allocateInstance(

u\_class

)

=> g\_instance

Creates and returns an empty instance of the specified class. All slots of the new instance are unbound.

amsGetInstanceName(

A\_formatterId

A\_instanceId

[x\_iteration]

)

=> t\_instanceName/nil

Helper function that returns the print name of the instance specified by A\_instanceId or the print name of the specified iteration of the instance.

amsPrintInstance(

A\_formatterId

A\_cellViewId

A\_instanceId

)

=> t/nil

Default netlisting procedure for printing an instance, including the instance master name, the instance parameters override list, the attributes for the instance, the name of the instance, and the instance port list.

amsPrintInstanceMasterName(

A\_formatterId A\_cellViewId

A\_instanceId

)

=> t/nil

Default netlisting procedure for printing the name of an instance master. By overriding this function, you can modify the name of the instance master.

amsPrintInstanceParameter(

A\_formatterId

t\_instanceName

A\_parameterId

)

=> t/nil

Helper function that prints the instance parameter specified by A\_parameterId.

amsPrintInstanceParameters(

A\_formatterId

A\_cellViewId

A\_instanceId

)

=> t/nil

Default netlisting procedure for printing instance parameters.

amsPrintInstancePorts(

A\_formatterId

A\_instanceId

[x\_iteration]

)

=> t/nil

Default netlisting procedure for printing the port list of an instance or of a particular iteration of an instance. The ports in the port list are arranged according to the value specified by the CDF termOrder property, if the termOrder property exists for the instance. Otherwise, the order of the ports is undetermined. The amsPrintInstancePorts function prints the port list by order or by named port maps, according to the effective options and settings.

asiMapInstanceName( t\_dataDir l\_specifier )

=> l\_specifier

Maps the hierarchical schematic instance name to the name in the netlist.

ciMatchedParamsForInstanceSymmetry(

u\_cache

l\_instances

)

=> nil

Used by the Circuit Prospector assistant as a generator to set Matched Parameters constraints to the selected pair of instance members.

ciMatchedParamsForSameSizeInstances(

u\_cache

l\_instances

)

=> nil

Used by the Circuit Prospector assistant as a generator to set Matched Parameters constraints to the selected instance members.

cpfDefinePowerSwitchInstance(

t\_cellName

t\_instanceName

?type t\_type

?stage1Enable

[?stage2Enable]

?power

?powerSwitchable

?ground

?groundSwitchable

)

=> l\_switchAttributes | nil

Registers a transistor or native instance as a power switch.

cpfGetInstancePowerGroundNetPair()

=> l\_list | nil

Returns a list of lists containing names of instance, power net and ground net registered using the cpfRegInstancePowerGroundNetPair function.

cpfGetPowerSwitchInstanceInfo(

cellName

instName

)

=> l\_switchAttributes | nil

Gets the information about a registered switched instance.

cpfIsPowerSwitchInstance(

cellName

instName

)

=> t | nil

Checks whether an instance has been registered as a switched instance.

cpfRegInstancePowerGroundNetPair(

t\_instancename

t\_powernet

t\_groundnet

)

=> nil | t

Registers an instance with a power net and ground net pair to ensure that the registered instance are added to a power domain that has the given pair of power net and ground net.

cpfRemovePowerSwitchInstance(

cellName

instName

)

=> t | nil

Clears an existing power switch registration.

cphVisitedInstance(

g\_physConfigID

)

=> d\_instID | nil

Returns the database ID of the instance of the occurrence currently being visited.

dbGetInstanceByName(

d\_cellView

t\_name

)

=> d\_inst | nil

Retrieves an instance or mosaic.

dbGetInstanceByName is maintained only for backward compatibility; you should now use dbFindAnyInstByName.

dbHasInstance(

d\_cellViewId

)

=> t | nil

Checks whether the specified cellview contains any instances.

geGetSelectedInstancesCount(

[ d\_cellviewId ]

)

=> n\_count

Counts selected instances in a cellview.

hnlGetInstanceCount()

Returns the number of instances in the design most recently netlisted in the same session. This does not include instances with the nlAction=ignore property that are ignored during netlisting.

initializeInstance(

g\_instance

[ u\_?initArg1 value1 ]

[ u\_?initArg2 value2 ] ...

)

=> t

Initializes the newly created instance of a class. initializeInstance is called by the makeInstance function.

leHiRemasterInstances(

[ w\_windowId ]

)

=> t | nil

Opens the Remaster Instances form, in which you specify instances to search for based on the view name, cell name, and the library name for a given cellviewID, then re-masters those instances with the master that matches the update library, update cell and update view names you provide in the form. If you do not specify w\_windowId, the layout editor uses the current window.

leRemasterInstances(

d\_cellViewId

t\_searchLibrary

t\_searchCellview

t\_searchViewType

t\_updateLibrary

t\_updateCellview

t\_updateViewType

[ g\_checkTerminals ]

)

=> t | nil

Searches for instances based on the view name, cell name, and the library name for a given cellviewID, then re-masters those instances with the master that matches the update library, update cell and update view names you provide. The \* wildcard can be used in any search string to replace any number of characters prior to or following the asterisk. If any of the search strings are empty then all matches in the cds.lib are considered, if matches exist. If any of the update strings are empty the update string uses the equivalent search string value. If both the search and the update strings are empty, only exact matches of the respective strings are considered. Due to the extensive changes available when using this command, making a backup copy of your design to guard against errors when replacing data is recommended.

makeInstance(

us\_class

[ u\_?initArg1 value1 ]

[ u\_?initArg2 value2 ] ...

)

=> g\_instance

Creates an instance of a class, which can be given as a symbol or a class object.

ocnGetInstancesModelName( [l\_instance] )

=> l\_instance/nil

This function returns the model name used by the instance in opened simulation results.

odcRegInstance(

t\_deAppName

t\_functionText

)

=> t | nil

Allows you to define a custom SKILL function to specify your own information to be displayed in the Info Balloon box for instances. The application must be specified and currently only Layout-XL is supported.

pcUserGenerateInstance(

d\_inst

t\_masterTag

p\_port

)

=> t | nil

A user-defined procedure called by the compiler before it processes any instances in a master pcell. The procedure is normally used to suppress instance generation or to modify instances.

pcUserGenerateInstancesOfMaster(

d\_masterCV

l\_instanceList

t\_tag

p\_port

)

=> t | nil

A user-defined procedure called by the compiler for every master pcell in a master pcell. The compiler calls the procedure before it generates code for instances (but not arrays) for the master. The procedure is normally used to generate code to switch masters.

schHiRenumberInstances(

)

=> t

Opens the Renumber Instances form which can be used to renumber instances automatically.

schRenumberInstances(

g\_objId

[ t\_scope ]

[ g\_verbose ]

[ t\_sequence ]

[ x\_startIndex ]

[ t\_applyTo [ t\_libraryName t\_cellName t\_viewName ] ]

)

=> t | nil

Resequences instances using the format instNamePrefix number that results in unique numbering indexes for each component name prefix encountered. Any voids in a numbering sequence are resolved by renaming instances with the highest numbers to fill the voids.

updateInstanceForDifferentClass(

g\_previousObj

g\_currentObj

@rest initargs

)

=> t

A generic function, which is called from changeClass to update the specified instance (g\_currentObj).

updateInstanceForRedefinedClass(

obj

l\_addedSlots

l\_deletedSlots

l\_dplList

)

=> t

It is a generic function, which is called to update all instances of a class, when a class redefinition occurs.

vfoChopInstance(

g\_obj

instId

chopShapeData

@rest l\_args

)

=> t | nil

This function chops the fluid shape of a given instance. The information of the chop rectangle or polygon is provided by chopShapeData which is a class object of vfoShapeData. The shape data points are in the design coordinates. This function transforms the chopShapeData before chopping the fluid shape.

Note: The Chop command for fluid Pcells does not pass these arguments. This function updates the relevant Pcell parameters. If the chop operation results in multiple shapes, it may need to create multiple Pcell instances.

vfoMergeInstances(

g\_obj

inst1

inst2

@rest l\_args

)

=> dbId | nil

Merges the fluid Pcell instances inst1 and inst2.

Note: The Merge command for fluid Pcells does not pass these args. This function updates the relevant Pcell parameters.