis supports the following two methods:

Steady-State Thermal Analysis

In steady-state thermal simulation, electrothermal simulation starts with an electrical simulation with the device temperatures initially set to the ambient value.

Upon completion of the electrical simulation, the average power of all the devices is computed and sent to the thermal solver to compute the temperature rise for each device as a result of the steady-state instance power. The calculated instance temperatures are then used to update the instance temperatures for the next iteration of the electrothermal simulation. The process continues until the user-specified iteration count is reached. The output of the steady-state thermal simulation is a text report, which lists the temperature rise and the average power of each device in the DSPF netlist.

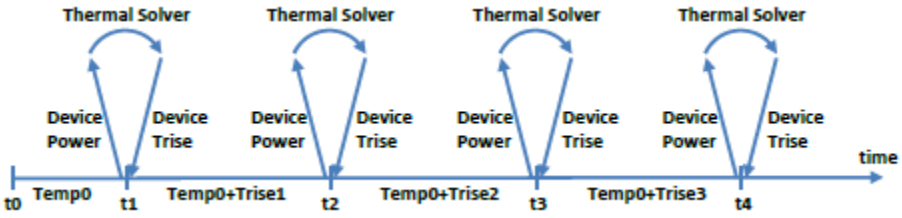
The thermal model is then used in thermal simulation to simulate the temperature rise due to device power dissipation.



Dynamic Thermal Analysis

Dynamic thermal analysis is run by performing thermal solving at each time step of the Spectre transient analysis. Because thermal variation is usually much slower than electrical variation, it is assumed that convergence of electrothermal co-simulation is always achieved with just one iteration at each time step. Therefore, no iterations are needed for the

electrothermal co-simulation during dynamic thermal analysis. This greatly increases the efficiency of the dynamic thermal analysis algorithm without introducing significant errors.



Performing Electrothermal Analysis in Spectre Command-Line Mode

Note: Ensure that you are using Spectre 18.1 ISR5 or a higher version.

To perform electrothermal analysis in Spectre command-line mode, follow these steps:

1. Prepare the simulation testbench including the postlayout data of the design, and the DSPF files with device location and dimension information for the transistors, resistors, diodes, and so on, in the instance section. A `dspf_include` statement is required in the electrothermal analysis flow to identify the DSPF data.

   ```
   dspf_include "sram.spf" (Spectre syntax)
   .dspf_include "sram.spf" (SPICE syntax)
   ```

   `dspf_include` provides special features for reading the DSPF format data, for example, port order adjustments, or handling of duplicated subcircuits. Do not use the `include`/`.include` commands to read the DSPF format data because these commands do not have the special functions of the `dspf_include`/`.dspf_include` command.

   Note: After setting up the testbench, you should perform a regular (non-electrothermal) postlayout simulation with the spectre command to ensure that the testbench contains no error, and the circuit behaves as expected.

   ```
   % spectre +aps input.scs (Spectre test bench)
   ```

2. Add a reliability block containing the thermal analysis statement which defines the thermal configuration file, as shown below.

   ```
   rel reliability
   {
       myThermal thermal config "thermal.conf" <options>
       {
           tran_thermal tran stop=100n
   ```

```
    }
}
```

The thermal configuration file consists of two parts, the location of the data files required for thermal analysis, and additional options. The following two files are required for electrothermal analysis:

- Thermal technology file
- Thermal package file

The thermal technology file is defined with the keyword `tech_file` in the thermal configuration file and describes the die stack. It is the thermal equivalent of the ICT file and contains the thickness and the thermal properties of the materials of the die stack.

```
thermal tech_file = "tech.txt"
```

The thermal package file is defined with the keyword `package_file` in the thermal configuration file and describes the thermal characteristics of the package, which define the boundary conditions for thermal analysis.

```
thermal package_file = "pkg.txt"
```

3. Run the simulation, as shown below.

```
% spectre +aps input.scs
```

Note: The electrothermal analysis is enabled by the presence of the `reliability` block. All electrothermal related options are defined either as options to the thermal analysis statement or in the thermal configuration file.

Viewing Electrothermal Analysis Output in Spectre Results

As the simulation runs, Spectre outputs messages to the screen and the simulation log file that shows the progress of the simulation and provides statistical information. After the simulation completes, electrothermal analysis generates a report file with the extension `.thermal_rpt` and the iteration count in the `raw` file directory after each iteration. It reports the temperature rise and power value on each device in the DSPF file. The last of these

reports can be used as the final result of the electrothermal analysis. The following is a sample report that is generated

```
Table Legends:
 Device           : Name of Device.
 Trise      (Cent): Temperature changes due to thermal analysis.
 Pwr        (Watt): Average Power.
 Trise_range(Cent): Minimum and Maximum of Trise.
 Pwr_range  (Watt): Minimum and Maximum of Power.
 Total_Pwr        : Sum of the Power for all of devices.
 Number_of_Devices: The number of devices display.
 Sorting          : The order of devices are sorted by Trise or Pwr.

+--------------------------------------------------------------------+

Trise_range       = [ 7.917030e+01 ~ 7.922372e+01 ]
Pwr_range         = [ 1.261553e-15 ~ 2.110023e-04 ]
Total_Pwr         = 3.394335e-03
Number_of_Devices = 3094
Sorting           = Trise

   Device                          Trise             Pwr
------------                    ------------      ------------
I0.MMN_cap@15.m1                7.922372e+01      4.719894e-13
I0.MMN_cap@16.m1                7.922372e+01      4.719641e-13
I0.MMN_cap@17.m1                7.922371e+01      5.043528e-13
I0.MMP_out.m1                   7.922292e+01      2.047078e-04
I0.MMP_out@10.m1                7.922292e+01      2.047007e-04
I0.MMP_out@11.m1                7.922292e+01      2.046164e-04
I0.MMP_out@12.m1                7.922292e+01      2.047996e-04
I0.MMP_out@13.m1                7.922292e+01      2.051595e-04
I0.MMP_out@14.m1                7.922292e+01      2.059644e-04
I0.MMP_out@7.m1                 7.922292e+01      2.043795e-04
I0.MMP_out@8.m1                 7.922292e+01      2.044011e-04
I0.MMP_out@9.m1                 7.922292e+01      2.046469e-04
I0.MMP_out@15.m1                7.922281e+01      2.110023e-04
I0.MMN_cap.m1                   7.922237e+01      4.719401e-13
I0.MMN_cap@14.m1                7.922237e+01      4.719389e-13
I0.rrpo413                      7.922227e+01      9.889157e-20
I0.rrpo885                      7.922227e+01      2.679198e-15
I0.rrpog_1_972                  7.922227e+01      1.541470e-15
I0.rrpo2090                     7.922227e+01      1.286555e-14
I0.rrpog_1_2145                 7.922227e+01      1.286552e-15
I0.rrpo2051                     7.922222e+01      1.261556e-14
I0.rrpog_1_2110                 7.922222e+01      1.261553e-15
I0.rrpo2050                     7.922222e+01      2.543516e-14
I0.rrpog_1_2109                 7.922222e+01      2.543509e-15

.............
```

For more information on Electrothermal analysis refer to the *Spectre Thermal Analysis* section in the Spectre Classic Simulator, Spectre Accelerated Parallel Simulator (APS), and Spectre Extensive Partitioning Simulator (XPS) User Guide.

Performing Electrothermal Analysis in Virtuoso ADE Assembler

Pre-requisites for Electrothermal Analysis

Before running electrothermal analysis in Virtuoso ADE Assembler, ensure that you have the following files:

- A thermal configuration file that contains the path to the following two files:
  - A thermal technology file, which describes the die stack
  - A thermal package file, which describes the thermal characteristics of the package that define the boundary conditions for thermal analysis.
- A DSPF file that contains the electrical and geometric information required for thermal analysis

In addition, ensure that you are using Spectre 18.1 ISR5 or a higher version.

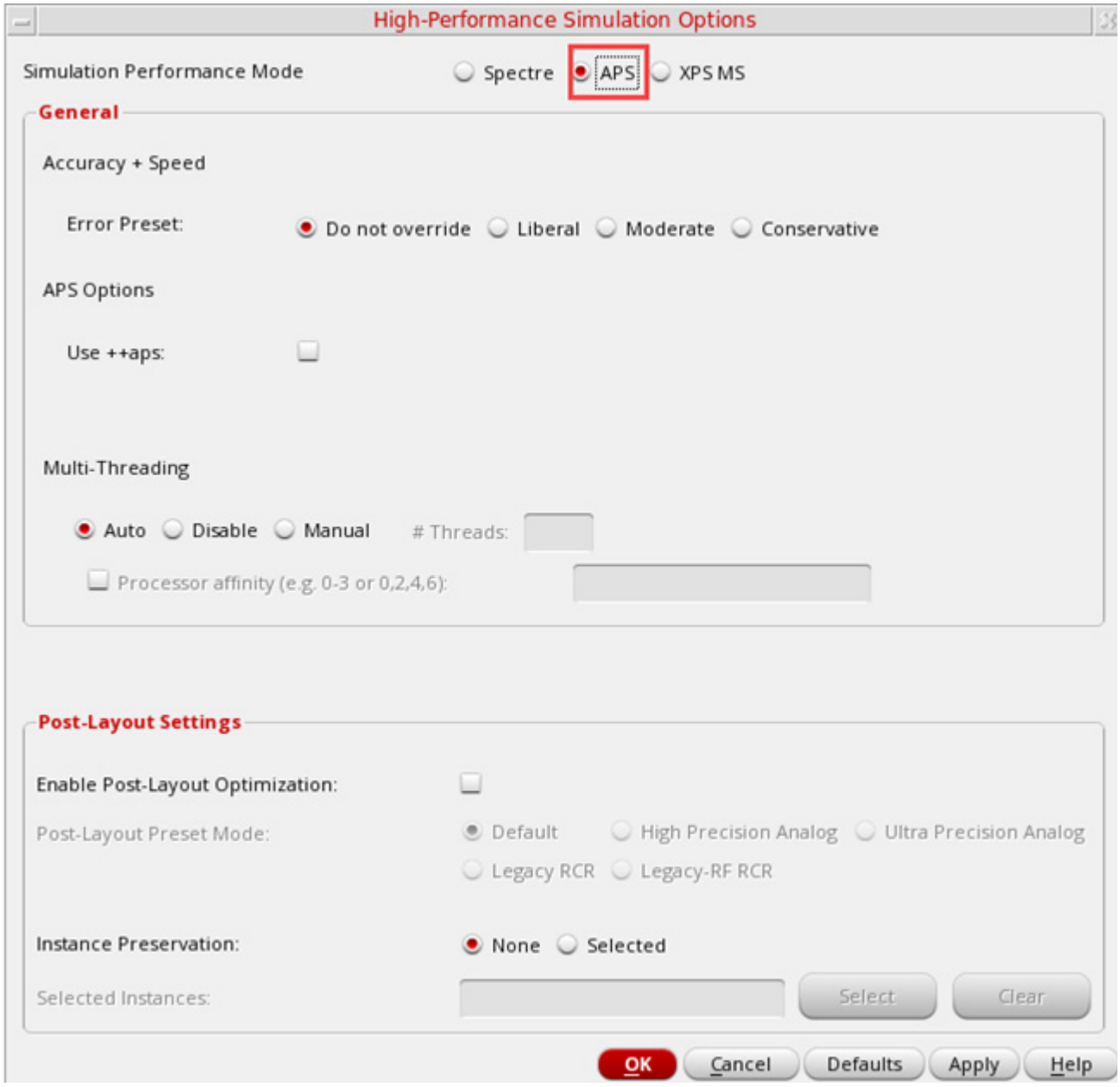
Running the Electrothermal Analysis

To perform electrothermal analysis in Virtuoso ADE Assembler, follow these steps:

1. In the Data View assistant, right-click a test and choose *High-Performance Simulation*.

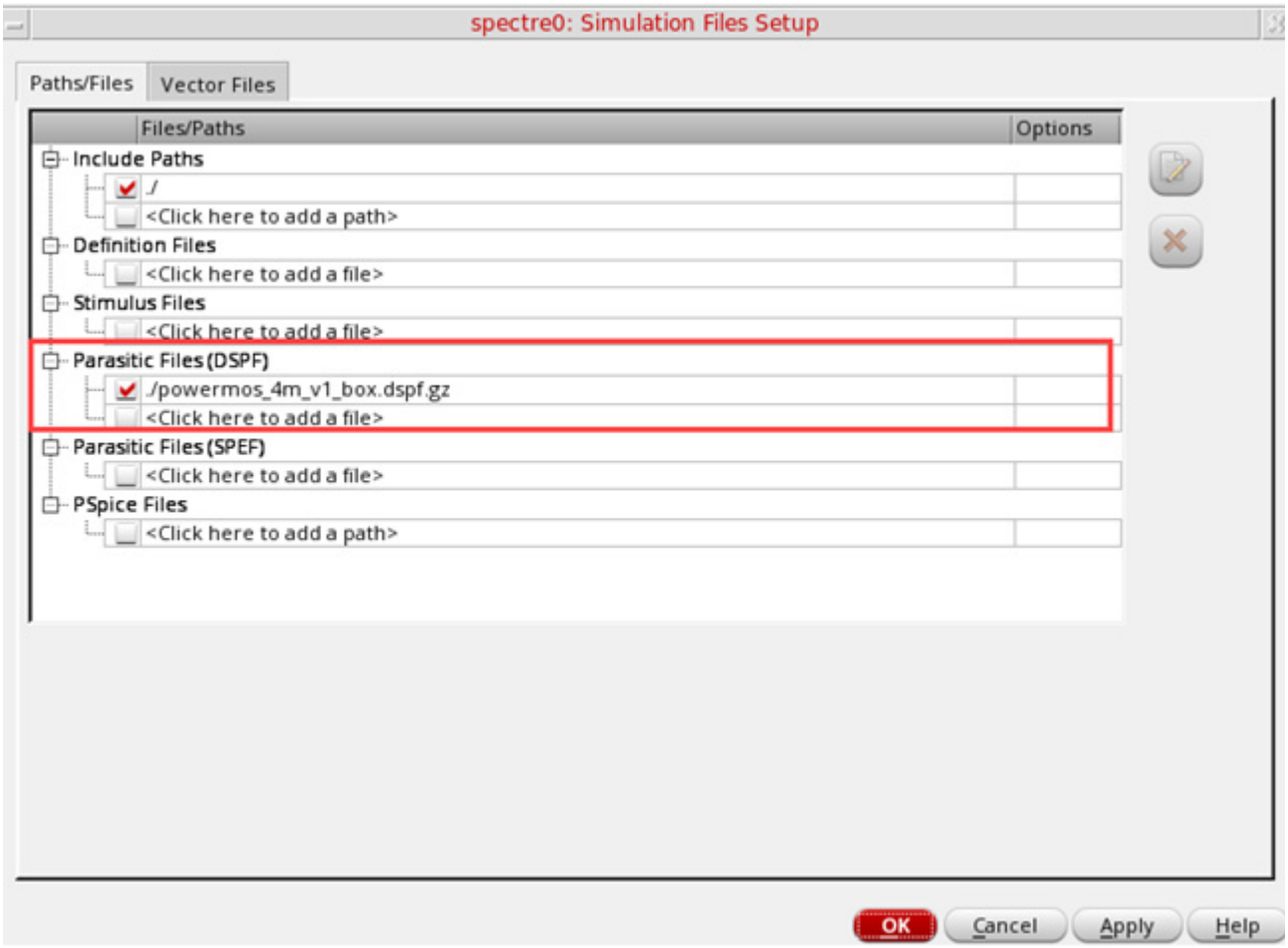
   The High-Performance Simulation Options form is displayed.

2. Enable the Spectre APS mode, as shown below.



3. In the Data View assistant, right-click a test and click Simulation Files to open the Simulation Files Setup form.

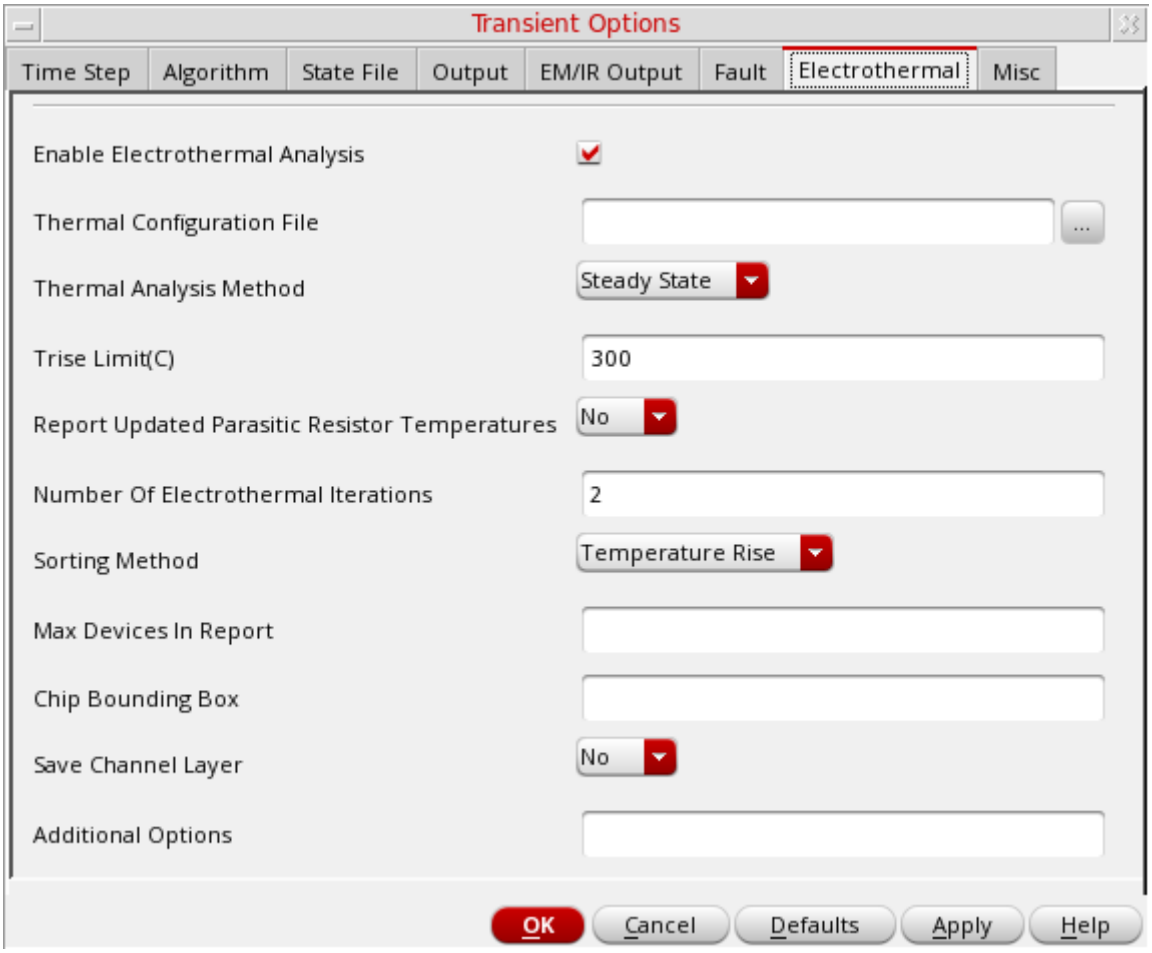
4. In the Simulation Files Setup form, specify the name of the DSPF file mentioned in Pre-requisites for Electrothermal Analysis.



5. In the Data View assistant, expand the *Analysis* tree for a test, and click *Click to add analysis* to add a tran (transient) analysis.

6. In the Choose Analysis form, select *tran*, and click *Options* to open the Transient Options form.

7. Open the *Electrothermal* tab and check the *Enable Electrothermal Analysis* check box.



The other fields on this form are enabled.

8. Perform the following steps on the *Electrothermal* tab:

   a. Specify the path to the thermal configuration file mentioned in Pre-requisites for Electrothermal Analysis.

   b. Choose the required method, `Steady State` or `Dynamic State`, in the *Thermal Analysis Method* drop-down list.

   For details about the thermal analysis methods, refer to Electrothermal Analysis Methods.

   c. Review and if required, modify other settings on this tab.

   The following table describes the other fields on the *Electrothermal* tab:

Field	Description
Trise Limit	Specifies the upper limit value to be used for updating the device temperatures. This helps to prevent device temperatures from going out of valid range during electrical simulation. Unit is degree centigrade. Default value: `300`
Report Updated Parasitic Resistor Temperatures	Specifies whether or not to report the resistor temperature values in thermal reports. Default value: `Yes`
Number of Electrothermal Iterations	Specifies the number of iterations to run for in steady state thermal analysis method. Default value: `2`
Sorting Method	Device sorting order in debug reports (`.dbg_trise_iter*` and `.thermal_pwr.iter*`) for steady state thermal analysis. Possible values: ■ `Temperature Rise (trise values)`. This is the default value. ■ `Power`
Max Devices in Report	Specifies the number of instances to be included in steady state thermal analysis reports.
Chip Bounding Box	Specifies a vector of four numbers [llx lly urx ury] representing a user-defined chip bounding box. By default, the chip bounding box is calculated from the device geometries so that a tight bounding box containing all the devices is used. Default value: `nil`
Devices in Thermal Report	Specifies a list of instance names for which temperature and power are to be included in dynamic thermal analysis reports. Default value: `nil`

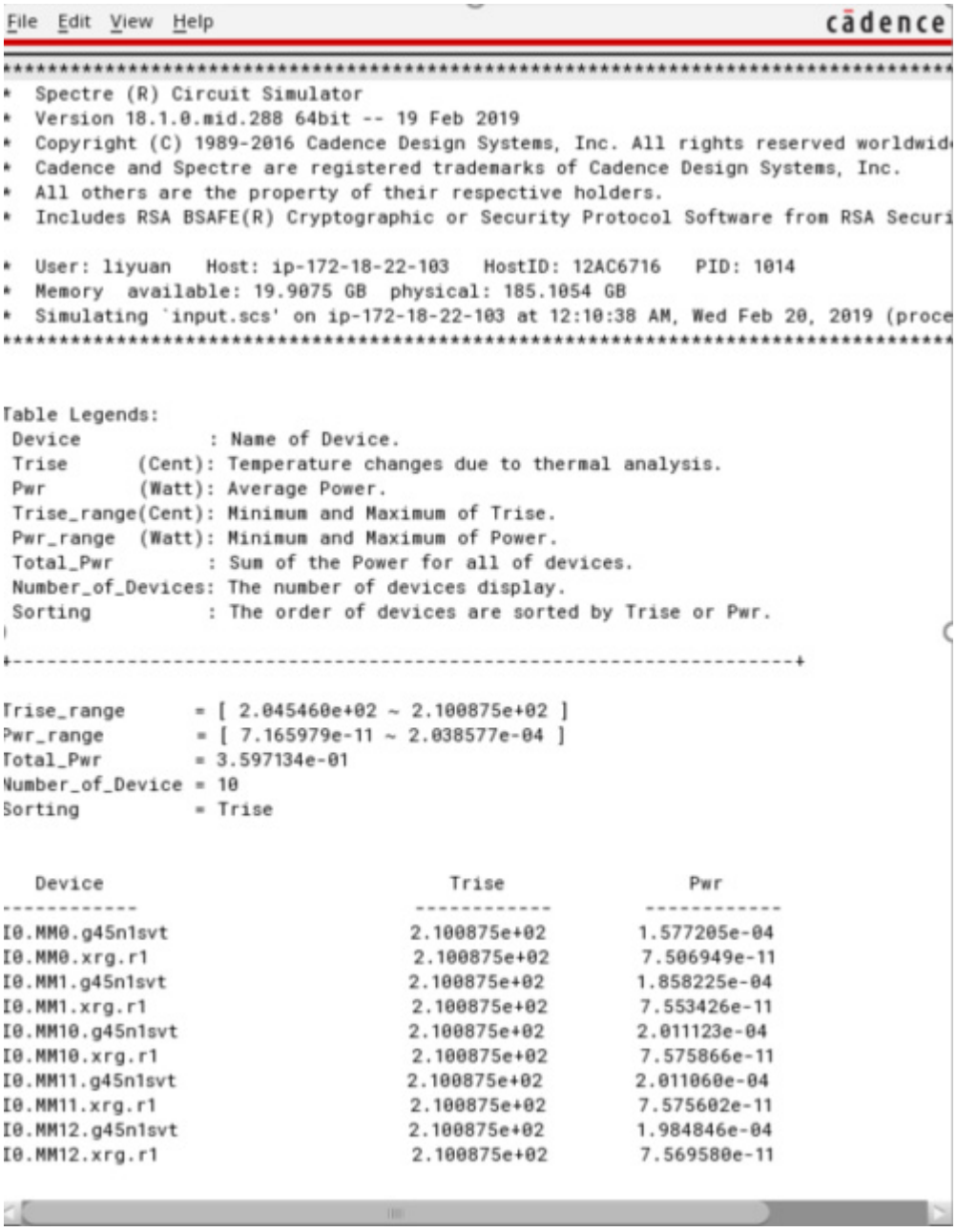
Thermal Time Step	Specifies the thermal time step (in seconds) for dynamic thermal solver. Set this parameter to zero to get identical electrical and thermal time steps. Default value: `1.0`
Trise for Thermal Update	Specifies the Trise threshold for device temperature update. Default value: `1.0`
Save Channel Layer	Specifies whether the Trise data is to be saved for the channel layer only or for all layers. When set to `Yes`, the data is saved only for the channel layer. Otherwise, the data is saved for all layers. Default value: `No`
Additional Options	Placeholder where you can specify any additional option for Spectre Electrothermal analysis.

9. Click *OK* to save the settings and close the Transient Options form.
10. Run simulation.

Viewing Results of Electrothermal Analysis

After the simulation for a steady state electrothermal analysis is complete, the results are displayed in the *Results* tab.

Right-click on an output value and choose *Electrothermal Report* to view a detailed report, as shown below.

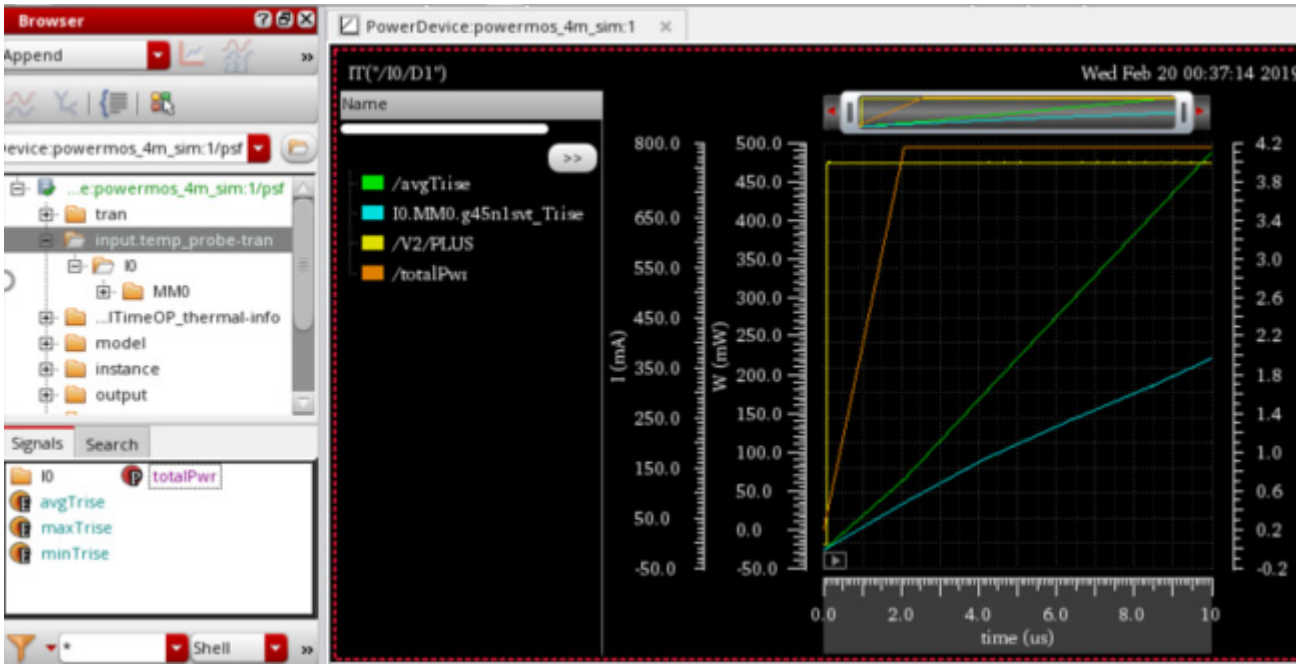


File Edit View Help cadence

```
*********************************************************************************
*  Spectre (R) Circuit Simulator
*  Version 18.1.0.mid.288 64bit -- 19 Feb 2019
*  Copyright (C) 1989-2016 Cadence Design Systems, Inc. All rights reserved worldwid
*  Cadence and Spectre are registered trademarks of Cadence Design Systems, Inc.
*  All others are the property of their respective holders.
*  Includes RSA BSAFE(R) Cryptographic or Security Protocol Software from RSA Securi

*  User: liyuan   Host: ip-172-18-22-103   HostID: 12AC6716   PID: 1014
*  Memory  available: 19.9075 GB  physical: 185.1054 GB
*  Simulating `input.scs' on ip-172-18-22-103 at 12:10:38 AM, Wed Feb 20, 2019 (proce
*********************************************************************************


Table Legends:
 Device           : Name of Device.
 Trise      (Cent): Temperature changes due to thermal analysis.
 Pwr        (Watt): Average Power.
 Trise_range(Cent): Minimum and Maximum of Trise.
 Pwr_range  (Watt): Minimum and Maximum of Power.
 Total_Pwr        : Sum of the Power for all of devices.
 Number_of_Devices: The number of devices display.
 Sorting          : The order of devices are sorted by Trise or Pwr.

+-------------------------------------------------------------------+

Trise_range      = [ 2.045460e+02 ~ 2.100875e+02 ]
Pwr_range        = [ 7.165979e-11 ~ 2.038577e-04 ]
Total_Pwr        = 3.597134e-01
Number_of_Device = 10
Sorting          = Trise


   Device                          Trise            Pwr
------------                    ------------     ------------
I0.MM0.g45n1svt                 2.100875e+02     1.577205e-04
I0.MM0.xrg.r1                   2.100875e+02     7.506949e-11
I0.MM1.g45n1svt                 2.100875e+02     1.858225e-04
I0.MM1.xrg.r1                   2.100875e+02     7.553426e-11
I0.MM10.g45n1svt                2.100875e+02     2.011123e-04
I0.MM10.xrg.r1                  2.100875e+02     7.575866e-11
I0.MM11.g45n1svt                2.100875e+02     2.011060e-04
I0.MM11.xrg.r1                  2.100875e+02     7.575602e-11
I0.MM12.g45n1svt                2.100875e+02     1.984846e-04
I0.MM12.xrg.r1                  2.100875e+02     7.569580e-11
```

You can see a similar report for the dynamic electrothermal analysis too. In addition, you can plot graphs to show the Trise values for the devices specified in the *Devices in Thermal Report* field on the *Electrothermal* tab of the Transient Options form.

To plot the graphs for the saved devices, open Results Browser and plot the results of transient analysis, as shown below.



Alternatively, you can create outputs on the Outputs Setup tab by using the following expressions:

```
(getData "totalPwr" ?result "input.temp_probe-tran")
(getData "maxTrise" ?result "input.temp_probe-tran")
(getData "minTrise" ?result "input.temp_probe-tran")
(getData "avgTrise" ?result "input.temp_probe-tran")
```

After the simulation run is complete, these expressions result into waveforms that you can double-click and plot in Virtuoso Visualization and Analysis XL.

3

Advanced Aging Analysis

Device wear-out results in end-of-life failures. To extend device lifetime, designers need to accurately predict the effect of stress on a device lifetime. Until now, designers had to account for each source of device degradation in isolation.They use reliability analysis to estimate device in one analysis due to electrical stress, then re-rate the lifetime based on estimated die temperature and the effect of process variation.

The Advanced aging analysis feature of the Cadence Reliability solution enables designers to accurately predict product wear-out, and therefore allows them to achieve their design target for extended lifetime of devices without over-designing each transistor, which results in increased die area and increased product cost.

Advanced aging analysis has been enhanced to include other effects that contribute to device degradation, besides electrical stress, such self-heating and process variation.

Self-Heating and Aging Flows

You can enable the self-heating flow by setting the value of `type` parameter in the `rel_mode` control statement to `she`. For example:

```
rel_mode type=she
```

The self-heating and aging flow can be enabled by setting the `type` parameter in the `rel_mode` control statement to `all`, as shown below.

```
rel_mode type=all
```

You can output the device power in the self-heating flow by using the `output_she_power` control statement as follows:

```
output_she_power value=yes
```

If you use the URI self-heating flow, you can specify the `uri_lib file=<`*`name of the Library`*`>` control statement to define the URI library to be used.

Note: For self-heating calculation, you can use URI or the internal self-heating model (agelevel=3). When using URI, ensure that a standalone agelevel is used for self-heating calculation. Do not use URI and the internal self-heating model at the same time.

Aging Monte Carlo Analysis

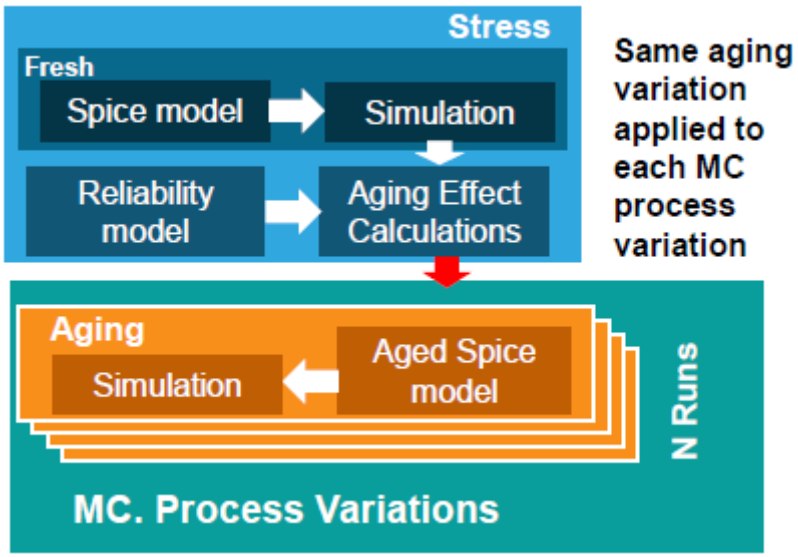
To account for process variation, you can combine advanced aging analysis with Monte Carlo analysis. As a result, you can specify the process variation models and aging models together in one simulation run.

The advanced aging analysis supports the following two flows to perform aging MonteCarlo analysis for the agemos and appendage flow:

- Single Stress (1+N) flow
- Multi-Stress (N+N) flow

Single Stress (1+N) flow

In this flow, a nominal stress analysis is run first. Next, MonteCarlo analysis is performed on the aging analysis. No variations are performed on the stress analysis. This is shown in the figure below.



The reliability block for the stress simulation can be defined as follows:

```
mc1 montecarlo numruns=0 seed=12345 variations=all sampling=standard \
    donominal=yes savefamilyplots=yes savedatainseparatedir=yes \
rel reliability {
    // reliability control statements
    age time = [10y]
    deltad value = 0.1
```

```
    report_model_param value=yes
    simmode type=stress

    // fresh/stress simulation.
    tran_stress tran start = 0 step = 1u stop = 10u
}
```

The reliability block for the aging MonteCarlo simulation can be defined as follows:

```
mc1 montecarlo numruns=100 seed=12345   variations=all   sampling=standard \
savedatainseparatedir=yes savefamilyplots=yes {
    rel reliability {
        // reliability control statements
        age time = [10y]
        deltad value = 0.1
        report_model_param value=yes
        // fresh/stress simulation.
        simmode type=aging file="stress.bs0"
        // aging simulation statements.
        tran_aged tran start = 0 step = 1us stop = 10us
    }
}
```

1+N Flow in ADE Assembler

You can use the 1+N flow in ADE Assembler by performing the following steps:

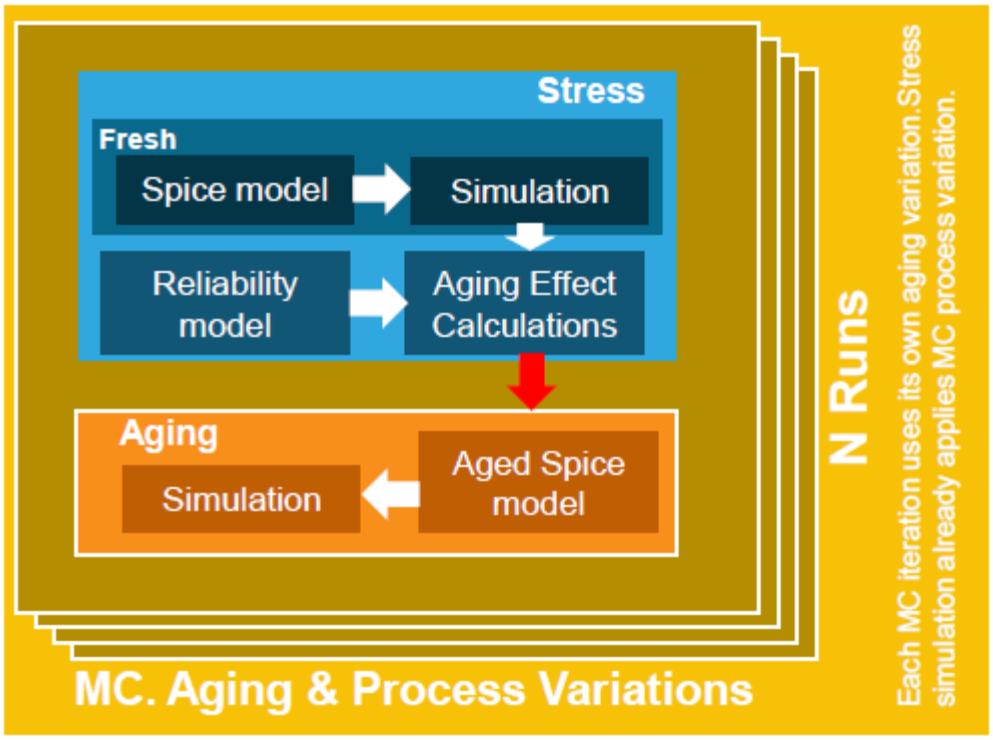
1. Set up reliability analysis by using the *Reliability Analysis* tree of the Data View Assistant.
2. Enable Monte Carlo Sampling run mode and set up the Monte Carlo simulation options.
3. Run simulation.

In 1+N mode, a nominal Monte Carlo simulation is run for stress followed by Monte Carlo aging simulations. It is optional to run Monte Carlo fresh simulation, but is supported. Each fresh or aged pair uses the same statistical sample.

Also see: Important Points to Note

Multi-Stress (N+N) flow

In this flow, Monte Carlo analysis is performed on both stress and aging simulations, as shown in the figure below.



In this flow, simulation can be run on a single netlist or separate netlists. The following is an example of running stress and aging analyses in the same netlist:

```
mc1 montecarlo numruns=100 seed=12345 variations=all  sampling=standard \
savedatainseparatedir=yes savefamilyplots=yes {

    rel reliability {
        // reliability control statements
        age time = [10y]
        deltad value = 0.1
        report_model_param value=yes
        // fresh/stress simulation.
        tran_stress tran start = 0 step = 1u stop = 10u
        // aging simulation statements.
        tran_aged tran start = 0 step = 1us stop = 10us
    }
}
```

The following is an example of running stress and aging analyses in separate netlists:

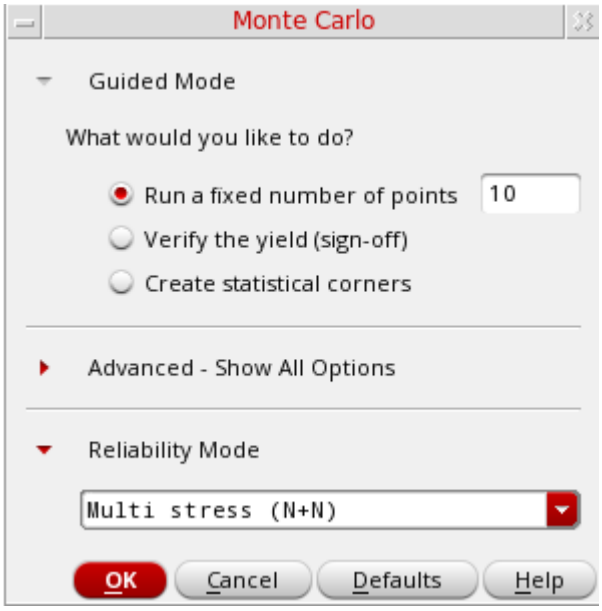
```
mc1 montecarlo numruns=100 seed=12345 variations=all sampling=standard
savedatainseparatedir=yes savefamilyplots=yes {
rel reliability {
// reliability control statements
    age time = [10y]
    deltad value = 0.1
    uri_lib file="libURI.so" uri_mode=appendage
    report_model_param value=yes
    simmode type=stress
// fresh/stress simulation.
    tran_stress tran start = 0 step = 1u stop = 10u
    }
}
mc1 montecarlo numruns=100 seed=12345 variations=all sampling=standard
savedatainseparatedir=yes savefamilyplots=yes {
rel reliability {
// reliability control statements
    age time = [10y]
    deltad value = 0.1
    report_model_param value=yes
    uri_lib file="libURI.so" uri_mode=appendage
// fresh/stress simulation.
    simmode type=aging file="stress.bs0"
// aging simulation statements.
    tran_aged tran start = 0 step = 1us stop =10us
}
}
```

N+N Flow in ADE Assembler

You can use the N+N flow in ADE Assembler by performing the following steps:

1. Set up reliability analysis by using the *Reliability Analysis* tree of the Data View Assistant.
2. Select the `Monte Carlo Sampling` run mode in the Run Mode toolbar.

3. In the Monte Carlo options form, select `Multi stress (N + N)` in the drop-down list in the *Reliability Mode* section.



Note: You can set the default value for this drop-down list by using the maestro.monte reliabilitymode environment variable in .cdsenv.

4. Run simulation.

As shown in the example of results given below, in N+N mode, the fresh, stress, and aging analyses are run for all Monte Carlo iterations.

Point		Test	Output	Nominal	Spec	Weight	Pass/Fail
All		Osc_Aged	Max_Mas	2.572	< 3		pass
All		Osc_Fresh	Max_Mas	2.572	< 3		pass
All		Osc_Stress	Max_Mas	2.572	< 3		pass

)/120 rows

Point	Age	Test	Output	Nominal	Spec	Weight	Pass/Fail
ilter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
arameters: mc_iteration=1							
1	fresh	Osc_Fresh	/out				
1	fresh	Osc_Fresh	Frequency	15.29M	> 15M		pass
1	fresh	Osc_Fresh	yMax	2.527	< 3		pass
1	stress	Osc_Stress	/out				
1	stress	Osc_Stress	Frequency	15.29M			
1	stress	Osc_Stress	yMax	2.527	< 3		pass
1	10 yr	Osc_Aged	/out				
1	10 yr	Osc_Aged	Frequency	14.29M	> 15M		near
1	10 yr	Osc_Aged	yMax	2.57	< 3		pass
arameters: mc_iteration=2							
2	fresh	Osc_Fresh	/out				
2	fresh	Osc_Fresh	Frequency	15.14M	> 15M		pass
2	fresh	Osc_Fresh	yMax	2.526	< 3		pass
2	stress	Osc_Stress	/out				
2	stress	Osc_Stress	Frequency	15.14M			
2	stress	Osc_Stress	yMax	2.526	< 3		pass
2	10 yr	Osc_Aged	/out				
2	10 yr	Osc_Aged	Frequency	14.13M	> 15M		near
2	10 yr	Osc_Aged	yMax	2.529	< 3		pass
arameters: mc_iteration=3							
3	fresh	Osc_Fresh	/out				

MonteCarlo.0

Important Points to Note

- To run reliability analysis with Monte Carlo, ADE Assembler requires a Virtuoso_Variation_Option (VVO) license.The basic Monte Carlo flow is supported with reliability analysis
- Non-statistical corners and global variable sweep are supported

- Manual creation of statistical corners (both sequence and parameter value based) is supported
- Mismatch contribution is supported
- Sample re-ordering, auto-stop, high-sigma and automatic k-sigma statistical corner creation are not supported
- Gradual aging, self-heating, and reliability scenarios are not supported