cadence®

Virtuoso Parasitic Aware Design SKILL Reference

Product Version IC23.1
June 2023

Printed in the United States of America.

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

Contents

1

Parasitic Aware Design Functions

The Parasitic Aware Design flow is available in:

- The Virtuoso® Analog Design Environment (ADE L/XL/GXL)
- The Virtuoso® Schematic Editor (VSE L/XL) applications

This topic lists the Cadence® SKILL functions associated with Virtuoso® Parasitic Aware Design.

Only the functions listed below are supported for public use. All other functions, regardless of their name or prefix, and undocumented aspects of the functions described below, are private and are subject to change at any time.

- aelDisplayOPParam
- aelSumOPParam
- auLvsGetLabelSuffix
- axlGetParasiticViewName
- axlMapInstTermToNet
- axlSetParasiticViewName
- mspsMapNetName
- parCacheFind
- parCacheGet
- parCacheListFilters
- parCacheListModels
- parCachePurge
- parCacheSave
- parDelete

- parFilterCreate
- parFind
- parModelCreateCustom
- parModelCreateNetC
- parModelCreateNetK
- parModelCreateNetL
- parModelCreateNetR
- parModelListSimParams
- parModelListSimSweeps
- parModelUpdateSimParams
- parModelUpdateSimSweeps
- parObjectListFilters
- parObjectListModels
- parRemoveMembers
- parResetAllParams
- parResetParams
- parSetNote
- parUpdateMembers
- parUpdateParams
- sumOPParamV2

For more information, see the *Virtuoso Parasitic Aware Design User Guide*.

Licensing Requirements

To run Virtuoso Parasitic Aware Design from Schematic L, Schematic XL, or Analog Design Environment (ADE) L, you need either an ADE L (95200) or an ADE XL (95210) license. If an ADE L license is available, it is checked out, if not already checked out. If an ADE L license is not found, an ADE XL license is checked out.

To run Virtuoso Parasitic Aware Design from ADE XL or ADE GXL, an additional ADE GXL license token (95220) is checked out.

For information about licensing in the Virtuoso Studio design environment, see *Virtuoso Software Licensing and Configuration Guide*.

Related Topics

Parasitic Aware Design in Virtuoso ADE Explorer, Virtuoso ADE Assembler, and Virtuoso Schematics L/XL

Parasitic Aware Design in ADE Explorer and ADE Assembler

aelDisplayOPParam

```
aelDisplayOPParam(
     instName
     simParam
     [ labelParam ]
     [ resName ]
     )
     => string_list / nil
```

Description

Returns a string list whose elements are the *simParam* of each of the instances being processed.

The instances being processed depend on the given *instName*. The function creates a list with all of the instances being considered. The instance may be a schematic instance (the result of `inst()`), or an extracted instance (for example "`/I0/M0_1_qrc`").

If a schematic instance is given in out-of-context, then the mapped extracted instances are considered, for example if `inst()` is given, the instances considered could be (`"/I0/M0" "/I0/M0_1_qrc" "/I0/M0_2_qrc" "/I0/M0_2_qrc" "I0/M0_3_qrc" "I0/M0_4_qrc"`).

Once the list is created, the *param* specified for each instance is added to the return list. This *simParam* can be any of the simulation parameters.

The default is `id`.

To know more about the function definition, see aelDisplayOPParam.

Arguments

instName	A string that can be a schematic instance, the result of the method `inst()`, or an extracted instance name.
simParam	Any simulation parameter, for example, `id`.
labelParam	A parameter required when the name of the label parameter defined by `opParamExprList` is different than the simulation parameter being processed. For example, if the label parameter is `mFactorF` and the simulation parameter being processed is `id`, then *labelParam* must be given with the value `mFactorF`.

resName	A string used to select the type of results from a particular analysis, for example `dcOpInfo-info`. The type of results available can be obtained using the following command. `results(?noAlias t)` As a default, this input is set to the current type of results.

Value Returned

string_list	A string with a list of comma-separated numbers.
`nil`	The instance failed to map.

Related Topics

Parasitic Aware Design Functions

Specifying Parameters to be Displayed

aelSumOPParam

```
aelSumOPParam(
     instName
     simParam
     [ labelParam ]
     [ resName ]
     )
     => integer / nil
```

Description

Returns a number which is the result of adding the values of the parameters specified by *instName*.

The argument *instName* can be a schematic name which maps to multiple m-factor devices, one device, or a specific extracted name which allows you to display specific m-factor devices values.

To do this, `aelSumOPParam` creates a list with all the instances being considered. The instance may be a schematic instance (the result of `inst()`), or an extracted instance. For example `I0/M0_1_qrc`.

If a schematic instance is given in out-of-context, then the mapped extracted instances are considered, for example if `inst()` is given, the instances considered could be (`"/I0/M0" "/I0/M0_1_qrc" "/I0/M0_2_qrc" "/I0/M0_2_qrc" "I0/M0_3_qrc" "I0/M0_4_qrc"`). Once the list is created, the *param* specified for each instance is added. This *param* can be any of the simulation parameters.

The default is `id`.

To know more about the function definition, see aelSumOPParam.

Arguments

instName	A string that can be a schematic instance, the result of the method `inst()`, or an extracted instance name.
simParam	Any simulation parameter. For example, `id`.
labelParam	A parameter required when the name of the label parameter defined by `opParamExprList` is different from the simulation parameter being processed. For example, if the label parameter is `mFactorF` and the simulation parameter being processed is `id`, then *labelParam* must be given with the value `mFactorF`.

resName	A string used to select the type of results from a particular analysis, for example `dcOpInfo-info`. The type of results available can be obtained using the following command. `results(?noAlias t)` As a default, this input is set to the current type of results.

Value Returned

integer	A number which is the result of adding all of the *simParam* available in the specified *instName*.
`nil`	The instance has failed to map.

Related Topics

Parasitic Aware Design Functions

Specifying Parameters to be Displayed

sumOPParamV2

auLvsGetLabelSuffix

```
auLvsGetLabelSuffix(
     schInstanceName
     param
     )
     => suffix / nil
```

Description

Returns the label suffix that is used when annotating `dcOp` or transient op points.

Arguments

schInstanceName	The schematic instance being annotated.
param	The parameter, for example, `V` (voltage) and `I` (current).

Value Returned

suffix	A label suffix value.
`nil`	The operation was unsuccessful.

Examples

The following is an example of a mFactored transistor using custom labels, where instance `/IO/MO` with parameter `id` returns suffix value of `Sum`.

```
auLvsGetLabelSuffix("/IO/MO" "id")
=> "Sum"
```

Related Topics

Parasitic Aware Design Functions

Backannotation of dcOp / Transient Values for M-Factor Devices

axlGetParasiticViewName

```
axlGetParasiticViewName(
     t_sessionName
     t_flowName
     )
     => t_viewName
```

Description

Gets the name of the parasitic view set for the given flow in ADE XL.

Arguments

t_sessionName	Name of the ADE XL session.
t_flowName	Name of the flow for which you need to get the name of the parasitic view. Valid values are `Estimated`, `Extracted`, or `Layout`.

Value Returned

t_viewName	Name of the parasitic view that is set to be used in the given flow.

Examples

The following example shows how to get the view names by using this function.

```
session = axlGetWindowSession()
=> "session0"
axlGetParasiticViewName("session0" 'Extracted)
=> "av_extracted_rc"
axlGetParasiticViewName("session0" 'Estimated)
=> "estimated"
axlGetParasiticViewName("session0" 'Layout)
=> "netlist_layout"
```

Related Topics

Parasitic Aware Design Functions

axlSetParasiticViewName

axlMapInstTermToNet

```
axlMapInstTermToNet(
     t_instPathName
     t_termName
     [ t_dataDir ]
     [ g_verbose ]
     )
     => t_netName
```

Description

Maps an instance terminal to its corresponding net connection in the configured view, which can be a schematic, a parasitic/LDE, or an extracted view.

This function is useful while doing out-of-context probing with a config view. Instead of directly using net names in calculator expressions (in an ADE output), you can call `axlMapInstTermToNet` from within the expression to dynamically return the name of the net connected to an instance terminal. In this case, even if the configured view is modified and the net connected to a terminal is changed, the calculator function can get the correct net name connected to the given instance terminal.

Arguments

t_instPathName	Path to the instance terminal in the schematic design hierarchy
t_termName	Name of the instance terminal.
t_dataDir	Path to the results directory.
g_verbose	Sets the verbose mode on or off. When set to `t`, the function prints log details with design name, extracted cellview name, and the extracted net name.

Value Returned

t_netName	Net name in the extracted view to which the instance terminal is mapped.

Examples

In this example, a parasitic RC extracted simulation is run in ADE. In the output, the calculator function, `VT`, uses `axlMapInstTermToNet` to use the net name mapped to the instance terminal `I3/MP0:D`.

```
VT(axlMapInstTermToNet( "/I1/I2/I3/MP0" "D" ) )
```

In the above example, the `axlMapInstTermToNet` function internally returns the mapped net name as `"/I1/14:I2|I3|net1"`.

Related Topics

Parasitic Aware Design Functions

axlSetParasiticViewName

```
axlSetParasiticViewName(
     t_sessionName
     t_flowName
     t_viewName
     )
     => t_flowName
```

Description

Sets the name of the parasitic view to be used for the given flow in ADE XL.

Arguments

t_sessionName	Name of the ADE XL session.
t_flowName	Name of the flow for which you need to set a parasitic view. Valid values are `Estimated`, `Extracted`, or `Layout`.
t_viewName	Name of the parasitic view to be used in the given flow.

Value Returned

t_flowName	Name of the flow for which the view name is set.

Examples

The following example describes how to set a view name to be used for the extracted flow.

```
session = axlGetWindowSession()
=> "session0"
axlSetParasiticViewName("session0" 'Extracted "av_extracted_rc")
=> (Extracted)
```

Related Topics

Parasitic Aware Design Functions

axlGetParasiticViewName

mspsMapNetName

```
mspsMapNetName(
     h_hdbConfigId
     t_name
     )
     => t_frag_name / nil
```

Description

Maps a hierarchical schematic net name to a fragment of the equivalent net in the extracted view.

This function is required to be used only with the Virtuoso executable and not with OCEAN executable. OCEAN and ADE data access functions automatically map schematic names into the simulated extracted view and it is no longer necessary to call this function.

Arguments

h_hdbConfigId	Configuration cellview identifier returned by `hdbOpen`.
t_name	Hierarchical schematic net name.

Value Returned

t_frag_name	Hierarchical name of a net fragment in the extracted view that maps to the net name in the schematic view. This can be any one of the net fragments in the extracted view that map to schematic net. If none of the instances in the hierarchical schematic name is bound to an extracted view, the schematic name is returned unmodified. In this case the supplied hierarchical name does not lead into an extracted view. For example, if you have the name `/I0/I1/I2/netA` it means that none of `I0`, `I1`, `I2` have been bound to an extracted view in the config. In this case, the name is returned unmodified as it does not need to be mapped into an extracted view.
`nil`	The schematic name does not match the extracted view. In this case, the supplied name does lead into an extracted view, but the net name cannot be mapped into that view because, for example, a wrong net name is given. For example, if you bind I1 to an extracted view in the previous example, the name `/I0/I1/I2/netA` needs to be mapped now. But, if `netA` does not exist, then the function returns `nil`.

Examples

In these examples, instance `/X1` is bound to an extracted view and instance `/XA` to the schematic view.

```
cfg = hdbOpen("worklib" "TB1_vco_RCXcompare" "config" "r")
=> hdbcConfigType:0x0xbe9d948
```

Here, net `/X1/I15|n3` in the schematic maps to net `/X1/2:I15/n3` in the extracted view.

```
mspsMapNetName( cfg "/X1/I15/n3" )
=> "/X1/2:I15|n3"
```

Here, instance `XA` is bound to the schematic view. Therefore, the name does not need to be mapped.

```
mspsMapNetName( cfg "/XA/I15/n3" )
=> "/XA/I15/n3"
```

Here, `inx2` is not a valid net name in the design.

```
mspsMapNetName( cfg "/X1/I15/inx2" )
=> nil
```

Related Topics

Parasitic Aware Design Functions

parCacheFind

```
parCacheFind(
     t_libName
     t_cellName
     t_viewName
     )
     => cache_id / nil
```

Description

Finds an existing parasitic cache for a given design specified using library, cell, and view names.

This command also works with a single `dbCellViewId` argument (a library, cell, view name triplet).

Arguments

t_libName	Name of the library in which the parasitic cache is located.
t_cellName	Name of the cell in which the parasitic cache is located.
t_viewName	Name of the view in which the parasitic cache is located.

Value Returned

cache_id	The parasitic cache, if the cache has already been built for a SKILL list that contains library, cell, and view names.
`nil`	No cache was found.

Examples

```
parCacheFind( "libName" "cellName" "viewName" )
```

Related Topics

Parasitic Aware Design Functions

parCacheGet

```
parCacheGet(
     t_libName
     t_cellName
     t_viewName
     )
     => cache_id / nil
```

Description

Finds an existing parasitic cache or creates and populates the cache for a given design, specified using library, cell, and view names.

Arguments

t_libName	Name of the library in which the parasitic cache is located or created.
t_cellName	Name of the cell in which the parasitic cache is located or created.
t_viewName	Name of the view in which the parasitic cache is located or created.

Value Returned

cache_id	The parasitic cache is returned if it is already built through a previous call to parCacheGet, otherwise the cache for the given cellview is built and returned.
nil	No cache was found.

Examples

```
cache = parCacheGet( "amsPLL" "vco" "schematic" )
=> ci:0x127cfba0
```

Related Topics

Parasitic Aware Design Functions

parCacheListFilters

```
parCacheListFilters(
     d_cache
     [ g_includeOutOfContext ]
     )
     => filter_id_list / nil
```

Description

Lists all parasitic filters from a given cache.

Arguments

d_cache	The parasitic cache in which the filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
g_includeOutOfContext	If set to `t`, the out-of-context filters are listed.

Value Returned

filter_id_list	An `id` list of all parasitic filters in the cache.
`nil`	No filters were found.

Examples

```
filters = parCacheListFilters( cache )
=> (ci:0x12e4d890 ci:0x12f36620)
filters~>parameters
=> ((("type" enum "R")
     ("subtype" enum "both")
     ("include" enum "all")
     ("threshold" float 0.0)
      )
      (("type" enum "C")
     ("subtype" enum "both")
     ("include" enum "all")
     ("threshold" float 0.0)
      )
   )
```

Related Topics

Parasitic Aware Design Functions

parCacheListModels

```
parCacheListModels(
     d_cache
     [ g_includeOutOfContext ]
     )
     => model_id_list / nil
```

Description

Lists all parasitic models from a given cache.

Arguments

d_cache	The parasitic cache in which the model belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
g_includeOutOfContext	If set to `t`, the out of context models are listed.

Value Returned

model_id_list	An `id` list of all parasitic models in the cache.
`nil`	No filters were found.

Examples

```
models = parCacheListModels( cache )
=> (ci:0x12d4f1c0 ci:0x12f82f30)

models~>type
=> (NetR NetC)
```

Related Topics

Parasitic Aware Design Functions

parCachePurge

```
parCachePurge(
     d_cache
     )
     => t / nil
```

Description

Purges a constraint cellview containing parasitic estimates from memory. The constraint view may contain constraints as well as parasitic estimates and filters. Using this function causes loss of unsaved modifications to constraints as well as to parasitic objects.

Arguments

d_cache	The parasitic cache in which the model or filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.

Value Returned

t	The constraint view was purged.
nil	The operation was unsuccessful.

Examples

```
cache = parCacheFind( "amsPLL" "vco" "schematic" )
=> ci:0x127cfba0
parCachePurge( cache )
=> t
```

Related Topics

Parasitic Aware Design Functions

parCacheSave

```
parCacheSave(
     d_cache
     )
     => t / nil
```

Description

Saves a constraint cellview containing parasitic estimates as well as the constraints and parasitic objects. The constraint view may contain constraints as well as parasitic estimates and filters.

Arguments

d_cache	The parasitic cache in which the model or filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.

Value Returned

t	The constraint view was saved.
nil	The operation was unsuccessful.

Examples

```
cache = parCacheFind( "amsPLL" "vco" "schematic" )
=> ci:0x127cfba0
parCacheSave( cache )
=> t
```

Related Topics

Parasitic Aware Design Functions

parDelete

```
parDelete(
     d_parasitic_id
     )
     => t / nil
```

Description

Deletes a parasitic model or filter. After deleting the object, the *d_parasitic_id* becomes invalid.

Using the *d_parasitic_id* after the original object has been deleted can cause fatal errors.

Arguments

d_parasitic_id	The id of the parasitic model or filter to be deleted.

Value Returned

t	The parasitic object is deleted.
nil	The parasitic object is not deleted.

Examples

```
parDelete( filter )
=> t
parDelete( model )
=> t
```

Related Topics

Parasitic Aware Design Functions

parFilterCreate

```
parFilterCreate(
     d_cache
     [ ?type S_type ]
     [ ?subtype S_subtype ]
     [ ?members l_member_list ]
     [ ?include S_include ]
     [ ?threshold f_threshold ]
     [ ?name t_name ]
     [ ?note t_note ]
     [ ?verbose g_verbose ]
     )
     => filter_id / nil
```

Description

Creates parasitic filters that refer to a given design object as a member.

Arguments

d_cache	The parasitic cache in which the filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
?type *S_type*	A valid filter type that can be R, C, L, or K. The filter only applies to parasitics of the corresponding type.
?subtype *S_subtype*	When *S_type* is set to C, indicates whether coupled or decoupled capacitance should be filtered. Valid values are coupled, decoupled, or both. This argument is ignored for other filter types.
?members *l_member_list*	An ordered filter member list. A *member_list* is a list of members where each member is a list in the form of (design_object_name design_object_type [parameter_list]).
?include *S_include*	Parasitics to be included. Valid values are all, none, or threshold. When set to threshold, only parasitics with a component value greater than *f_threshold* are included.
?threshold *f_threshold*	A floating point value that is ignored unless *S_include* is set to threshold.
?name *t_name*	A string that uniquely identifies the filter in the cache. If unspecified, a name is generated automatically.
?note *t_note*	A string note to be attached to the filter.
?verbose *g_verbose*	A boolean argument that controls whether a message is displayed to inform of the successful creation of a filter. The default is t.

Value Returned

filter_id	The id of the new filter.
nil	The filter was not created.

Examples

The following example creates the following three filters when refining an extracted view.

- filterC - Removes all parasitic capacitance between ibias and gnd!.
- filterCC - Includes all parasitic coupled capacitance between ibias and all other nets (excluding supply nets).
- filterR - Removes all parasitic resistors below 1 ohm on the gnd! net.

```
filterC = parFilterCreate( cache ?type "C" ?members list( list( "gnd!" 'net
) list( "ibias" 'net ) ) ?include "none" )
=> ci:0x129033a0

filterCC = parFilterCreate( cache ?type "C" ?subtype "coupled" ?members
list( list( "ibias" 'net ) ) ?include "all" )
=> ci:0x12e3d7f8

filterR = parFilterCreate( cache ?type "R" ?members list( list( "gnd!" 'net
) ) ?include "threshold" ?threshold 1.0 )
=> ci:0x12f9fac8
```

Related Topics

Parasitic Aware Design Functions

parFind

```
parFind(
     d_cache
     t_name
     )
     => parasitic_id / nil
```

Description

Finds a parasitic model or filter in a given cache.

Arguments

d_cache	The parasitic cache in which the model or filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
t_name	The name of the parasitic model or filter to be found.

Value Returned

parasitic_id	The `id` of the object found.
`nil`	No model or filter was found.

Examples

```
model = parFind( cache "Constr_5" )
=> ci:0x12f82f30

model->type
=> netC
```

Related Topics

Parasitic Aware Design Functions

parModelCreateCustom

```
parModelCreateCustom(
     d_cache
     [ ?net t_net ]
     [ ?type t_type ]
     [ ?simParams l_simParams ]
     [ ?simSweeps l_simSweeps ]
     [ ?parLib t_parLib ]
     [ ?parCell t_parCell ]
     [ ?parView t_parView ]
     [ ?terminalMap l_terminalMap ]
     [ ?name t_name ]
     [ ?note t_note ]
     [ ?verbose g_verbose ]
     )
     => model_id / nil
```

Description

Creates a new customization parasitic estimate model for selected nets. This model is defined in the cellview defined by the `parLib`, `parCell`, and `parView` arguments.

Arguments

d_cache	The parasitic cache in which the model belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
?net *t_net*	Name of the net for which the resistance model is created.
?type *t_type*	Type of estimate that can be a star model or stitched extracted net.
?simParams *l_simParams*	List of simulation parameters containing a name value pair for the parameter.
?simSweeps *l_simSweeps*	List of simulation sweeps containing a name value pair for the parameter.
?parLib *t_parLib*	Library where the extracted view is located.
?parCell *t_parCell*	Cell where the extracted view is located.
?parView *t_parView*	Name of the extracted view.
?terminalMap *l_terminalMap*	List of instance terminals connecting non-hierarchical instances to the net. This list is calculated automatically.
?name *t_name*	A string that uniquely identifies the model in the cache. If unspecified, a name is generated automatically.
?note *t_note*	A string note to be attached to the model.
?verbose *g_verbose*	A boolean argument that controls whether a message is displayed to inform of the successful creation of a model. The default is `t`.

Value Returned

model_id	The `id` of the new estimate model.
`nil`	The model is not created.

Examples

The following example describes how to create a parasitic resistance model for net `vdd!`.

```
cache = parCacheGet( "analogLib" "presister" "symbol" )
parModelCreateCustom( cache ?nets "vdd!" ?parLib "analogLib" ?parCell "presister"
?parView "symbol" )
=> ci:0x12be3998
```

Related Topics

Parasitic Aware Design Functions

parModelCreateNetC

```
parModelCreateNetC(
     d_cache
     [ ?net t_net ]
     [ ?type t_type ]
     [ ?extLib t_extLib ]
     [ ?extCell t_extCell ]
     [ ?extView t_extView ]
     [ ?netMap t_netMap ]
     [ ?include t_include ]
     [ ?threshold t_threshold ]
     [ ?members l_members ]
     [ ?simParams l_simParams ]
     [ ?simSweeps l_simSweeps ]
     [ ?name t_name ]
     [ ?note t_note ]
     [ ?verbose g_verbose ]
     )
     => model_id / nil
```

Description

Creates a new parasitic capacitance estimate between two nets.

Arguments

d_cache	The parasitic cache in which the model belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
?net *t_net*	Name of the net for which capacitance model must be created.
?type *t_type*	Type of estimate that can be a star model or stitched extracted net.
?extLib *t_extLib*	Library where extracted view is located.
?extCell *t_extCell*	Cell where extracted view is located.
?extView *t_extView*	Name of the extracted view.
?netMap *t_netMap*	Indicates the net from the extracted view to be used for the parasitic estimate. It is a paired value that specifies the mapping of a net member to a net.
?include *t_include*	Parasitics to be included. Possible values are `all`, `none`, `threshold`, or `lump`.
?threshold *t_threshold*	If `include` = `threshold`, parasitics with values below the threshold will not be stitched into the estimate view.
?members *l_members*	List of two net members. A *member_list* is a list of members where each member is a list in the form of `(design_object_name design_object_type [ parameter_list ])`.
?simParams *l_simParams*	List of simulation parameters specifying a name-value pair for the parameter.
?simSweeps *l_simSweeps*	List of simulation sweeps specifying a name-value pair for the parameter.
?name *t_name*	A string that uniquely identifies the model in the cache. If unspecified, a name is generated automatically.
?note *t_note*	A string note to be attached to the model.
?verbose *g_verbose*	A boolean argument that controls whether a message is displayed to inform of the successful creation of a model. The default is `t`.

Value Returned

model_id	The `id` of the new estimate model.
`nil`	The model is not created.

Examples

The following example describes how to create a `10f` parasitic capacitance estimate between nets `ibias` and `gnd!`.

```
parModelCreateNetC( cache ?members list( list( "gnd!" 'net ) list( "ibias"
'net ) ) ?simParams list( "c" "10f" ) )
=> ci:0x12fde108
```

Related Topics

Parasitic Aware Design Functions

parModelCreateNetK

```
parModelCreateNetK(
     d_cache
     [ ?members t_members ]
     [ ?simParams l_simParams ]
     [ ?simSweeps l_simSweeps ]
     [ ?name t_name ]
     [ ?note t_note ]
     [ ?verbose g_verbose ]
     )
     => model_id / nil
```

Description

Creates new parasitic mutual-inductance estimate models between the inductance of the specified instance terminals. The members specified are the instance terminals whose estimate inductance is to be considered for creating mutual inductance. If the instance terminals do not have an associated estimate inductance, it is created automatically.

Arguments

d_cache	The parasitic cache in which the model or filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
?members *t_members*	List of the instance terminal members whose estimate inductance is considered for creating mutual inductance. A *member_list* is a list of members where each member is a list in the form of `(design_object_name design_object_type [ parameter_list ])`.
?simParams *l_simParams*	List of simulation parameters containing a name-value pair for the parameter.
?simSweeps *l_simSweeps*	List of simulation sweeps containing a name-value pair for the parameter.
?name *t_name*	A string that uniquely identifies the model in the cache. If unspecified, a name is generated automatically.
?note *t_note*	A string note to be attached to the model.
?verbose *g_verbose*	A boolean argument that controls whether a message is displayed to inform of the successful creation of a model. The default is `t`.

Value Returned

model_id	The `id` of the new estimate model.
`nil`	The model was not created.

Examples

The following example describes how to create a parasitic mutual inductance model between estimate inductances of instance terminals `MP0:D` and `MN0:D`, where the `k` value of mutual

inductance is 1.

```
parModelCreateNetK( cache ?members list("/MP0:D" "/MNO:D") ?simParams
list("k" "1" ))
=> ci:0x12be3998
```

Related Topics

Parasitic Aware Design Functions

parModelCreateNetL

```
parModelCreateNetL(
     d_cache
     [ ?net t_net ]
     [ ?type t_type ]
     [ ?extLib t_extLib ]
     [ ?extCell t_extCell ]
     [ ?extView t_extView ]
     [ ?extNet t_extNet ]
     [ ?terminalMap l_terminalMap ]
     [ ?include t_include ]
     [ ?threshold t_threshold ]
     [ ?members t_members ]
     [ ?simParams l_simParams ]
     [ ?simSweeps l_simSweeps ]
     [ ?name t_name ]
     [ ?note t_note ]
     [ ?verbose g_verbose ]
     )
     => model_id / nil
```

Description

Creates a new parasitic inductance estimate model for a net. The model is star-shaped with an inductance connecting members to a central node. The members are the instance terminals connecting instances to the net. Also included are the terminals of the net. The member list provides an option to specify the list of instance terminals to include. If the members list is `nil`, all instances of the net are selected.

Arguments

d_cache	The parasitic cache in which the model or filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
?net *t_net*	Name of the net to create a resistance model for.
?type *t_type*	Type of estimate, that can be a star model or stitched extracted net.
?extLib *t_extLib*	Library where the extracted view is located.
?extCell *t_extCell*	Cell where the extracted view is located.
?extView *t_extView*	Name of the extracted view.
?extNet *t_extNet*	Name of net in extracted view.
?terminalMap *l_terminalMap*	List of pairs mapping the terminals of the current design to those of the extracted design. Both elements of the colon-separated pair provide the instance and terminal name in the schematic namespace.
?include *t_include*	Parasitics to be included. Possible values are `all`, `none`, `threshold`, or `lump`.
?threshold *t_threshold*	If `include` = `threshold`, parasitics with values below the threshold will not be stitched into the estimate view.
?members *t_members*	

List of the instance terminal members of the net from which the inductance model is built. If the list is `nil`, all instance terminals of the net are considered.

A *`member_list`* is a list of members where each member is a list in the form of `(design_object_name design_object_type [ parameter_list ])`.

`?simParams` *`l_simParams`*

List of simulation parameters containing a name value pair for the parameter.

`?simSweeps` *`l_simSweeps`*

List of simulation sweeps containing a name value pair for the parameter.

`?name` *`t_name`* A string that uniquely identifies the model in the cache. If not specified, a name will be generated automatically.

`?note` *`t_note`* A string note to be attached to the model.

`?verbose` *`g_verbose`*

A boolean argument that controls whether a message is displayed to inform of the successful creation of a model.

The default is `t`.

Value Returned

`model_id`	The `id` of the new estimate model.
`nil`	The model is not created.

Examples

The following example describes how to create a parasitic inductance model for net `vdd!`, where each inductance has a value of `5 henry`.

```
parModelCreateNetL( cache ?net "vdd!" ?simParams list ( "l" "5") ) )
=> ci:0x12be3998
```

Related Topics

Parasitic Aware Design Functions

parModelCreateNetR

```
parModelCreateNetR(
     d_cache
     [ ?net t_net ]
     [ ?type t_type ]
     [ ?extLib t_extLib ]
     [ ?extCell t_extCell ]
     [ ?extView t_extView ]
     [ ?extNet t_extNet ]
     [ ?terminalMap l_terminalMap ]
     [ ?include t_include ]
     [ ?threshold t_threshold ]
     [ ?members l_members ]
     [ ?simParams l_simParams ]
     [ ?simSweeps l_simSweeps ]
     [ ?name t_name ]
     [ ?note t_note ]
     [ ?verbose g_verbose ]
     )
     => model_id | nil
```

Description

Creates a new parasitic resistance model.

Arguments

d_cache	The parasitic cache in which the model or filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
?net *t_net*	Name of net for which the resistance model is created.
?type *t_type*	Type of estimate that can be a star model or stitched extracted net.
?extLib *t_extLib*	Library where extracted view is located.
?extCell *t_extCell*	Cell where extracted view is located.
?extView *t_extView*	Name of the extracted view.
?extNet *t_extNet*	Name of net in extracted view.
?terminalMap *l_terminalMap*	List of pairs mapping the terminals of the current design to those of the extracted design. Both elements of the colon-separated pair provide the instance and terminal name in the schematic namespace.
?include *t_include*	Parasitics to be included. Possible values are `all`, `none`, `threshold`, or `lump`.
?threshold *t_threshold*	If `include` = `threshold`, parasitics with values below the threshold are stitched into the estimate view.
?members *l_members*	List of instance terminals on the net. A *member_list* is a list of members where each member is a list in the form of `(design_object_name design_object_type [ parameter_list ])`.
?simParams *l_simParams*	

List of simulation parameters containing a name-value pair for the parameter.

`?simSweeps` *`l_simSweeps`*

List of simulation sweeps containing a name-value pair for the parameter.

`?name` *`t_name`* A string that uniquely identifies the model in the cache. If unspecified, a name is generated automatically.

`?note` *`t_note`* A string note to be attached to the model.

`?verbose` *`g_verbose`*

A boolean argument that controls whether a message is displayed to inform of the successful creation of a model. The default is `t`.

Value Returned

`model_id`	The `id` of the new estimate model.
`nil`	The model was not created.

Examples

The following example describes how to create a parasitic resistance model for net `vdd!`, where each resistor will be `5 ohms`.

```
parModelCreateNetR( cache ?net "vdd!" ?simParams list( "r" "5" ) )
=> ci:0x12be3998
```

Related Topics

Parasitic Aware Design Functions

parModelListSimParams

```
parModelListSimParams(
     d_model_id
     )
     => sim_param_list / nil
```

Description

Lists the simulation parameters associated with a parasitic estimate. These are the parameters that are set on the parasitic model that is inserted into the netlist.

Arguments

d_model_id	The parasitic model whose simulation parameters must be listed.

Value Returned

sim_param_list	List of simulation parameters specifying a name-value pair for the parameter.
nil	No simulation parameters were set on the estimate.

Examples

```
parModelListSimParams( model )
=> ("r" "5")
```

Related Topics

Parasitic Aware Design Functions

parModelListSimSweeps

```
parModelListSimSweeps(
     d_model_id
     )
     => sim_sweep_list / nil
```

Description

Lists the simulation sweeps associated with a parasitic estimate.

Arguments

d_model_id	The parasitic model whose simulation sweeps you want to list.

Value Returned

sim_sweep_list	List of simulation sweeps specifying a name-value pair for the parameter.
nil	No simulation sweeps have been set on the estimate.

Examples

```
parModelListSimSweeps( model )
=> ("r" "1:2:5")
```

Related Topics

Parasitic Aware Design Functions

parModelUpdateSimParams

```
parModelUpdateSimParams(
     d_model_id
     l_sim_param_list
     )
     => t / nil
```

Description

Updates the values of the listed simulation parameters.

Arguments

d_model_id	The parasitic model whose simulation parameters must be updated.
l_sim_param_list	List of simulation parameters to be updated specifying a name-value pair for the parameter.

Value Returned

t	Parameter values were updated.
nil	Parameter values were not updated.

Examples

The following example describes how to set the component values for a parasitic resistance model to `1 ohm`.

```
parModelUpdateSimParams( model list( "r" "1" ) )
=> t
```

Related Topics

Parasitic Aware Design Functions

parModelUpdateSimSweeps

```
parModelUpdateSimSweeps(
     d_model_id
     l_sim_sweep_list
     )
     => t / nil
```

Description

Updates the sweeps of the listed simulation parameters.

Arguments

d_model_id	The parasitic model whose simulation sweeps you want to update.
l_sim_sweep_list	The list of sweeps to be updated. This should be a list that alternates between name and sweep value.

Value Returned

t	The sweeps were updated.
nil	The sweeps were not updated.

Examples

The following example describes how to set the components on a parasitic resistance estimate to sweep from 1 to 5 ohms in steps of 2 ohms.

```
parModelUpdateSimSweeps( model list( "r" "1:2:5" ) )
=> t
```

Related Topics

Parasitic Aware Design Functions

parObjectListFilters

```
parObjectListFilters(
     d_cache
     t_design_object_name
     t_design_object_type
     )
     => filter_id_list / nil
```

Description

Lists all parasitic filters that refer to a given design object as a member.

Arguments

d_cache	The parasitic cache in which the model or filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
t_design_object_name	A valid design object name in the CDBA name space.
t_design_object_type	A symbol that describes the database object type of a design object. Possible values are 'inst, 'instTerm, 'master, or 'net.

Value Returned

filter_id_list	An id list for all filters that refer to the named design object as a member.
nil	No filters were found.

Examples

```
filters = parObjectListFilters( cache "gnd!" 'net )
=> (ci:0x129033a0 ci:0x12f9fac8)
filters~>parameters
=> ((("type" enum "C")
     ("subtype" enum "both")
     ("include" enum "none")
     ("threshold" float 0.0)
        )
        (("type" enum "R")
```

```
    ("subtype" enum "both")
    ("include" enum "threshold")
    ("threshold" float 0.0)
       )
)
```

Related Topics

Parasitic Aware Design Functions

parObjectListModels

```
parObjectListModels(
     d_cache
     t_design_object_name
     t_design_object_type
     )
     => model_id_list / nil
```

Description

Lists all parasitic models that refer to a given design object as a member.

Arguments

d_cache	The parasitic cache in which the model or filter belongs. It includes a *cache_id*, which is returned by parCacheFind or parCacheGet for a SKILL list that contains library, cell, and view names.
t_design_object_name	A legal design object name in the CDBA name space.
t_design_object_type	A symbol that describes the database object type of a design object. Possible values are 'inst, 'instTerm, 'master, or 'net.

Value Returned

model_id_list	An id list for all models that refers to the named design object as a member.
nil	No models were found.

Examples

```
models = parObjectListModels( cache "gnd!" 'net )
=> (ci:0x131973e0 ci:0x131a4e28 ci:0x131b7528 ci:0x12f82f30)

models~>type
=> (netC netC netC netC)
```

Related Topics

Parasitic Aware Design Functions

parRemoveMembers

```
parRemoveMembers(
     d_parasitic_id
     l_member_index_list
     )
     => t / nil
```

Description

Removes members from a parasitic model or filter.

Arguments

d_parasitic_id	The parasitic model or filter from which the members must be removed.
l_member_index_list	List of integer indexes indicating the members to be removed. For example, `(2 4)` for 2nd and 4th member.

Value Returned

`t`	Members were removed from the parasitic model or filter.
`nil`	Members were not removed.

Examples

The following example describes how to remove two instance terminals from a parasitic resistance estimate. As a result, no resistors are inserted for these members and they are connected directly to the center of the star network.

```
parRemoveMembers( model list( list( "D2:MINUS" 'instTerm ) list( "D3:MINUS"
'instTerm ) ) )
=> t
```

Related Topics

Parasitic Aware Design Functions

parResetAllParams

```
parResetAllParams(
     d_parasitic_id
     )
     => t / nil
```

Description

Resets all model or filter parameters to default values.

Arguments

d_parasitic_id	The `id` of the parasitic model or filter whose parameter values must be reset.

Value Returned

`t`	Parameter values were successfully reset to default values.
`nil`	Parameter values were not reset.

Examples

```
parResetAllParams( filter )
=> t
```

Related Topics

Parasitic Aware Design Functions

parResetParams

```
parResetParams(
     d_parasitic_id
     l_parameter_name_list

     )
     => t / nil
```

Description

Resets the specified model or filter parameters to their default values.

Arguments

d_parasitic_id	The `id` of the parasitic model or filter whose parameter values you want to reset.
l_parameter_name_list	List of parameter names to be reset to their default values.

Value Returned

`t`	Parameter values are successfully reset to default values.
`nil`	Parameter values were not reset.

Examples

```
parResetParams( filterR list( "include" "threshold" ) )
=> t
```

Related Topics

Parasitic Aware Design Functions

parSetNote

```
parSetNote(
     d_parasitic_id
     t_note_string
     )
     => t / nil
```

Description

Replaces the note on a parasitic model or filter.

Arguments

d_parasitic_id	The parasitic model or filter id for which the note must be replaced.
t_note_string	The string that represents the new note.

Value Returned

t	The note was replaced.
nil	The note was not replaced.

Examples

```
parSetNote( filter "Filter out all C between gnd! and ibias" )
=> t
```

Related Topics

Parasitic Aware Design Functions

parUpdateMembers

```
parUpdateMembers(
     d_parasitic_id
     l_member_list
     )
     => t / nil
```

Description

Updates parasitic model or filter members and their parameters. Existing members have their parameter list updated and new members are added at the end.

This function does not reorder existing members. Reordering should be done in conjunction with `parRemoveMembers`.

Arguments

d_parasitic_id	The parasitic model or filter id whose members and parameters must be updated.
l_member_list	The list of members to be updated. A *member_list* is a list of members where each member is a list in the form of `(design_object_name design_object_type [ parameter_list ])`.

Value Returned

`t`	Members and parameters are successfully updated.
`nil`	Members and parameters were not updated.

Examples

The following example describes how to update a parasitic capacitance filter so that it applies to all nets contained under a hierarchical instance.

```
parUpdateMembers( filterC list( list( "I15" 'inst ) ) )
=> t
```

Related Topics

Parasitic Aware Design Functions

parUpdateParams

```
parUpdateParams(
     d_parasitic_id
     l_parameter_list
     )
     => t / nil
```

Description

Updates the parameter values of the listed parameters. Default values will reset the parameter to default and the storage for the default value will be deleted. Enumerated values will be reset first, then updated rather than appended.

Arguments

d_parasitic_id	The `id` of the parasitic model or filter whose parameter values must be updated.
l_parameter_list	The list of parameters to be updated. A *parameter_list* is a list of parameters where each parameter is a list in the form of `(name_string [parameter_type] parameter_value)`. Valid values for the optional `parameter_type` are `'int`, `'float`, `'string`, `'intrange`, `'floatrange`, `'enum`, `'enumset` and `'stringset`.

Value Returned

`t`	Parameter values are updated.
`nil`	Parameter values were not updated.

Examples

The following example describes how to update a parasitic resistance filter to exclude all resistors with a value less than `1.0 ohm`.

```
parUpdateParams( filterR list( list( "include" "threshold" ) list("threshold" 1.0
) ) )
=> t
```

Related Topics

Parasitic Aware Design Functions

sumOPParamV2

```
sumOPParamV2(
     instName
     simParam
     [ labelParam ]
     [ resName ]
     )
     => integer / nil
```

Description

Returns a number which is the result of adding the values of the parameters specified by *instName*.

This method is not provided, therefore, you will have to introduce it by copying and pasting the code. This function does effectively the same as `aelSumOPParam`, but differs in the following:

- `sumOPParamV2` requires that the label parameter name is different than the processed simulator name. For example, it does not allow you to have a label parameter name `id` processing the simulator parameter `id`.
- Using this method is approximately 50% faster than using `aelSumOPParam`.

You should not use `aelSumOPParam` ***and*** `sumOPParamV2` ***at the same time, in the same library or cell.***

The know more about the function definition, see sumOPParamV2.

Arguments

instName	A string that can be a schematic instance, the result of the method `inst()`, or an extracted instance name.
simParam	Any simulation parameter. For example, `id`.
labelParam	A string that is the label parameter defined by `opParamList`. For example, `mFactorF`.

resName	A string used to select the type of results from a particular analysis, for example `dcOpInfo-info`. The type of results available can be obtained using the following command. `results(?noAlias t)` As a default, this input is set to the current type of results.

Value Returned

integer	A number which is the result of adding all of the *simParam* available in the specified *instName*.
`nil`	The instance has failed to map.

Related Topics

Parasitic Aware Design Functions

Specifying Parameters to be Displayed

aelSumOPParam