Digital Design Netlisting and Simulation SKILL Reference

Product Version IC23.1
June 2023

Contents

1
Verilog Netlister Functions . . . 10
- Licensing Requirements . . . 11
- vicOpenVlogCallBack . . . 12
- vlVicCrossSelectionForm . . . 13
- vlVicPSForm . . . 14
- vlogifCrossSelectionCB . . . 15

2
OSS Functions . . . 16
- Licensing Requirements . . . 21
- cat . . . 23
- cdsGetNetlistMode . . . 24
- cdsSetNetlistMode . . . 25
- ERC . . . 26
- fnlAbortNetlist . . . 27
- fnlCurrentCell . . . 28
- fnlCurrentCellCdsName . . . 29
- fnlCurrentInst . . . 30
- fnlCurrentInstCdsName . . . 31
- fnlCurrentIteration . . . 32
- fnlCurrentModelExtName . . . 33
- fnlCurrentSig . . . 34
- fnlCurrentSigPathName . . . 35
- fnlGetGlobalSigNames . . . 36
- fnlInstCdsNameExtName . . . 37
- fnlPathList . . . 38
- fnlPrint . . . 39
- fnlSearchPropString . . . 40
- fnlSigCdsNameExtName . . . 41
- fnlTermCdsNameExtName . . . 42

fnlTermExtName 43
fnlTopCell 44
hnlAbortNetlist 45
hnlAddExtraParameters 46
hnlCatIncrementalNetlistFiles 47
hnlCellExtracted 48
hnlCellInAllCells 49
hnlCloseCellFiles 50
hnlCloseMasterList 51
hnlCompletePrint 52
hnlDeRegPostNetlistTrigger 53
hnlDeRegPreNetlistTrigger 54
hnlDoInstBased 55
hnlDoNetBased 57
hnlEMHGetDigitaGlobalNets 58
hnlEMHGetDigitalNetlistFileName 59
hnlEMHSetVerbosityLevel 60
hnlFindAllCells 61
hnlFindAllInstInCell 63
hnlGenIncludeFile 64
hnlGetCellHdbProps 66
hnlGetGlobalModelMappedName 67
hnlGetGlobalNetMappedName 68
hnlGetInstanceCount 69
hnlGetMappedInstNames 70
hnlGetMappedModelNames 71
hnlGetMappedNames 72
hnlGetMappedNetNames 73
hnlGetMasterCells 74
hnlGetPrintLinePrefix 75
hnlGetPropVal 76
hnlGetRoundProp 77
hnlGetScaleCapacitance 78
hnlGetScaleMarginalDelay 79
hnlGetScaleTimeUnit 80
hnlGetSimulator 81

hnlGetSourceFile . . . 82
hnlGetSourceFileModels . . . 83
hnlGetSymbolPropVal . . . 84
hnlGetTermByName . . . 85
hnlGetTermNameOfSig . . . 86
hnlIfNoProcedure . . . 87
hnlIgnoreTerm . . . 88
hnlInitMap . . . 89
hnlInitPrint . . . 98
hnlIsAPatchCord . . . 100
hnlIsAStoppingCell . . . 101
hnlIsCellNetlistable . . . 102
hnlIsCurrentInstStopping . . . 103
hnlIsCVInUserStopCVList . . . 104
hnlMakeNetlistFileName . . . 105
hnlMapCellModuleName . . . 107
hnlMapCellName . . . 110
hnlMapInstName . . . 112
hnlMapModelName . . . 113
hnlMapName . . . 115
hnlMapNetName . . . 117
hnlMapTermName . . . 118
hnlMultipleCells . . . 119
hnlNameOfSignal . . . 120
hnlNetNameOnTerm . . . 121
hnlNetNameOnTermName . . . 122
hnlNmpSetNameSpaces . . . 123
hnlOpenTopCell . . . 124
hnlPcellIsParamOverridden . . . 125
hnlPostNetlistTriggerList . . . 126
hnlPreNetlistTriggerList . . . 127
hnlPrintDevices . . . 128
hnlPrintMessage . . . 130
hnlPrintNetlist . . . 131
hnlPrintSignal . . . 133
hnlPrintString . . . 134

hnlRegPostNetlistTrigger . . . 136
hnlRegPreNetlistTrigger . . . 138
hnlRunNetlister . . . 140
hnlScaleCapacitance . . . 141
hnlScaleMarginalDelay . . . 142
hnlScaleTimeUnit . . . 143
hnlSetCellFiles . . . 144
hnlSetDef . . . 145
hnlSetMappingType . . . 146
hnlSetPrintLinePrefix . . . 147
hnlSetPseudoTermDir . . . 148
hnlSetVars . . . 150
hnlSortTerms . . . 151
hnlSortTermsToNets . . . 152
hnlStartNetlist . . . 153
hnlStopNetlist . . . 154
hnlStringToList . . . 155
hnlWriteBlockControlFile . . . 156
hnlWriteMap . . . 158
iseCloseSchWindow . . . 159
iseCloseSimWindow . . . 160
iseCommToSimulator . . . 161
iseCompleteInteractive . . . 162
iseEnterNodeNamesList . . . 163
iseExitSimulator . . . 164
iseGetExtName . . . 165
iseGetInputFromEncapWindow . . . 166
iseGetMappedProbeList . . . 167
iseGetProbeList . . . 168
iseInitSchematicWindow . . . 169
iseInitSimWindow . . . 170
iseInterruptSimulator . . . 171
iseNetExtNameCdsName . . . 172
iseOpenWindows . . . 173
isePrintName . . . 174
isePrintNameCB . . . 175

isePrintSimulatorCommand 176
iseReleaseNodeFrom 177
iseSearchForASchWindow 178
iseSendOutputToEncapHistory 179
iseSetEncapBindKeys 180
iseSetNodeTo 181
iseSimulate 182
iseStartInteractive 183
iseStartSimulator 185
iseUpdateNetlist 187
iseUpdateStimulus 188
netlist 189
runsim 191
sim 193
simAddProbeCapByName 195
simAddProbeCapByScreen 196
simAddProbeCapForBusBit 197
simCheckExist 198
simCheckHeader 199
simCheckVariables 200
simCheckViewConfig 201
simCleanRun 202
simDateStamp 203
simDeleteRunDirFile 204
simDeRegPostNetlistTrigger 205
simDeRegPreNetlistTrigger 206
simDesignVarCdsNameExtName 207
simDesignVarExtNameCdsName 208
simDrain 209
simEditFileWithName 210
simExecute 211
simFindFile 212
simFlattenWithArgs 213
simGetLoginName 217
simGetTermList 218
simIfNoProcedure 219

simIlSleep 220
simin 221
simInitControl 223
simInitEnv 224
simInitEnvWithArgs 226
simInitRaw 228
simInitRunDir 229
simInitSimulator 231
simInstCdsNameExtName 232
simInstExtNameCdsName 233
simInWithArgs 234
simJobMonitor 237
simLoadNetlisterFiles 238
simLoadSimulatorFiles 239
simNetCdsNameExtName 240
simNetExtNameCdsName 241
simNetlistWithArgs 242
simNoNetlist 246
simout 247
simOutWithArgs 249
simPostNameConvert 251
simPostNetlistTriggerList 252
simPreNameConvert 253
simPreNetlistTriggerList 254
simPrintEnvironment 255
simPrintError 257
simPrintErrorLine 258
simPrintMessage 259
simPrintTermList 260
simReadNetCapFile 261
simRegPostNetlistTrigger 262
simRegPreNetlistTrigger 263
simRunDirInfile 264
simRunDirLoad 265
simRunDirOutfile 266
simRunNetAndSim 267

simRunNetAndSimWithArgs 269
simRunNetAndSimWithCmd 271
simSetDef 273
simSetDefWithNoWarn 274
simStringsToList 275
simSubProbeCapByName 276
simSubProbeCapByScreen 277
simVertToHoriz 278
simViewFileWithArgs 280
simWaveOpen 281

3
VHDL Toolbox Functions 282

Licensing Requirements 282
vhdlHiImport 283
vhdlHiInvokeToolBox 284
vhdlImport 285
vhdlPinListToVHDL 287
vhdlRegisterSimulator 289
vhdlToPinList 291
vhmsCompilationFailure 293
vhmsDefaultEdit 294
vhmsGetCellParameters 295
vhmsPinListToVHDLAMS 296
vhmsSaveFile 297
vhmsSymbolToPinListGen 298
vhmsToPinList 299
vhmsUpdateCellCDFParams 300
vosHiDisplayNetlist 301
vosLaunchIrunSimulation 302

1

Verilog Netlister Functions

This topic describes the SKILL APIs of tools used to netlist and simulate digital designs in the Virtuoso® design environment. It provides information on the SKILL functions that you can use with the following Virtuoso applications:

- Virtuoso Verilog Environment for NC-Verilog Integration
- Virtuoso Verilog Environment for SystemVerilog Integration
- Virtuoso Open Simulation System
- Virtuoso VHDL Toolbox

This topic is aimed at designers of integrated circuits and assumes that you are familiar with:

- The Virtuoso Studio design environment and application infrastructure mechanisms designed to support consistent operations between all Cadence tools.
- The applications used to design and develop integrated circuits in the Virtuoso Studio design environment, notably Virtuoso Layout Suite and Virtuoso Schematic Editor.
- The design and use of parameterized cells.
- Component Description Format (CDF), which lets you create and describe your own components for use with ADE.
- Cadence SKILL™ language.

This topic lists the Cadence® SKILL functions associated with the Verilog netlister for invoking the form and update cellviews.

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.

vicOpenVlogCallBack	vlVicCrossSelectionForm	vlVicPSForm
vlogifCrossSelectionCB		

Licensing Requirements

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

Related Topics

About NC-Verilog Environment

OSS Functions

Verilog Netlister Functions

vicOpenVlogCallBack

```
vicOpenVlogCallBack(
      )
      => t / nil
```

Description

Opens the Verilog Integration form.This function is a CIW menu callback function and is called when the Verilog integration environment is invoked.

Arguments

None

Value Returned

`t`	The Verilog Integration form is opened.
`nil`	The operation was unsuccessful.

Examples

The following example opens the Verilog Integration form.

```
vicOpenVlogCallBack()
=> t
```

Related Topics

Verilog Netlister Functions

vlVicCrossSelectionForm

```
vlVicCrossSelectionForm(
      )
      => t / nil
```

Description

Opens the Cross Selection Setup form. This form is used to cross-select objects of a design between SimVision and Virtuoso Schematic Editor during interactive simulation.

Arguments

None

Value Returned

`t`	The Cross Selection Setup form is displayed.
`nil`	The Cross Selection Setup form is not displayed.

Examples

The following example opens the Cross Selection Setup form.

```
vlVicCrossSelectionForm()
=> t
```

Related Topics

Cross Selection Setup Form

Verilog Netlister Functions

vlVicPSForm

```
vlVicPSForm(
      )
      => t / nil
```

Description

Opens the Post Simulation Analysis form. This form is used to cross-select objects of a design between SimVision and Virtuoso Schematic Editor during post-simulation analysis.

Values in the Post Simulation Analysis form can be set using the following SKILL variables:

- simPSHierPrefix
- simPSSimulationDir
- simPSSimulationFile
- simPSVerilogRunDir

Arguments

None

Value Returned

`t`	The Post Simulation Analysis form is displayed.
`nil`	The operation was unsuccessful.

Examples

The following example opens the Post Simulation Analysis form.

```
vlVicPSForm()
=> t
```

Related Topics

Post Simulation Analysis Form

Verilog Netlister Functions

vlogifCrossSelectionCB

```
vlogifCrossSelectionCB(
     g_enable
     )
     => t / nil
```

Description

Controls the cross-selection between SimVision and Virtuoso Schematic Editor when performing interactive simulation.

Arguments

g_enable	Boolean flag, specifying whether cross-selection should be enabled or disabled. Valid values are `t` and `nil`.

Value Returned

`t`	Cross-selection was enabled.
`nil`	Cross-selection was disabled.

Examples

The following example describes how you can add the function in the `.cdsinit` file with the required argument before launching Virtuoso.

```
hiSetBindKey("Schematics" "<Key>1" "vlogifCrossSelectionCB(t)")
hiSetBindKey("Schematics" "<Key>2" "vlogifCrossSelectionCB(nil)")
```

Now, run the simulation in interactive mode and enable cross-selection.

Additionally, you can also control this cross-selection in the schematic window by doing the following:

- Press the `1` key to enable cross-selection
- Press the `2` key to disable cross-selection

Related Topics

Verilog Netlister Functions

2

OSS Functions

Open Simulation System (OSS) gives you quick access to the simulators that it supports and lets you integrate and customize new simulators into the Virtuoso design environment.

This topic describes the following SKILL functions associated with OSS:

- FNL Functions - The following functions are the flat netlister (FNL) SKILL functions.

fnlAbortNetlist	fnlCurrentCell
fnlCurrentCellCdsName	fnlCurrentInst
fnlCurrentInstCdsName	fnlCurrentIteration
fnlCurrentModelExtName	fnlCurrentSig
fnlCurrentSigPathName	fnlGetGlobalSigNames
fnlInstCdsNameExtName	fnlPathList
fnlPrint	fnlSearchPropString
fnlSigCdsNameExtName	fnlTermCdsNameExtName
fnlTermExtName	fnlTopCell

- SE Functions - The following SKILL functions defined by the simulation environment (SE) let you simplify the integration of your simulator. These functions are in both the Cadence graphics program and the SE program.

cat	cdsGetNetlistMode
cdsSetNetlistMode	ERC
netlist	runsim
sim	simAddProbeCapByName
simAddProbeCapByScreen	simAddProbeCapForBusBit
simCheckExist	simCheckHeader

simCheckVariables	simCheckViewConfig
simDateStamp	simDeleteRunDirFile
simDesignVarCdsNameExtName	simDesignVarExtNameCdsName
simDrain	simExecute
simFindFile	simFlattenWithArgs
simGetLoginName	simGetTermList
simIfNoProcedure	simIlSleep
simin	simInitControl
simInitRaw	simInitRunDir
simInitSimulator	simInstCdsNameExtName
simInstExtNameCdsName	simInWithArgs
simLoadNetlisterFiles	simLoadSimulatorFiles
simNetCdsNameExtName	simNetExtNameCdsName
simNetlistWithArgs	simNoNetlist
simout	simOutWithArgs
simPrintEnvironment	simPrintError
simPrintErrorLine	simPrintMessage
simPrintTermList	simReadNetCapFile
simRunDirInfile	simRunDirLoad
simRunDirOutfile	simSetDef
simSetDefWithNoWarn	simStringsToList
simSubProbeCapByName	simSubProbeCapByScreen
simVertToHoriz	

- SE Graphics Functions - The following functions, which are defined by SE, simplify integrating your simulator into the Cadence graphics environment. These functions represent the functionality available on the Simulation menu. If you want to create your own menus and/or forms for the Simulation user interface, you can use them to perform the required SE functionality.

These functions are defined only in the Cadence graphics environment and cannot be executed in the SI program.

simCleanRun	simEditFileWithName
simInitEnv	simInitEnvWithArgs
simJobMonitor	simPostNameConvert
simPreNameConvert	simRunNetAndSim
simRunNetAndSimWithArgs	simRunNetAndSimWithCmd
simViewFileWithArgs	simWaveOpen

- ISE Functions - The following functions are defined in the Interactive Simulation Environment (ISE).

iseCloseSchWindow	iseCloseSimWindow
iseCommToSimulator	iseCompleteInteractive
iseEnterNodeNamesList	iseExitSimulator
iseGetExtName	iseGetInputFromEncapWindow
iseGetMappedProbeList	iseGetProbeList
iseInitSchematicWindow	iseInitSimWindow
iseInterruptSimulator	iseNetExtNameCdsName
iseOpenWindows	isePrintName
isePrintNameCB	isePrintSimulatorCommand
iseReleaseNodeFrom	iseSearchForASchWindow
iseSendOutputToEncapHistory	iseSetEncapBindKeys
iseSetNodeTo	iseSimulate
iseStartInteractive	iseStartSimulator
iseUpdateNetlist	iseUpdateStimulus

- HNL Access Functions - The following functions include the property, database, and print SKILL functions of the Hierarchical Netlister (HNL).

- Property functions can be used to search for properties and to scale property values.

hnlEMHGetDigitaGlobalNets	hnlEMHGetDigitalNetlistFileName
hnlEMHSetVerbosityLevel	hnlGetCellHdbProps
hnlGetInstanceCount	hnlGetPropVal
hnlGetRoundProp	hnlGetScaleCapacitance
hnlGetScaleMarginalDelay	hnlGetScaleTimeUnit
hnlGetSimulator	hnlGetSourceFile
hnlGetSourceFileModels	hnlGetSymbolPropVal
hnlOpenTopCell	hnlPcellIsParamOverridden
hnlScaleCapacitance	hnlScaleMarginalDelay
hnlScaleTimeUnit	

- Database functions provide information about the design and the internal data structures of HNL specific to netlisting.

hnlAddExtraParameters	hnlCellExtracted
hnlCellInAllCells	hnlIgnoreTerm
hnlIsAPatchCord	hnlIsAStoppingCell
hnlIsCurrentInstStopping	hnlMultipleCells
hnlNameOfSignal	hnlNetNameOnTerm
hnlNetNameOnTermName	

- Print functions control netlist formatting and printing.

hnlCompletePrint	hnlInitPrint
hnlPrintString	

- Miscellaneous Functions - The following functions include the general utility functions in HNL.

hnlAbortNetlist	hnlDoInstBased
hnlFindAllCells	hnlGetPrintLinePrefix

hnlIfNoProcedure	hnlPrintDevices
hnlPrintMessage	hnlPrintNetlist
hnlRunNetlister	hnlSetDef
hnlSetPrintLinePrefix	hnlSetPseudoTermDir
hnlSetVars	hnlSortTerms
hnlSortTermsToNets	hnlStartNetlist
hnlStopNetlist	hnlStringToList

- Name-mapping functions implement the name-mapping feature of HNL. You can use the `hnlMapName`, `hnlMapNetName`, `hnlMapTermName`, `hnlMapInstName`, and `hnlMapModelName` functions during netlisting. The initialization and write functions are called by the HNL driver as part of the netlisting process.

hnlGetMappedInstNames	hnlGetMappedModelNames
hnlGetMappedNames	hnlGetMappedNetNames
hnlInitMap	hnlIsCVInUserStopCVList
hnlMapInstName	hnlMapModelName
hnlMapName	hnlMapNetName
hnlMapTermName	hnlNmpSetNameSpaces
hnlSetMappingType	hnlWriteMap

- Incremental netlisting functions can be used when you write an incremental netlist formatter.

hnlCatIncrementalNetlistFiles	hnlCloseCellFiles
hnlGenIncludeFile	hnlGetGlobalModelMappedName
hnlGetGlobalNetMappedName	hnlMakeNetlistFileName
hnlMapCellModuleName	hnlMapCellName
hnlSetCellFiles	hnlWriteBlockControlFile

- HNL Trigger Functions for Pre- and Post-Netlist Customization - Pre-netlist triggers are called after the `.simrc` file is loaded and before the netlisting starts. These triggers must be defined and registered in the `.simrc` file. Post-netlist triggers are called after netlisting finishes and can be defined and registered in the `libInit.il` file. To maintain consistency, all trigger functions must be defined and registered in the `.simrc` file. Additionally, a trigger must be defined before it is registered or deregistered. These functions let you customize the netlist as per user requirements after the netlisting is done by the tool.

The following HNL functions are available for customizing the netlist as per user requirements after the netlisting is done by the tool.

hnlDeRegPostNetlistTrigger	hnlDeRegPreNetlistTrigger
hnlPostNetlistTriggerList	hnlPreNetlistTriggerList
hnlRegPostNetlistTrigger	hnlRegPreNetlistTrigger
simDeRegPostNetlistTrigger	simDeRegPreNetlistTrigger
simPostNetlistTriggerList	simPreNetlistTriggerList
simRegPostNetlistTrigger	simRegPreNetlistTrigger

- HNL Net-Based Netlisting Functions - The following HNL functions are available for use by the formatter for net-based netlisting.

hnlCloseMasterList	hnlDoNetBased
hnlFindAllInstInCell	hnlGetMasterCells
hnlGetTermByName	hnlGetTermNameOfSig
hnlIsCellNetlistable	hnlPrintSignal

Licensing Requirements

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

Related Topics

Introducing the Open Simulation System (OSS)

Customizing the Simulation Environment (SE)

Customizing the Interactive Simulation Environment (ISE)

Customizing the Hierarchical Netlister (HNL)

Customizing the HNL Net-Based Netlister

Customizing the Flat Netlister (FNL)

Verilog Netlister Functions

VHDL Toolbox Functions

cat

```
cat(
     t_filename
     )
     => t / nil
```

Description

Prints the contents of filename on standard output. The function is defined in `/etc/skill/si/simcap.ile`. You can modify this function.

Arguments

`t_filename`	Name of the file.

Value Returned

`t`	The operation was successful.
`nil`	The operation was unsuccessful.

Examples

```
cat( "si.env" )
```

Related Topics

OSS Functions

cdsGetNetlistMode

```
cdsGetNetlistMode(
     )
     => Analog / Digital / Compatibility
```

Description

Returns the value of the shell environment variable `CDS_Netlisting_Mode` that is set in the current shell to invoke Virtuoso.

The valid values for this shell environment variable are `Digital`, `Analog`, and `Compatibility`. If you set any other value, the function returns the default value, `Digital`.

Arguments

None

Value Returned

`Digital`	The shell environment variable `CDS_Netlisting_Mode` is set to `Digital`.
`Analog`	The shell environment variable `CDS_Netlisting_Mode` is set to `Analog`.
`Compatibility`	The environment variable `CDS_Netlisting_Mode` is set to `Compatibility`.

Examples

```
$ virtuoso -nograph
> setShellEnvVar("CDS_Netlisting_Mode=Analog")
=> t
> cdsSetNetlistMode()
=> t
> cdsGetNetlistMode()
=> "Analog"
```

Related Topics

OSS Functions

cdsSetNetlistMode

```
cdsSetNetlistMode(
     )
     => t / nil
```

Description

Sets netlisting mode, which is based on the current value of the SHELL environment variable `CDS_Netlisting_Mode`. The valid values for this shell environment variable are `Digital`, `Analog`, or `Compatibility`. The function `cdsSetNetlistMode` is used in the Virtuoso SKILL environment to update the settings without restarting Virtuoso.

Arguments

None

Value Returned

`t`	The operation is successful.
`nil`	The operation is unsuccessful.

Examples

The following example sets netlisting mode.

```
$ virtuoso -nograph
> setShellEnvVar("CDS_Netlisting_Mode=Analog")
=> t
> cdsSetNetlistMode()
=> t
> cdsGetNetlistMode()
=> "Analog"
> setShellEnvVar("CDS_Netlisting_Mode=Digital")
=> t
> cdsSetNetlistMode()
=> t
> cdsGetNetlistMode()
=> "Digital"
```

Related Topics

OSS Functions

ERC

```
ERC(
    )
    => t / nil
```

Description

Defines the sequence of steps for performing an Electric Rule Checking (ERC). This function is invoked if you provide no overriding function. It checks the variables, the sequence of steps, such as netlisting (`ercNetlist`), and calls the ERC program for execution (`ercSimout`).

Arguments

None

Value Returned

`t`	The ERC operation was completed.
`nil`	The ERC was done with errors.

Examples

The following example defines the sequence of steps for performing an ERC.

```
ERC()
=> t
```

Related Topics

OSS Functions

fnlAbortNetlist

```
fnlAbortNetlist(
      )
      => nil
```

Description

Aborts netlisting. When the formatter detects an error during netlisting, it calls this function to inform the netlister to abort netlisting.

Arguments

None

Value Returned

`nil`	Always returns `nil`.

Examples

The following example aborts netlisting.

```
fnlAbortNetlist()
=> t
```

Related Topics

OSS Functions

fnlCurrentCell

```
fnlCurrentCell(
     )
     => d_cellviewId / nil
```

Description

Returns the master of the current instance being expanded. This function must be called only during the evaluation of the `NLPcompleteElementString` and `NLPcreateModelString` properties.

Arguments

None

Value Returned

d_cellviewId	An object identifier for the master cell of the current instance.
`nil`	The operation was unsuccessful.

Examples

The following example returns the master of the current instance being expanded.

```
fnlCurrentCell()
=> nil
```

Related Topics

OSS Functions

fnlCurrentCellCdsName

```
fnlCurrentCellCdsName(
     )
     => t_cellName / nil
```

Description

Returns the master cell name of the current instance being expanded. This function corresponds to the netlister-defined `BlockName` property and must be called only during evaluation of the `NLPcompleteElementString` and `NLPcreateModelString` properties.

Arguments

None

Value Returned

t_cellName	The name of the master cell of the current instance.
`nil`	The operation was unsuccessful.

Examples

```
fnlCurrentCellCdsName()
```

Related Topics

OSS Functions

fnlCurrentInst

```
fnlCurrentInst(
      )
      => d_instanceId / nil
```

Description

Returns the current instance being expanded. This function must be called only during expansion of the `NLPcompleteElementString` property. Do not use the master field of the resulting `instanceId`. If you use it, you get the symbol cellview rather than the stopping cellview corresponding to the symbol placed in the schematic. To get the instance master in a format instruction, use the `fnlCurrentCell` function.

Arguments

None

Value Returned

d_instanceId	An object identifier for the current instance.
`nil`	The operation was unsuccessful.

Examples

```
fnlCurrentInst()
```

Related Topics

OSS Functions

fnlCurrentInstCdsName

```
fnlCurrentInstCdsName(
    )
    => t_pathName / nil
```

Description

Returns the full instance pathname to the current instance being expanded. This function must be called only during expansion of the `NLPcompleteElementString` property.

Arguments

None

Value Returned

t_pathName	The path to the current instance.
`nil`	The operation was unsuccessful.

Examples

```
fnlCurrentInstCdsName()
```

Related Topics

OSS Functions

fnlCurrentIteration

```
fnlCurrentIteration(
      )
      => x_index
```

Description

Returns an index of the current iterated instance being expanded. Only an instance can be iterated and placed in the schematic.

Arguments

None

Value Returned

x_index	The index of the current iterated instance.

Examples

```
fnlCurrentIteration()
```

Related Topics

OSS Functions

fnlCurrentModelExtName

```
fnlCurrentModelExtName(
      )
      => t_modelName / nil
```

Description

Returns the netlister-assigned model name of the current instance being expanded. This function corresponds to the netlister-defined `ModelNumber` property and must be called only during evaluation of the `NLPcompleteElementString` and `NLPcreateModelString` properties.

Arguments

None

Value Returned

t_modelName	The netlister-assigned model name of the current instance.
`nil`	The operation failed and encountered an error while performing the task.

Examples

```
fnlCurrentModelExtName()
```

Related Topics

OSS Functions

fnlCurrentSig

```
fnlCurrentSig(
    )
    => d_sigId / nil
```

Description

Returns the current signal being expanded.

Arguments

None

Value Returned

d_sigId	An object identifier for the current signal.
`nil`	The operation failed and encountered an error while performing the task.

Examples

```
fnlCurrentSig()
```

Related Topics

OSS Functions

fnlCurrentSigPathName

```
fnlCurrentSigPathName(
    )
    => t_pathName / nil
```

Description

Returns the full pathname of the current signal being expanded. It corresponds to the netlister-defined `NetPathName` property. As the corresponding property, this function is valid only during evaluation of the `NLPcreateNetString` property.

Arguments

None

Value Returned

t_pathName	Represents the path of the current signal.
`nil`	The operation failed and encountered an error while performing the task.

Examples

```
fnlCurrentSigPathName()
```

Related Topics

OSS Functions

fnlGetGlobalSigNames

```
fnlGetGlobalSigNames(
     )
     => l_sigNames / nil
```

Description

Returns a list of strings that are the names for all of the global signals contained in the design hierarchy. If the netlister-assigned name is required, you can pass these names to the `fnlSigCdsNameExtName` function to translate them during the header or footer evaluation. You can use the `fnlGetGlobalSigName` function at any time during the netlisting process.

Arguments

None

Value Returned

l_sigNames	A list of strings that contain names of the global signals.
`nil`	The operation failed and encountered an error while performing the task.

Examples

```
fnlGetGlobalSigNames()
```

Related Topics

OSS Functions

fnlInstCdsNameExtName

```
fnlInstCdsNameExtName(
    t_instName
    )
    => t_netName / nil
```

Description

Returns the netlister-assigned name for the instance name specified as an argument, if the instance is found.

Arguments

t_instName	The instance for which the netlister-assigned name must be found.

Value Returned

t_netName	The netlister-assigned name for the instance.
nil	The operation failed and encountered an error while performing the task.

Related Topics

OSS Functions

fnlPathList

```
fnlPathList(
      )
      => l_pathList / nil
```

Description

Returns a list representing the current instance path down the schematic hierarchy to the current instance or signal being expanded.

Arguments

None

Value Returned

l_pathList	A list of lists, where each sublist is a (inst cellview index) triplet, containing the `inst`, `cellview`, and `index`. The `inst` member can also be a `signal`.
`nil`	The operation failed and encountered an error while performing the task.

Examples

```
fnlPathList()
```

Related Topics

OSS Functions

fnlPrint

```
fnlPrint(
     g_general
     )
     => t / nil
```

Description

Prints the specified argument in the netlist file.

Arguments

g_general	The information that you need to print to a netlist file. The information can be of any of the SKILL types, such as `string`, `fixnum`, or `flonum`. The argument can also be `t` or `nil`, which do not add any text to the netlist file and are only a convenience.

Value Returned

`t`	The operation was successful.
`nil`	The operation was unsuccessful.

Related Topics

OSS Functions

fnlSearchPropString

```
fnlSearchPropString(
     t_propName
     g_localSearch
     )
     => t_propValue / nil
```

Description

Returns the property value for the property name specified as an argument.

Arguments

t_propName	The name of the property for which the value must be found.
g_localSearch	Boolean value. When set to `nil`, the function call corresponds to the substitution expression \"`[@ propName]`\". When set to `t`, it corresponds to the expression \"`[.propName]`\".

Value Returned

t_propValue	The property value corresponding to the property name specified as an argument.
`nil`	The operation failed and encountered an error while performing the task.

Related Topics

OSS Functions

fnlSigCdsNameExtName

```
fnlSigCdsNameExtName(
     t_sigName
     )
     => t_netName / nil
```

Description

Returns the netlister-assigned name for the signal name specified as an argument, if the signal is found.

Arguments

t_sigName	Represents the signal name for which the netlister-assigned name must be found.

Value Returned

t_netName	The netlister-assigned name for the signal.
nil	The signal was not found.

Related Topics

OSS Functions

fnlTermCdsNameExtName

```
fnlTermCdsNameExtName(
     t_sigName
     )
     => t_netName / nil
```

Description

Returns each netlister-assigned signal name for the signal attached to the terminal whose name is specified as an argument. The function is equivalent to the substitution expression `[ |name ]`. As with the matching expression, the terminal names allowed as arguments are restricted to the terminals attached to the current instance.

Arguments

t_sigName	Represents the signal name for which you need to find the netlister-assigned name.

Value Returned

t_netName	The netlister-assigned signal name for the signal specified as an argument.
`nil`	The terminal was not found.

Related Topics

OSS Functions

fnlTermExtName

```
fnlTermExtName(
     d_termId
     x_bit
     )
     => t_netName / nil
```

Description

Returns the netlister-assigned name for the signal attached to the bit of the terminal specified as an argument. The function is similar to the substitution expression `[ |name ]`. As with the matching expression, the terminals allowed as arguments are restricted to the formal terminals of the master of the current instance. This function must be called only during the evaluation of the `NLPcompleteElementString` property.

Arguments

d_termId	The formal terminal for which the netlister-assigned name must be found.
x_bit	The bit of the terminal for which the netlister assigned name must be found.

Value Returned

t_netName	The netlister-assigned name for the signal returned by the function, if the requested bit of the terminal is found.
`nil`	The requested bit of the terminal was not found.

Related Topics

OSS Functions

fnlTopCell

```
fnlTopCell(
    )
    => d_cellviewId / nil
```

Description

Returns the top level cellview being netlisted. The function can be called throughout the netlist process. It may be especially useful during the evaluation of the `NLPnetlistHeader` and `NLPnetlistFooter` properties, where you can use it to output information about the top level design for the header and footer of the netlist. There is no equivalent netlister-defined property.

Arguments

None

Value Returned

d_cellviewId	An object identifier of the top-level cellview.
`nil`	The operation failed and encountered an error while performing the task.

Examples

```
fnlTopCell()
```

Related Topics

OSS Functions

hnlAbortNetlist

```
hnlAbortNetlist(
      )
      => nil
```

Description

Aborts netlisting. When the formatter detects an error during netlisting, it calls this function to inform the netlister to abort netlisting. This function aborts netlisting at the next convenient point, usually after the current cell is processed.

Arguments

None

Value Returned

`nil`	Always returns `nil`.

Examples

```
hnlAbortNetlist()
```

Related Topics

OSS Functions

hnlAddExtraParameters

```
hnlAddExtraParameters(
    l_list
    )
    => t / nil
```

Description

Adds user-specified design variables to the OSS design variables list. For example, ADE-XL requires to add more design variables to OSS design variables list that are obtained after CDF callback evaluation.

These variables appear in control files created by OSS for internal use and therefore make sure that the design is not entirely re-netlisted on subsequent netlisting.

Arguments

l_list	List of strings, where each string is a parameter name

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

This function returns `nil` in the following cases:

- When OSS is not generating a netlist.
- When the input list is incorrectly formatted.

Examples

```
hnlAddExtraParameters((list "CAP0" "CAP1"))
```

Related Topics

OSS Functions

hnlCatIncrementalNetlistFiles

```
hnlCatIncrementalNetlistFiles(
     p_catFile
     l_netlistFileList
     )
     => t / nil
```

Description

Concatenates the list of netlist files into a single file whose file handle is given as the first parameter to this function. The second parameter is the list of files to be concatenated.

Arguments

p_catFile	File handle of the output file.
l_netlistFileList	Names of the netlist files to be concatenated.

Value Returned

t	The command is successful.
nil	The command was unsuccessful.

Examples

```
hnlCatIncrementalNetlistFiles( hnlIncludeFile,
                          hnlNetlistFileList )
```

Related Topics

OSS Functions

hnlCellExtracted

```
hnlCellExtracted(
     d_cellView
     )
     => t / nil
```

Description

Checks if the cellview has not been modified since it was last extracted.

Arguments

d_cellView	The `cellViewId` of the cellview checked.

Value Returned

`t`	The given cellview has not been modified since it was last extracted.
`nil`	The command was unsuccessful.

Examples

```
hnlCellExtracted( cellView )
```

Related Topics

OSS Functions

hnlCellInAllCells

```
hnlCellInAllCells(
     d_cellView
     )
     => t / nil
```

Description

Checks if the cellview argument exists as a cellview in the list of all cells. If not, this function returns `nil`.

Arguments

d_cellView	The `cellViewId` of the cell view checked. If hierarchical configuration is used, the following arguments must also be supplied: ■ *t_viewList*: Effective view list. ■ *t_pathName*: Path name string. ■ *t_LibName*: Configuration library name. ■ *t_cellName*: Cell name. ■ *t_viewName*: View name. ■ *t_isCellTopCell*: If this cell is the top-cell.

Value Returned

`t`	The `cellview` argument exists as a cellview in the list of all cells.
`nil`	The command was unsuccessful.

Examples

```
hnlCellInAllCells( cellView )
```

Related Topics

OSS Functions

hnlCloseCellFiles

```
hnlCloseCellFiles(
      )
      => t / nil
```

Description

Closes all opened files to netlist the cell from `hnlCurrentCell`. This function calls the `hnlWriteBlockControlFile` function to create the `control` file under the directory for `hnlCurrentCell`.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
hnlCloseCellFiles()
```

Related Topics

OSS Functions

hnlCloseMasterList

```
hnlCloseMasterList(
     l_inlists
     )
```

Description

Closes a list of cellviews one by one. This is the reverse function of `hnlGetMasterCells`.

Arguments

l_inlists	A list of cellviews.

Examples

```
hnlCloseMasterList( inlists )
```

Related Topics

OSS Functions

hnlCompletePrint

```
hnlCompletePrint(
    )
    => t
```

Description

Clears the buffers needed for the `hnlPrintString` function and closes the netlist file.

Arguments

None

Value Returned

`t`	The command is successful.

Examples

```
hnlCompletePrint()
```

Related Topics

OSS Functions

hnlDeRegPostNetlistTrigger

```
hnlDeRegPostNetlistTrigger(
     S_triggerFunc
     )
     => t / nil
```

Description

Deregisters a trigger, which is a user-defined SKILL procedure that has been registered using `hnlRegPostNetlistTrigger`.

Arguments

S_triggerFunc	A symbol representing a user-defined function that needs to be deregistered.

Value Returned

`t`	The trigger was deregistered successfully.
`nil`	The trigger could not be deregistered.

Examples

Thefollowing example deregisters the user-defined SKILL procedure `Func`, which was registered through `hnlRegPostNetlistTrigger`.

```
procedure( Func()
   let(()
      printf("In Func\n")
   )
)

when(simSimulator == "verilog"
   hnlRegPostNetlistTrigger('Func)
)

hnlDeRegPostNetlistTrigger('Func)
=> t
```

Related Topics

OSS Functions

hnlDeRegPreNetlistTrigger

```
hnlDeRegPreNetlistTrigger(
     S_triggerFunc
     )
     => t / nil
```

Description

Deregisters a trigger, which is a user-defined SKILL procedure that has been registered using `hnlRegPreNetlistTrigger`.

Arguments

`S_triggerFunc`	A symbol representing a user-defined function that needs to be deregistered.

Value Returned

`t`	The trigger was deregistered successfully.
`nil`	The trigger could not be deregistered.

Examples

Thefollowing example deregisters the user-defined SKILL procedure `Func`, which was registered through `hnlRegPreNetlistTrigger`.

```
procedure( Func()
   let(()
      printf("In Func\n")
   )
)

when(simSimulator == "verilog"
   hnlRegPreNetlistTrigger('Func)
)

hnlDeRegPreNetlistTrigger('Func)
=> t
```

Related Topics

OSS Functions

hnlDoInstBased

```
hnlDoInstBased(
     l_hnlListOfAllCells
     )
     => t
```

Description

Driver for instance-based netlists. It determines the order in which most of the output functions are called. For each cellview in the schematic hierarchy that is determined by the *hnlListOfAllCells* variable, it sets the *hnlCurrentCell* global variable to the cellview to be netlisted.

Then, the following global variables are set:

hnlCellInputs
hnlCellOutputs
hnlCellOthers
hnlCellOutTerms
hnlCellInTerms
hnlCellOtherTerms
hnlCellNetsOnTerms

Next, if this cellview is the top-level cellview (the cellview specified to be netlisted), each of the functions specified in the `hnlTopCellFuncs` list is evaluated in order. If this is not the top-level schematic, each of the functions specified in the `hnlMacroCellFuncs` list is evaluated in order. None of these global functions take arguments.

The current environment is always stored in global variables so that they are available to the output functions. Upon completion, `hnlCurrentCell` is set to `nil`.

Arguments

l_hnlListOfAllCells

List of all cell views to be netlisted. This is a list of lists, and the first element of each sublist is a cell. For example:

```
'( (cell1) (cell2) (cell3) )
```

Value Returned

`t`	Always returns `t`.

Examples

```
hnlDoInstBased( '( ( cellView1 ) ( cellView2 ) )
```

Related Topics

OSS Functions

hnlDoNetBased

```
hnlDoNetBased(
     g_cellList
     )
```

Description

The driver for net-based netlists. `cellList` is the list of cellviews, or macros used to generate the netlist. By default, this procedure is called by the `hnlPrintNetlist` function with `hnlListOfAllCells` as the list of cells to be netlisted.

Arguments

g_cellList	A list of cellviews used to generate the netlist.

Examples

```
hnlDoNetBased( cellList )
```

Related Topics

OSS Functions

hnlEMHGetDigitaGlobalNets

```
hnlEMHGetDigitaGlobalNets(
      )
      => globalNetList
```

Description

Returns a list of global nets found in digital design during mixed signal netlisting. This is used by the auCdl netlister to print `*.GLOBAL` statements for these nets in the top level netlist. This function must be called from Cadence internal formatters or ADE third-party integration formatters only when mixed signal netlisting is in progress.

Arguments

None

Value Returned

globalNetList	List of global net names.

Examples

```
hnlEMHGetDigitalGlobalNets()
=> ("vdd" "gnd")
```

Related Topics

OSS Functions

hnlEMHGetDigitalNetlistFileName

```
hnlEMHGetDigitalNetlistFileName(
      )
      => path
```

Description

Returns the netlist file path for digital netlist during mixed signal netlisting. This path is exported for use by formatters like auCdl to include digital netlist file name in the top-level netlist. This function must be called from Cadence internal formatters or ADE third-party integration formatters only when mixed signal netlisting is in progress.

Arguments

None

Value Returned

path	The netlist file path.

Examples

```
hnlEMHGetDigitalNetlistFileName()
=> "runDir/top.cdl"
```

Related Topics

OSS Functions

hnlEMHSetVerbosityLevel

```
hnlEMHSetVerbosityLevel(
     x_integer
     )
     => t
```

Description

Raises the level of informative output from the mixed signal netlister. To avoid cluttering of output, the level should be set to one.

Arguments

x_iteger	Level of informative output.

Value Returned

`t`	The level of informative output has been set to the specified value.

Examples

```
hnlEMHSetVerbosityLevel(1)
```

Related Topics

OSS Functions

hnlFindAllCells

```
hnlFindAllCells(
     d_cellView
     l_hnlListOfAllCells
     )
```

Description

Returns a list of all the unique devices and levels of the hierarchy that need to be netlisted. This is a list of lists where the first element of each sublist is a cell.

```
'( (cell1) (cell2) (cell3) )
```

This function recursively traverses the instance hierarchy and adds the master cellviews for each of the devices with the following characteristics to the returned list:

The instance is a “true” instance (i.e., not a terminal).
The master of the instance is not a stopping cellview.

The master does not have a string property `nlAction` set to `ignore`.
The master of the instance contains instances.

Each cellview is added to the `hnlListOfAllCells` list of cells only once, and the list is ordered so the cells that reference other cells occur later in the list than the cells they reference.

Because cellviews can come from multiple libraries, multiple entries may exist with the same `cellName`, but exist in different reference libraries.

In addition, any global net is added to the `hnlAllGlobals` list for later use by the formatting functions.

This function has been optimized for fast database traversal and instance access. Do not override this function unless absolutely necessary because you can slow down the netlisting run time.

Arguments

d_cellView	The `cellviewId` of the cellview.

l_hnlListOfAllCells

List of all cellviews to be netlisted. This is a list of lists, and the first element of each sublist is a cell. For example:

```
'( (cell1) (cell2) (cell3) )
```

Value Returned

None

Related Topics

OSS Functions

hnlFindAllInstInCell

```
hnlFindAllInstInCell(
     t_cellview
     )
```

Description

Finds all instances that can be netlisted in a cellview.

Arguments

t_cellview	Name of a cellview.

Examples

```
hnlFindAllInstInCell( cellview )
```

Related Topics

OSS Functions

hnlGenIncludeFile

```
hnlGenIncludeFile(
     )
     => value
```

Description

Creates the `include` file. You must define this function for a formatter that needs an `include` file. Use the *`hnlIncludeFileName`* variable to specify the name of the `include` file. Use the *`hnlIncludeFile`* variable to access the file pointer. IHNL calls the `hnlGenIncludeFile` function after it generates the netlist files.

By default this function does not create an `include` file.

Arguments

None

Value Returned

`value`	Value depends on the function logic.

Examples

```
hnlIfNoProcedure( hnlGenIncludeFile()
   let( ( name )
      if( hnlIncrementalMode
      then t
      else if( hnlAllGlobals
         then fprintf( hnlIncludeFile ".GLOBAL" )
            foreach( name hnlAllGlobals
               fprintf( hnlIncludeFile " %s"
                     hnlGetGlobalNetMappedName( name ) )
            )
            fprintf( hnlIncludeFile "\n" )
            t
         else hnlCatIncrementalNetlistFiles( hnlIncludeFile,
                     hnlNetlistFileList )
         )
      )
   )
)
```

Related Topics

OSS Functions

hnlGetCellHdbProps

```
hnlGetCellHdbProps(
     d_cellView
     )
     => list / nil
```

Description

Returns the list of HDB properties and their values for the given cellview, when called by the formatter during netlisting. The function uses the DBId of cellview to extract the `libName` and `cellName`. It further uses the `libName` and `cellName` combination to query the HDB properties from `prop.cfg` property file.

Arguments

d_cellView	DBId of the cellview that is searched to get `libName` and `cellName`.

Value Returned

list	The list of properties for the given cellview from the HDB.
nil	The command was unsuccessful.

Examples

```
hnlGetCellHdbProps( )
```

Related Topics

OSS Functions

hnlGetGlobalModelMappedName

```
hnlGetGlobalModelMappedName(
      t_cellName
      t_switchList
      )
      => name
```

Description

Returns a new mapped name for a given cell, if the argument specified as *cellName* is not a valid name for the target tool. Otherwise, an empty string is returned.

Arguments

t_cellName	Name of the global cell.
t_switchList	If a configuration is used, this argument is the switch list, else this argument is not used

Value Returned

name	The mapped name of the given global cell.

Examples

```
hnlGetGlobalModelMappedName( "nand" "schematic spice" )
```

Related Topics

OSS Functions

hnlGetGlobalNetMappedName

```
hnlGetGlobalNetMappedName(
      t_netName
      )
      => name
```

Description

Returns a new mapped name for a given global net, if the argument specified as *netName* is not a valid name for the target tool. Otherwise, an empty string is returned.

Arguments

t_netName	Name of global net.

Value Returned

name	The mapped names of a given global net.

Examples

```
hnlGetGlobalNetMappedName( "GND!" )
```

Related Topics

OSS Functions

hnlGetInstanceCount

```
hnlGetInstanceCount(
      )
      => numInstances
```

Description

Returns the number of instances netlisted in a single session. The function does not count the instances for which the `nlAction` property is set to `ignore`. This functions helps estimate the time taken to netlist and simulate based on the number of instances in a design. For example, this function can be used to implement a progress bar. A formatter can use this function only when OSS calls formatter functions to print a netlist.

Arguments

None

Value Returned

numInstances	An integer value, which represents the number of instances netlisted in a single session.

Examples

```
hnlGetInstanceCount()
```

Related Topics

OSS Functions

hnlGetMappedInstNames

```
hnlGetMappedInstNames(
      )
      => list
```

Description

Returns the list of all names mapped using the `hnlMapInstName` function. This function can only be called during netlisting.

Arguments

None

Value Returned

list	The list of all names mapped using the `hnlMapInstName` function.

Examples

```
hnlGetMappedInstNames()
```

Related Topics

OSS Functions

hnlGetMappedModelNames

```
hnlGetMappedModelNames(
      )
      => list
```

Description

Returns the list of all names mapped using the `hnlMapModelName` function. This function can only be called during netlisting.

Arguments

None

Value Returned

list	A list of all names mapped using the `hnlMapModelName` function.

Examples

```
hnlGetMappedModelNames()
```

Related Topics

OSS Functions

hnlGetMappedNames

```
hnlGetMappedNames(
      )
      => list
```

Description

Returns the list of all names mapped using the `hnlMapName` function. This function can only be called during netlisting.

In the case of inherited terminals and inherited connections, the netlister-generated names are returned.

Arguments

None

Value Returned

list	A list of all names mapped using the `hnlMapName` function. Assuming that the `hnlMapName` SKILL function is used to map all names in the entire design, the value is `list( "inh_gnd" "inh_vdd" )`. These two are mapped names for the local nets `vdd!` and `gnd!` in the `schematic` view of the cell `inv`.

Examples

```
hnlGetMappedNames()
```

Related Topics

OSS Functions

hnlGetMappedNetNames

```
hnlGetMappedNetNames(
      )
      => list
```

Description

Returns the list of all names mapped using the `hnlMapNetName` function. This function can only be called during netlisting.

In the case of inherited terminals and inherited connections, the netlister-generated names are returned.

Arguments

None

Value Returned

list	The list of all names mapped using the `hnlMapNetName` function. Assuming that the `hnlMapNetName` SKILL function is used to map all net names in the entire design, the value is `list( "inh_gnd" "inh_vdd" )`. These two are mapped names for the local nets `vdd!` and `gnd!` in the `schematic` view of the cell `inv`.

Examples

```
hnlGetMappedNetNames()
```

Related Topics

OSS Functions

hnlGetMasterCells

```
hnlGetMasterCells(
     l_instlist
     )
```

Description

Returns a list of cellviews of the view-switched masters for a list of instances, which are stored in the same order as the given list.

Arguments

l_instlist	A list of instances.

Examples

```
hnlGetMasterCells( instlist )
```

Related Topics

OSS Functions

hnlGetPrintLinePrefix

```
hnlGetPrintLinePrefix(
     )
     => t_value
```

Description

Returns the line prefix set using `hnlSetPrintLinePrefix` function, which is used to print individual subcircuits during auCdl netlisting to indicate the continuation of the text on the next line, when the line exceeds the maximum character limit.

Arguments

None

Value Returned

t_value	The prefix set as the indicator of the continuation of the subcircuit text in auCdl netlists. The prefix can be one of the following: ■ `+` ■ `*.PININFO`

Examples

```
hnlGetPrintLinePrefix()
```

Related Topics

OSS Functions

hnlSetPrintLinePrefix

hnlGetPropVal

```
hnlGetPropVal(
      t_propName
      d_cellView
      d_inst
      )
      => propValue / nil
```

Description

Returns the value of the property whose string name is given as the first argument. First, the property is searched for on the instance given as the third argument, and if not found there, on the cellview given as the second argument. If the property is found, the value is returned. Otherwise, `nil` is returned.

Arguments

t_propName	Name of the property to be retrieved.
d_cellView	The `propNameId` of the cellview that is searched for the property value.
d_inst	The `instId` of the instance that is searched for the property value.

Value Returned

propValue	The value of the property upon successful completion.
`nil`	The command was not successful.

Examples

```
hnlGetPropVal( "l" cellView inst )
```

Related Topics

OSS Functions

hnlGetRoundProp

```
hnlGetRoundProp(
      t_propName
      )
      => propValue / nil
```

Description

Locates the property of the given name, using `hnlGetPropVal`, `hnlCurrentMaster`, and `hnlCurrentInst`, and then returns the result rounded to the nearest integer. If the named property is not found, `nil` is returned.

If the property is of string type, the property value is evaluated before it is scaled.

Arguments

t_propName	Name of the property retrieved.

Value Returned

propValue	The value of the property rounded to the nearest integer
`nil`	The command was not successful.

Examples

```
hnlGetRoundProp( "l" )
```

Related Topics

OSS Functions

hnlGetScaleCapacitance

```
hnlGetScaleCapacitance(
     t_propName
     )
     => propValue / nil
```

Description

Locates the property of the given name, using `hnlGetPropVal`, `hnlCurrentMaster`, and `hnlCurrentInst`, divides the value by the value of the *`simCapUnit`* variable, and then returns the result rounded to the nearest integer. If the named property is not found, `nil` is returned. If `simCapUnit` is set to `nil`, the value of the property is returned, again rounded to the nearest integer.

If the property is of string type, the property value is evaluated before it is scaled.

Arguments

`t_propName`	Name of the property retrieved

Value Returned

`propValue`	The value of the property divided by the *`simCapUnit`* variable. The returned value is rounded to the nearest integer.
`nil`	The command was not successful.

Examples

```
hnlGetScaleCapacitance( "capValue" )
```

Related Topics

OSS Functions

hnlGetScaleMarginalDelay

```
hnlGetScaleMarginalDelay(
     t_propname
     )
     => propValue / nil
```

Description

Locates the property of the given name, using `hnlGetPropVal`, `hnlCurrentMaster`, and `hnlCurrentInst`, divides the value by the value of the *`simTimeUnit`* variable, multiplies it by the value of the *`simCapUnit`* variable, and then returns the result rounded to the nearest tenth. If the named property is not found, `nil` is returned. If *`simTimeUnit`* is set to `nil`, the value of the property is returned, again rounded to the nearest tenth.

If the property is of the string type, the property value is evaluated before it is scaled.

Argument

t_propName	Name of the property retrieved.

Value Returned

propValue	The value of the property divided by the *`simTimeUnit`* variable and then multiplied by the *`simCapUnit`* variable. The returned value is rounded to the nearest tenth.
`nil`	The command was not successful.

Examples

```
hnlGetScaleMarginalDelay( "l" )
```

Related Topics

OSS Functions

hnlGetScaleTimeUnit

```
hnlGetScaleTimeUnit(
     t_propName
     )
     => propValue / nil
```

Description

Locates the property of the given name, using `hnlGetPropVal`, `hnlCurrentMaster`, and `hnlCurrentInst`, divides the value by the value of the *`simTimeUnit`* variable, and then returns the result rounded to the nearest integer. If the named property is not found, `nil` is returned. If *`simTimeUnit`* is set to `nil`, then the value of the property is returned, again rounded to the nearest integer.

If the property is of the string type, the property value is evaluated before it is scaled.

Argument

t_propName	Name of the property retrieved.

Value Returned

propValue	The value of the property divided by the *`simTimeUnit`* variable. The returned value is rounded to the nearest integer.
`nil`	The command was not successful.

Examples

```
hnlGetScaleTimeUnit( "l" )
```

Related Topics

OSS Functions

hnlGetSimulator

```
hnlGetSimulator(
      )
      => ams
```

Description

Provides a distinction between AMS and other flows. When it is called from the AMS flow, the function displays AMS as output, otherwise it behaves the same as `simSimulator`.

Arguments

None

Value Returned

`ams`	Returns AMS when it is called from the AMS flow.

Examples

```
if(hnlGetSimulator()=="ams" then hnlVerilogIgnoreTerm=nil)
```

Related Topics

OSS Functions

hnlGetSourceFile

```
hnlGetSourceFile(
     d_instId | d_cellviewId
     )
     => path
```

Description

Retreives the path of source file of the given instance or cellview as specified in the HDB (`prop.cfg` property file), when this function is called by the formatter during netlisting.

Arguments

d_instId	The `instId` of the instance.
d_cellViewId	The `cellviewId` of the cellview.

Value Returned

path	The path of the source file of the given cellview or instance.

Examples

```
hnlGetSourceFile(hnlCurrentInst)
=> "/tmp/s1.v"

hnlGetSourceFile(hnlCurrentMaster)
=> "/tmp/cell.v"
```

Related Topics

OSS Functions

hnlGetSourceFileModels

```
hnlGetSourceFileModels(
      )
      => listCells
```

Description

Retreives a list of names of the netlisted cells, when this function is called by the formatter during netlisting. For the cells that have the `-subckt` value set using the `sourcefile_opts` property, the cell names in the list are replaced by the subcircuit names, for example, `-subckt <name>`.

Arguments

None

Value Returned

listCells	A list of names of the netlisted cells.

Examples

```
hnlGetSourceFileModels()
=> ("cell1" "cell2" "subckt_opts_name" "cell4")
```

Related Topics

OSS Functions

hnlGetSymbolPropVal

```
hnlGetSymbolPropVal(
    s_propSymbol
    d_cellView | d_inst
    )
    => propValue / nil
```

Description

Returns the value of the property whose symbol name is given as the first argument. First the property is searched on the instance given as the third argument, and if not found there, on the cellview given as the second argument. If the property is found, the value is returned; otherwise, `nil` is returned.

Arguments

s_propSymbol	Symbol name of the property retrieved.
d_cellView	The `cellViewId` of the cellview that is searched for the property value.
d_inst	The `instId` of the instance that is searched for the property value.

Value Returned

propValue	The value of the property whose symbol name is given as the first argument.
`nil`	The command was unsuccessful.

Examples

```
hnlGetSymbolPropVal( "l" cellView inst )
```

Related Topics

OSS Functions

hnlGetTermByName

```
hnlGetTermByName(
      t_cellview
      t_termname
      )
```

Description

Returns the terminal ID and index corresponding to the member *termname* found in the specified cellview, where the ID is the first element of the list and the index is the second. If the given terminal name implies a width greater than `1`, a `-1` is returned as the index. If a single bit name is given, then the index returned is the index to the terminal.

Arguments

t_cellview	Name of the cellview.
t_termname	Name of a terminal in the specified cellview.

Examples

```
hnlGetTermByName( cellview termname )
```

Related Topics

OSS Functions

hnlGetTermNameOfSig

```
hnlGetTermNameOfSig(
     g_sig
     )
```

Description

Finds the names of all the terminals that the specified signal connects to. The names are the single-bit names of the bits of the terminals that connect to the signal.

Arguments

g_sig	Name of a signal.

Examples

```
hnlGetTermNameOfSig( sig )
```

Related Topics

OSS Functions

hnlIfNoProcedure

```
hnlIfNoProcedure(
     t_progarg
     )
     => value
```

Description

Defines procedures in HNL. By using this function, the netlister only defines a procedure if it is not already defined, thus allowing user-override of netlister functionality by loading user-supplied functions before loading the Cadence netlister.

Source code for this procedure is available for CAD developers in the *install_dir*/tools/dfII/src/hnl/hnl.il file.

Arguments

t_progarg	Body of the procedure.

Value Returned

value	The value depends on the procedure logic.

Examples

```
hnlIfNoProcedure( simExecute( cmd )
let( ( status )
simDrain()
status = sh( cmd )
status
)
)
```

Related Topics

OSS Functions

hnlIgnoreTerm

```
hnlIgnoreTerm(
     d_term
     )
     => t / nil
```

Description

Checks if the terminal argument belongs to a patchcord or to an instance whose master is not a stopping cell and does not contain any instances. This function excludes terminals attached to instances such as `vdd` and `gnd`.

Arguments

d_term	The `termId` of the terminal.

Value Returned

`t`	The terminal belongs to a patchcord or to an instance whose master is not a stopping cell and which contains no instances.
`nil`	The command was not successful.

Examples

```
hnlIgnoreTerm( termA )
```

Related Topics

OSS Functions

hnlInitMap

```
hnlInitMap(
     x_maxNameLength
     x_maxInstNameLength
     x_maxModelNameLength
     x_maxNetNameLength
     x_maxTermNameLength
     t_namePrefix
     t_netNamePrefix
     t_instNamePrefix
     t_modelNamePrefix
     t_termNamePrefix
     t_hierarchyDelimeter
     l_invalidFirstChars
     l_invalidCharsInName
     l_hnlMapNetFirstChar
     l_hnlMapNetInName
     l_hnlMapInstFirstChar
     l_hnlMapInstInName
     l_hnlMapModelFirstChar
     l_hnlMapModelInName
     l_hnlMapTermFirstChar
     l_hnlMapTermInName
     t_mapName
     t_designName
     t_viewList
     t_stopList
     l_hnlInvalidNames
     l_hnlInvalidNetNames
     l_hnlInvalidInstNames
     l_hnlInvalidModelNames
     l_hnlInvalidTermNames
     )
     => t / nil
```

Description

Initializes the variables needed for name translation.

Arguments

x_maxNameLength Maximum number of characters that can exist in a name.

x_maxInstNameLength

Maximum number of characters that can exist in an instance name. If not given, `maxNameLength` is used as the default.

x_maxModelNameLength

Maximum number of characters that can exist in a model name. If not given, `maxNameLength` is used as the default.

x_maxNetNameLength Maximum number of characters that can exist in a net name. If not given, `maxNameLength` is used as the default.

x_maxTermNameLength

Maximum number of characters that can be in a terminal name. If not given, `maxNameLength` is used as the default.

t_namePrefix String prefix used when creating new names using the `hnlMapName` function. If this argument is `hnl_`, subsequent calls to the `hnlMapName` function return names such as `hnl_9` when a name requires full mapping.

t_netNamePrefix String prefix used when creating new names using the `hnlMapNetName()` function. If this argument is `net`, subsequent calls to the function `hnlMapNetName` return names such as `net9` when a name requires full mapping.

t_instNamePrefix String prefix used when creating new names using the `hnlMapInstName` function. If this argument is `inst`, subsequent calls to the `hnlMapInstName` function return names such as `inst9` when a name requires full mapping.

t_modelNamePrefix String prefix that should be used when creating new names using the `hnlMapModelName` function. If this argument is `model`, subsequent calls to the `hnlMapModelName` function return names such as `model9` when a name requires full mapping.

t_termNamePrefix String prefix that should be used when creating new names using the `hnlMapTermName` function. If this argument is `term`, subsequent calls to the `hnlMapTermName` function return names such as `term19` when a name requires full mapping.

t_hierarchyDelimeter

Hierarchy delimiter that the target simulator uses when creating flat names from the hierarchical netlist. For SILOS, this is "`(`". This argument should be a SKILL string containing a single character. This argument is later used when translating full pathnames from the simulator output.

l_invalidFirstChars

SKILL list that specifies which characters may not begin a name for the target simulator when using the `hnlMapName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping names with the `hnlMapName` function.

l_invalidCharsInName

SKILL list that specifies which characters may not be contained in a name for the target simulator when using the `hnlMapName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name containing this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character when it is encountered. This list is used when mapping names with the h`nlMapName` function.

l_hnlMapNetFirstChar

SKILL list that specifies which characters may not begin a name for the target simulator when using the `hnlMapNetName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping names with the `hnlMapNetName` function.

l_hnlMapNetInName

SKILL list that specifies which characters may not be contained in a name for the target simulator when using the `hnlMapNetName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name containing this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character when it is encountered. This list is used when mapping names with the `hnlMapNetName` function.

l_hnlMapInstFirstChar

SKILL list that specifies which characters may not begin a name for the target simulator when using the `hnlMapInstName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping names with the `hnlMapInstName` function.

l_hnlMapInstInName

SKILL list that specifies which characters may not be contained in a name for the target simulator when using the `hnlMapInstName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name containing this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character when it is encountered. This list is used when mapping names with the `hnlMapInstName` function.

l_hnlMapModelFirstChar

SKILL list that specifies which characters may not begin a name for the target simulator when using the `hnlMapModelName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping names with the h`nlMapModelName` function.

l_hnlMapModelInName

SKILL list that specifies which characters may not be contained in a name for the target simulator when using the `hnlMapModelName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name containing this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character when it is encountered. This list is used when mapping names with the `hnlMapModelName` function.

l_hnlMapTermFirstChar

SKILL list that specifies which characters may not begin a name for the target simulator when using the `hnlMapTermName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name beginning with this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character if it is encountered. This list is used when mapping names with the `hnlMapTermName` function.

l_hnlMapTermInName

SKILL list that specifies which characters may not be contained in a name for the target simulator when using the `hnlMapTermName` function. Each element in the list may be either a string or another list. If it is a string, it should contain a single character. This specifies that there is no replacement for this character and any name containing this character must be replaced. If an element is another list, it should contain two elements. The first element should be a string containing a single character that is the invalid character. If the second element of the sublist does not exist or is `nil`, the character specified by the first element is removed. If the second element is a string, that string replaces the character when it is encountered. This list is used when mapping names with the `hnlMapTermName` function.

t_mapName

Name of the file in which the name map is stored.

t_designName

Full name of the top level design.

Example: `/mnt/user1/alu/schematic/current.`

t_viewList

View switch list that is used for netlisting. Must be a SKILL string type.

Example: `"silos schematic"`

t_stopList

Stopping points used for netlisting. Must be a SKILL string type.

Example: `"silos mysilos"`

l_hnlInvalidNames A list of names which are invalid in the target tool name space. This is associated with generic names. This should not be used if you are using the following four arguments. They are mutually exclusive.

Example: `'("begin" "end" "for")`

l_hnlInvalidNetNames

A list of names which are invalid as net names in the target tool net name space.

Example: `'("begin" "end" "for" "net")`

l_hnlInvalidInstNames

A list of names which are invalid as instance names in the target tool instance name space.

Example: `'("begin" "end" "for" "instance")`

l_hnlInvalidModelNames

A list of names which are invalid as model names in the target tool model name space.

Example: `'("begin" "end" "for" "model")`

l_hnlInvalidTermNames

A list of names which are invalid as terminal names in the target tool terminal name space.

Example: `'("begin" "end" "for" "terminal")`

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
unless( hnlInitMap( hnlMaxNameLength hnlNamePrefix
        maxInstNameLength maxModelNameLength
       maxNetNameLength maxTermNameLength
       simNetNamePrefix simInstNamePrefix
       simModelNamePrefix simTermNamePrefix
       hnlHierarchyDelimeter hnlMapIfFirstChar
       hnlMapIfInName hnlMapNetFirstChar
       hnlMapNetInName hnlMapInstFirstChar
```

```
      hnlMapInstInName hnlMapModelFirstChar
      hnlMapModelInName hnlMapTermFirstChar
      hnlMapTermInName hnlMapFileName
      FullMasterName viewListString
      stopListString hnlInvalidNames
      hnlInvalidNetNames hnlInvalidInstNames
      hnlInvalidModelName hnlInvalidTermNames )

   sprintf(errorMessage
      "Netlister: Can't initialize name mapping functions.")
   println( errorMessage )
   return( nil )
)
```

Related Topics

OSS Functions

hnlInitPrint

```
hnlInitPrint(
     t_fileName
     x_lineLength
     t_prefix
     t_postfix
     t_commentStr
     x_softLineLength
     )
     => t / nil
```

Description

Opens the netlist output file specified as the filename and sets up the information required by the `hnlPrintString` function.

Arguments

t_fileName	Name of the output file.
x_lineLength	Maximum length of a line in the netlist.
t_prefix	Continuation character placed at the beginning of a continued line.
t_postfix	Continuation character placed at the end of a line to be continued.
t_commentStr	Comment string used by the simulator.
x_softLineLength	Maximum length of a line of output after which folding and continuation of the line need to be considered.

Value Returned

`t`	The initialization process completed without any error.
`nil`	The command was unsuccessful.

Examples

```
hnlInitPrint( "netlist" 60 "r" "o" "{" 50)
```

Related Topics

OSS Functions

hnlIsAPatchCord

```
hnlIsAPatchCord(
     d_inst
     )
     => t / nil
```

Description

Checks if the master of the given instance is a patchcord.

Arguments

d_inst	The `instId` of the instance whose master is checked.

Value Returned

`t`	The master of the given instance is a patchcord
`nil`	The command was unsuccessful.

Examples

```
hnlIsAPatchCord( inst )
```

Related Topics

OSS Functions

hnlIsAStoppingCell

```
hnlIsAStoppingCell(
     d_cellView
     )
     => t / nil
```

Description

Checks if the cellview argument is a stopping cellview. A cell is a stopping point for expansion if its `viewName` is in the `hnlViewList` switch view list as well as the `hnlStopList` stopping list, or if the cell has a string property `nlAction` with stop as its value. This function is used throughout the netlister to determine when to stop expansion and whether to output a macro reference or a reference to a primitive device.

Arguments

d_cellView	The `cellViewId` of the cellview checked.

Value Returned

`t`	The cellview specified as the argument is a stopping cell.
`nil`	The command was unsuccessful.

Examples

```
hnlIsAStoppingCell( cellView )
```

Related Topics

OSS Functions

hnlIsCellNetlistable

```
hnlIsCellNetlistable(
     t_cellview
     )
```

Description

Determines if instantiation of the specified cellview can be netlisted to HNL. This procedure returns `nil` for a cellview that contains no connectivity and for a cellview that has the property `nlAction` set to the value `ignore`.

Arguments

`t_cellview`	Name of a cellview.

Examples

```
hnlIsCellNetlistable( cellview )
```

Related Topics

OSS Functions

hnlIsCurrentInstStopping

```
hnlIsCurrentInstStopping(
      )
      => t / nil
```

Description

Checks if the current instance, `hnlCurrentInst`, is a stopping instance. An instance is a stopping point for expansion if the instance master is in the instance specific `stopList`. An instance is also a stopping point if the instance has a string property, `nlAction` with `stop` as its value. This function is used throughout the netlister to determine when to stop expansion and whether to output a macro reference or a reference to a primitive device.

Arguments

None

Value Returned

`t`	The current instance is a stopping instance.
`nil`	The command was unsuccessful.

Examples

```
hnlIsCurrentInstStopping()
```

Related Topics

OSS Functions

hnlIsCVInUserStopCVList

```
hnlIsCVInUserStopCVList(
    d_cellViewId
    )
    => t / nil
```

Description

Determines whether a cellview, which is present in `hnlUerStopCVList`, is considered a primitive or a stop view while netlisting.

Arguments

`d_cellViewId`	Id of the cellview, which is present in `hnlUerStopCVList`.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
hnlIsCVInUserStopCVList( hnlCurrentMaster )
```

Related Topics

OSS Functions

hnlMakeNetlistFileName

```
hnlMakeNetlistFileName(
     d_cellViewId
     t_effectiveViewList
     t_currentCellPath
     t_configLibName
     t_configCellName
     t_configViewName
     g_isTopConfigCell
     )
     => name
```

Description

Returns the relative name of the netlist file for `cellViewId`. This relative name is the path in the current netlisting directory. The `cellViewId` must be a non-stopping cellview, which you can netlist.

Arguments

d_cellViewId	The `cellViewId` of the cellview for which netlist file name is to be created.
t_effectiveViewList	Effective view list.
t_currentCellPath	Path to the current cell.
t_configLibName	Library name for effective config.
t_configCellName	Cell name for effective config for this cellview
t_configViewName	View name for effective config for this cellview.
g_isTopConfigCell	If the cellview is the top cellview.

Value Returned

name	The relative name of the netlist file for `cellViewId`.

Examples

The following example shows how to call the `hnlMakeNetlistFileName` function when HDB configuration is used for netlisting.

```
hnlMakeNetlistFileName( hnlCurrentCell hnlCurrentCellViewList hnlCurrentCellPath
hnlCurrentCellConfigLibname hnlCurrentCellConfigCellname
hnlCurrentCellConfigViewname hnlIsTopConfigCell  )
```

The variables, in the example, are replaced with their values at run-time.

The following example shows how to call the `hnlMakeNetlistFileName` function when HDB configuration is not used for netlisting.

```
hnlMakeNetlistFileName( cellView )
```

Related Topics

OSS Functions

hnlMapCellModuleName

```
hnlMapCellModuleName(
     d_cellViewId
     [ t_viewList
     t_pathName
     t_LibName
     t_cellName
     t_viewName
     t_isCellTopCell ]
     )
     => name
```

Description

Returns the netlist module name of the cellview, the `cellViewId`. It adds the `<cell name><module name>` mapped pair into the model section of the name map associated with the `hnlCurrentCell` variable. The `cellViewId` must be a non-stopping instance. For a design opened in the configuration view, you must specify the additional arguments.

Arguments

d_cellViewId	The `cellViewId` of cell view.
t_viewList	Effective view list. This argument must also be supplied if hierarchical configuration is used.
t_pathName	Path name string. This argument must also be supplied if hierarchical configuration is used.
t_LibName	Configuration library name. This argument must also be supplied if hierarchical configuration is used.
t_cellName	Cell name. This argument must also be supplied if hierarchical configuration is used.
t_viewName	View name. This argument must also be supplied if hierarchical configuration is used.

t_isCellTopCell

If this cell is top cell.

This argument must also be supplied if hierarchical configuration is used.

Value Returned

name The netlist module name of the cellview, the `cellViewId`.

Examples

Example 1: You can generate a SILOS netlist in which an instance in a schematic called 122 appears as follows:

```
(INST1 CKT122
```

Use the following code in your formatter:

```
fprintf( hnlNetlistFile
         hnlMapName( hnlCurrentInstName) )
fprintf( hnlNetlistFile
         hnlMapCellModuleName( hnlCurrentMaster ) )
```

To create the netlist for the schematic instance 122, use the following code:

```
fprintf( hnlNetlistFile "\n.MACRO %s"
         hnlMapCellModuleName( hnlCurrentCell ) )
```

Example 2: When working with the configuration view of a design, you can use the `hnlMapCellModuleName` function for the cell header in the `hnlPrintTopCellHeader` or `hnlPrintDeviceHeader` functions as follows:

```
mapname = hnlMapCellModuleName(
                     hnlCurrentCell
                     hnlCurrentCellViewList
                     hnlCurrentCellPath
                     hnlCurrentCellConfigLibname
                     hnlCurrentCellConfigCellname
                     hnlCurrentCellConfigViewname
                     hnlIsTopConfigCell
                     )
```

In the configuration view, you can also use the `hnlMapCellModuleName` function for a non-stopping cell in the `hnlPrintInst` function as follows:

```
mapname = hnlMapCellModuleName(
                     hnlCurrentMaster
                     hnlCurrentMasterViewList
                     hnlCurrentMasterPathString
                     hnlCurrentMasterConfigLibName
```

```
hnlCurrentMasterConfigCellName
hnlCurrentMasterConfigViewName
hnlCurrentMasterIsTopCellConfig
)
```

Related Topics

OSS Functions

hnlMapCellName

```
hnlMapCellName(
     d_cellViewId
     [ t_viewList ]
     [ t_pathName ]
     [ t_LibName ]
     [ t_cellName ]
     [ t_viewName ]
     [ t_isCellTopCell ]
     )
     => name
```

Description

Returns the netlist module name of the cellview, which is the `cellViewId`. It adds the `<cell name><module name>` mapped pair into the model section of the name map associated with the *`hnlCurrentCell`* variable. The `cellViewId` must be a non-stopping instance.

Arguments

`d_cellViewId`	The `cellViewId` of the cellview.
`t_viewList`	Effective view list. This argument must also be supplied if hierarchical configuration is used.
`t_pathName`	Path name string. This argument must also be supplied if hierarchical configuration is used.
`t_LibName`	Configuration library name. This argument must also be supplied if hierarchical configuration is used.
`t_cellName`	Cell name. This argument must also be supplied if hierarchical configuration is used.
`t_viewName`	View name. This argument must also be supplied if hierarchical configuration is used.
`t_isCellTopCell`	If this cell is top cell. This argument must also be supplied if hierarchical configuration is used.

Value Returned

`name`	The netlist module name of the cellview, the `cellViewId`.

Examples

```
hnlMapCellName( cvId "schematic spice" "/I1/I3" "myLib" "middle" "schematic" FALSE
)
```

Related Topics

OSS Functions

hnlMapInstName

```
hnlMapInstName(
      t_cdsName
      t_prefix
      )
      => name
```

Description

Returns a new mapped name. The *cdsName* string is a mandatory parameter and the prefix string is an optional one. If the argument specified in *cdsName* is legal, the same name is returned. Otherwise, OSS searches for the prefix. In case, the prefix is legal, then the value of global counter, which starts with `0`, is appended to the prefix. If the prefix is not legal or `null`, a new name using the global instance prefix and the global counter is generated. However, if both the prefix and the global instance prefix are illegal or `null`, then an error message is displayed.

Arguments

t_cdsName	Name of an instance.
t_prefix	Name of an instance prefix.

Value Returned

name	A new mapped name if the argument specified as the *cdsName* is not a valid name for the target simulator. Otherwise, the string is returned or an error message is displayed.

Examples

```
hnlMapInstName( "inst1" )
```

Related Topics

OSS Functions

hnlMapModelName

```
hnlMapModelName(
    t_cdsName
    [ t_viewList
    t_pathName
    t_LibName
    t_cellName
    t_viewName
    t_isCellTopCell ]
    )
    => name
```

Description

Returns a new mapped name if the argument specified as the *cdsName* string is not a valid name for the target simulator. Otherwise, the string is returned. The *hnlMapModelFirstChar*, *hnlMapModelInName*, *hnlMaxNameLength*, and *simModelNamePrefix* variables are used to determine the mapping of the name.

Arguments

t_cdsName	Name of a model.
t_viewList	Effective view list. This argument must also be supplied if hierarchical configuration is used.
t_pathName	Path name string. This argument must also be supplied if hierarchical configuration is used.
t_LibName	Configuration library name. This argument must also be supplied if hierarchical configuration is used.
t_cellName	Cell name. This argument must also be supplied if hierarchical configuration is used.
t_viewName	View name. This argument must also be supplied if hierarchical configuration is used.
t_isCellTopCell	Boolean value to indicate if the cell is a top cell. This argument must also be supplied if hierarchical configuration is used.

Value Returned

name	A new mapped name if the argument specified as the *cdsName* string is not a valid name for the target simulator. Otherwise, the string is returned.

Examples

```
hnlMapModelName( "block1" )
```

Related Topics

OSS Functions

hnlMapName

```
hnlMapName(
      t_cdsName
      )
      => name
```

Description

Returns a new mapped name if the argument specified as the *cdsName* string is not a valid name for the target simulator. Otherwise, the *cdsName* string is returned.

The *hnlMapIfFirstChar*, *hnlMapIfInName*, *hnlNamePrefix*, and *hnlMaxNameLength* variables are used to determine if a name is invalid. If an alternate string is specified for a invalid character, the string replaces the invalid character to create a new name. Characters can also be deleted from a name. If the name is too long or there is no character map, a new name is generated by using the prefix specified by the *hnlNamePrefix* variable, and suffixing it with a unique number. Assume the variables are set as follows:

```
hnlNamePrefix = "hnl_"
hnlMaxNameLength = 10
hnlMapIfFirstChar = '(("@" "__") "0" "1")
hnlMapIfInName = '(("@" "__"))
```

Then, `hnlMapName("alu@99")` would return `"alu__99"` and `hnlMapName("0.Y")` would return something like `"hnl_12"`. For more information on the mapping abilities, refer to the *hnlMapIfInName* and *hnlMapIfFirstChar* variables, as well as the `hnlInitMap` function.

If you use this function, you cannot use the `hnlMapInstName`, `hnlMapModelName`, `hnlMapTermName`, and `hnlMapNetName` functions.

In the case of inherited terminals and inherited connections, netlister-generated names are returned. `hnlMapName` cannot be used interchangeably with `hnlMapNetName` or other type-specific name mapping functions in the same netlisting session.

Arguments

t_cdsName	String name to be mapped.

Value Returned

name	A new mapped name if the argument specified as the *cdsName* string is not a valid name for the target simulator; otherwise, the `cdsName` string is returned.

Examples

```
hnlMapName( "alu@99" )
```

For net `vdd!` in the schematic view of cell `inv`, `hnlMapName( "vdd!" )` returns `"inh_vdd"`. For net `gnd!` in the same cellview, `hnlMapName( "gnd!" )` returns `"inh_gnd"`. These two are netlister-generated names due to inherited connection.

Related Topics

OSS Functions

hnlMapNetName

```
hnlMapNetName(
    t_cdsName
    )
    => name
```

Description

Returns a new mapped name if the argument specified as the *cdsName* string is not a valid name for the target simulator. Otherwise, the *cdsName* string is returned.

The *hnlMapNetFirstChar*, *hnlMapNetInName*, *hnlMaxNameLength*, and *simNetNamePrefix* variables are used to determine the mapping of the name.

In the case of inherited terminals and inherited connections, netlister-generated names are returned. `hnlMapNetName` or other type-specific name mapping functions cannot be used interchangeably with `hnlMapName` in the same netlisting session.

Arguments

t_cdsName	Name of a net.

Value Returned

name	A new mapped name if the argument specified as the *cdsName* string is not a valid name for the target simulator; otherwise, the *cdsName* string is returned.

Examples

```
hnlMapNetName( "netA" )
```

For net vdd! in the schematic view of cell `inv`, `hnlMapName( "vdd!" )` returns `"inh_vdd"`. For net `gnd!` in the same cellview, `hnlMapName( "gnd!" )` returns `"inh_gnd"`. These two are netlister-generated names due to inherited connection.

Related Topics

OSS Functions

hnlMapTermName

```
hnlMapTermName(
     t_cdsName
     )
     => name / nil
```

Description

Returns a new mapped name if the argument specified as the *cdsName* string is not a valid name for the target simulator. Otherwise, the *cdsName* string is returned.

The *hnlMapTermFirstChar*, *hnlMapTermInName*, *hnlMaxNameLength*, and *simTermNamePrefix* variables are used to determine the mapping of the name.

Arguments

t_cdsName	Name of a terminal.

Value Returned

name	A new mapped name if the argument specified as the *cdsName* string is not a valid name for the target simulator; otherwise, the *cdsName* string is returned.
nil	The command was unsuccessful.

Examples

```
hnlMapTermName( "IN1" )
```

Related Topics

OSS Functions

hnlMultipleCells

```
hnlMultipleCells(
     t_name
     )
     => t / nil
```

Description

Checks if the `cellname` argument exists for more than one cell name or cellview in the list of all cells. This function is used to detect cases of cells having the same name, but which are found in more than one location of the design hierarchy. This happens when cells with the same name are found in more than one library.

Arguments

t_name	Name of the cell.

Value Returned

`t`	The `cellname` argument is listed for more than one cell name or cellview in the list of all cells.
`nil`	The command was unsuccessful.

Examples

```
hnlMultipleCells( "inv" )
```

Related Topics

OSS Functions

hnlNameOfSignal

```
hnlNameOfSignal(
     d_net
     x_netIndex
     )
     => name / nil
```

Description

Returns a string for the name of the signal that defines the given net-index pair. This name is considered to be the preferred name and is consistent even when called with two different nets whose `~> name` field differs, but which are electrically equivalent at this level of the design hierarchy. If the `netIndex` is not in the range of bits for the given net, `nil` is returned.

If this bit of net is an inherited net or is connected to an inherited terminal then the *netlister-generated* name is returned.

Arguments

d_net	The `netId` of the net.
x_netIndex	Index of a bit of the net.

Value Returned

name	The preferred name of the signal that defines the given net-index) pair.
`nil`	The `netIndex` is not in the range of bits for the given net.

Examples

```
hnlNameOfSignal( net 0 )
```

The call to this function for signal *vdd!* in the `schematic` view of the cell `inv` `(hnlNameOfSignal( db_id_signal_vdd!, 0))` returns `inh_vdd`, which is the local name. It does not return the inherited signal name since that depends on the hierarchy lineage.

Related Topics

OSS Functions

hnlNetNameOnTerm

```
hnlNetNameOnTerm(
     t_termName
     x_bit
     )
     => name / nil
```

Description

Returns the signal name of the net attached to the given bit of the terminal of the given name on the current instance being expanded. This function guarantees the same name is returned even if the net is aliased, and this function is later called with a terminal attached to one of the aliases of this net.

If this bit of net is connected to an inherited terminal then the netlister-generated name is returned.

Arguments

t_termName	The name of the terminal.
x_bit	The bit of the terminal attached to the returned signal name of the net.

Value Returned

name	The signal name of the net attached to the specified bit of the specified terminal for the current instance being expanded.
nil	The terminal is not found or if the specified bit is not in the range of bits of the given terminal.

Examples

```
hnlNetNameOnTerm( "IN<3:0>" 2 )
hnlNetNameOnTerm( "IN" 0 )
```

Related Topics

OSS Functions

hnlNetNameOnTermName

```
hnlNetNameOnTermName(
      t_termName
      )
      => name / nil
```

Description

Returns the signal name of the net attached to the terminal of the given name on the current instance being expanded. This function guarantees the same name is returned even if the net is aliased, and this function is later called with a terminal attached to one of the aliases of this net.

Arguments

t_termName	The name of the terminal

Value Returned

name	The signal name of the net attached to the specified terminal for the current instance being expanded.
nil	The terminal is not found.

Examples

```
hnlNetNameOnTermName( "IN<3:0>")
```

Related Topics

OSS Functions

hnlNmpSetNameSpaces

```
hnlNmpSetNameSpaces(
      t_source
      t_dest
      )
      => t / nil
```

Description

Specifies the `nmp` namespaces to perform namespace mapping during OSS netlisting. The function accepts Cadence `nmp` supported namespaces as arguments. You can run the `%> nmp getSpaceNames` command to display a list of valid namespaces.

Arguments

t_source	Source namespace from which mapping is to be done, usually `CDBA`.
t_dest	Namespace in which netlist is to be written.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
hnlNmpSetNameSpaces("CDBA" "VHDL")
```

Related Topics

OSS Functions

hnlOpenTopCell

```
hnlOpenTopCell(
     t_lib
     t_cell
     t_view
     )
     => obj
```

Description

Opens the top-level cellview. If the `lib/cell/view` is an HDB config, then open the config before opening the top-level cellview.

Arguments

t_lib	The library name.
t_cell	The cell name.
t_view	The view name.

Value Returned

obj	The SKILL object of the dbId for the cellview passed for opening.

Examples

Here, `topblock` will give the SKILL object for the cellview opened.

```
topblock = hnlOpenTopCell( libName cellName viewName )
```

Related Topics

OSS Functions

hnlPcellIsParamOverridden

```
hnlPcellIsParamOverridden(
    dbobject
    t_string
    )
    => t / nil
```

Description

Finds if the Pcell parameter of a master is overridden at the instance.

Arguments

dbobject	Specifies the database ID of the cellview that is searched to get the library name and the cell name.
t_string	Specifies the string to be searched.

Value Returned

t	The Pcell parameter is overridden.
nil	The param is not overridden and an error is reported if no related parameter is found.

Related Topics

OSS Functions

hnlPostNetlistTriggerList

```
hnlPostNetlistTriggerList(

    )
    => l_triggerFunc / nil
```

Description

Returns the list of functions registered through `hnlRegPostNetlistTrigger`.

Arguments

None

Value Returned

l_triggerFunc	List of functions registered through `hnlRegPostNetlistTrigger`.
`nil`	No function was registered.

Examples

The following example lets you view the list of functions registered through `hnlRegPostNetlistTrigger`.

```
hnlPostNetlistTriggerList()
=> nil
```

Related Topics

OSS Functions

hnlPreNetlistTriggerList

```
hnlPreNetlistTriggerList(

    )
    => l_triggerFunc / nil
```

Description

Returns the list of functions registered through `hnlRegPreNetlistTrigger`.

Arguments

None

Value Returned

l_triggerFunc	A list of functions registered through `hnlRegPreNetlistTrigger`.
`nil`	No function was registered.

Examples

Thefollowing example shows how to view the list of functions registered through `hnlRegPreNetlistTrigger`:

```
hnlPreNetlistTriggerList()
=> nil
```

Related Topics

OSS Functions

hnlPrintDevices

```
hnlPrintDevices(
    )
    => t / nil
```

Description

Sets up the global variables needed by the output functions to output the connectivity for a device and then calls the `hnlPrintInst` formatter function to print the connectivity for each iteration of each instance in the current cellview being netlisted. The current cellview is determined by the value of the *`hnlCurrentCell`* global variable. The global variables set up by this function, which can be used by the formatter function, are as follows:

`hnlCurrentOutTerms`
`hnlCurrentInTerms`
`hnlCurrentOtherTerms`
`hnlCurrentOutputs`
`hnlCurrentInputs`
`hnlCurrentOthers`
`hnlCurrentType`
`hnlCurrentInst`
`hnlCurrentInstName`
`hnlCurrentIteration`
`hnlCurrentTermsOnInst`
`hnlCurrentMaster`

This function has been optimized for fast database traversal and instance access. Overriding this function will slow down the netlisting process.

Arguments

None

Value Returned

`t`	No error occurred during processing.
`nil`	The command was unsuccessful.

Examples

```
hnlPrintDevices()
```

Related Topics

OSS Functions

hnlPrintMessage

```
hnlPrintMessage(
     t_text
     )
     => t
```

Description

Prints the text argument to the `stdout` port if executed in the Cadence non-graphical environment. If the function is called from within the Cadence graphics environment, the output is written to the CIW window.

Use this function instead of the following `fprintf` function:

```
fprintf( stdout text )
```

The function is defined in `/bin/si` and also in the Cadence graphics program. You *cannot* modify this function.

Arguments

t_text	Text string.

Value Returned

`t`	The command was unsuccessful.

Examples

```
hnlPrintMessage( "Cannot open file simout.tmp\n" )
```

Related Topics

OSS Functions

hnlPrintNetlist

```
hnlPrintNetlist(
     l_hnlListOfAllCells
     )
     => t / nil
```

Description

Prints the netlist header, then calls a function to output the connectivity for the netlist, and after that, prints the netlist footer. At this point, all cells are expected to be bound. Error detection for binding is done by the traversal functions.

The netlist is output by calling the following functions in the given order:

```
hnlPrintNetlistHeader()
hnlDoInstBased(hnlListOfAllCells)
hnlPrintNetlistFooter()
```

Netlisting is controlled by two lists, `hnlListOfAllStopCells` and `hnlListOfAllCells`. To control netlisting of cells such that all the cells of a library are not netlisted, update these two lists before the netlister is invoked, preferably in `hnlPrintNetlistHeader` or the function called before this.

Arguments

l_hnlListOfAllCells

List of all cell views to be netlisted. This is a list of lists, and the first element of each sublist is a cell.

```
'( (cell1) (cell2) (cell3) )
```

Value Returned

`t`	No error occured during processing.
`nil`	An error occured during processing.

Examples

```
hnlPrintNetlist( '( ( cellView1 ) ( cellView2 ) )
```

Related Topics

OSS Functions

hnlPrintSignal

```
hnlPrintSignal(
      )
```

Description

Finds and directs the netlisting of all signals in `hnlCurrentCell`. It processes each signal and sets up the needed variables for use later. It is responsible for calling the `hnlPrintSignalHeader`, `hnlPrintInst`, and `hnlPrintSignalFooter` functions.

Arguments

None

Examples

```
hnlPrintSignal( )
```

Related Topics

OSS Functions

hnlPrintString

```
hnlPrintString(
    t_text
    [ g_general ]
    )
    => t / nil
```

Description

Prints the SKILL string text argument to the netlist file.

If a comment is too long to fit on one line, as determined by the *hnlMaxLineLength* variable, this function converts the comment into multiple single-line comments and adds the comment character, as specified by the *hnlCommentStr* variable, to the beginning of each line.

If a line is not a comment, but is too long, a new line is inserted after the maximum line length is reached. The maximum line length is determined by the *hnlMaxLineLength* variable. If the *hnlLinePostfix* variable is not `nil`, the string specified by this variable is appended to all continued lines. If the *hnlLinePrefix* variable is not `nil`, the string specified by this variable is placed at the beginning of all continued lines after the inserted new line.

This function also accepts an optional argument that specifies if a new line should be created after a maximum line length.

Arguments

t_text	Text to be printed to the netlist file.
g_general	When set as `t`, it specifies that the text to be printed should not be split when *hnlSoftLineLength* is reached, even when the text has blank spaces. However, if the number of characters printed on a single line exceeds the length specified by the *hnlMaxLineLength* variable, the line is broken after printing the string specified by the *hnlLinePostFix* variable. This argument can be used to print parameter values in the netlist file when parameter values have blank spaces.

Value Returned

`t`	The given string is printed to the netlist file.

`nil`	The command was unsuccessful.

Examples

```
hnlPrintString( "The current cell is a netlist primitive." )
```

Related Topics

OSS Functions

hnlRegPostNetlistTrigger

```
hnlRegPostNetlistTrigger(
     S_triggerFunc
     )
     => t / nil
```

Description

Registers a trigger, which is a user-defined SKILL procedure that is called after netlist generation.

The registration can be done in the `.simrc` file, at the CIW, or any other location that will be executed before generating the netlist.

Arguments

S_triggerFunc	A symbol representing a user-defined function that needs to be called after netlist generation.

Value Returned

`t`	The trigger registration completed successfully.
`nil`	The trigger registration could not be completed.

Examples

User-defined SKILL procedure, which will be called after netlist generation:

```
procedure( Func()
   let(()
      printf("In Func\n")
   )
)
```

If you want the above procedure `Func` to be run only for Verilog and SytemVerilog netlisting, set `simSimulator` to `"verilog"`. For Spectre, set `simSimulator` to `"spectre"`. If you do not set this variable, then the trigger will run for all netlisters.

If more than one triggers have been registered, then they are run in the order of registration.

```
when(simSimulator == "verilog"
   hnlRegPostNetlistTrigger('Func)
)
```

Related Topics

OSS Functions

hnlRegPreNetlistTrigger

```
hnlRegPreNetlistTrigger(
     S_triggerFunc
     )
     => t / nil
```

Description

Registers a trigger, which is a user-defined SKILL procedure that is called before the netlist generation.

The registration can be done in the `.simrc` file, at the CIW, or any other location that will be executed before generating the netlist.

Arguments

S_triggerFunc	A symbol representing a user-defined function that needs to be called before netlist generation.

Value Returned

`t`	The trigger registration is successful.
`nil`	The trigger registration was unsuccessful.

Examples

The following example shows a user-defined SKILL procedure, which is called before netlist generation.

```
procedure( Func()
   let(()
      printf("In Func\n")
   )
)
```

If you want the above procedure `Func` to be run only for Verilog and SytemVerilog netlisting, set `simSimulator` to `"verilog"`. For Spectre, set `simSimulator` to `"spectre"`. If you do not set this variable, then the trigger will run for all netlisters.

If more than one triggers have been registered, then they are run in the order of registration.

```
when(simSimulator == "verilog"
   hnlRegPreNetlistTrigger('Func)
)
```

Related Topics

OSS Functions

hnlRunNetlister

```
hnlRunNetlister(
    )
    => t / nil
```

Description

Runs the hierarchical netlister and is the entry point for HNL.

Source code for this procedure is available for CAD developers in the `install_dir/tools/dfII/src/hnl/hnl.il` file.

Arguments

None

Value Returned

`t`	No error is encountered during netlisting.
`nil`	An error was encountered during netlisting.

Examples

```
hnlRunNetlister()
```

Related Topics

OSS Functions

hnlScaleCapacitance

```
hnlScaleCapacitance(
     g_propVal
     t_propName
     )
     => value
```

Description

Accepts the given value as an argument, divides it by the value of the `simCapUnit` variable, and then returns the result rounded to the nearest integer. If `simCapUnit` isset to `nil`, then the value of the property is returned, again rounded to the nearest integer. The property name given as the second argument is used for error messages if the value does not evaluate to a number.

If the property is of the string type, the property value is evaluated before it is scaled.

Arguments

g_propVal	The scaled value.
t_propName	Name of the property.

Value Returned

value	The value of the first argument divided by the `simCapUnit` variable. The returned value is rounded to the nearest integer. If `simCapUnit` is set to `nil`, then the value of the first argument is returned, again rounded to the nearest integer.

Examples

```
hnlScaleCapacitance( 5 "l" )
hnlScaleCapacitance( "5" "l" )
```

Related Topics

OSS Functions

hnlScaleMarginalDelay

```
hnlScaleMarginalDelay(
     g_propVal
     t_propName
     )
     => value
```

Description

Accepts the value given as argument, divides the value by the value of the *simTimeUnit* variable, multiplies it by the value of the *simCapUnit* variable, and then returns the result rounded to the nearest tenth. If the named property is not found, `nil` is returned. If *simTimeUnit* is set to `nil` or *simCapUnitis* set to `nil`, the value of the property is returned, again rounded to the nearest tenth. The property name given as the second argument is used for error messages if the value does not evaluate to a number.

If the property is of the string type, the property value is evaluated before it is scaled.

Arguments

g_propVal	The scaled value.
t_propName	Name of the property.

Value Returned

value	The result of the first argument divided by the *simTimeUnit* variable and then multiplied by the *simCapUnit* variable. The returned value is rounded to the nearest tenth. If either *simTimeUnit* or *simCapUnit* is `nil`, the value of the first argument is returned, again rounded to the nearest tenth.

Examples

```
hnlScaleMarginalDelay( 5 "l" )
hnlScaleMarginalDelay( "5" "l" )
```

Related Topics

OSS Functions

hnlScaleTimeUnit

```
hnlScaleTimeUnit(
     g_propVal
     t_propName
     )
     => value
```

Description

Accepts the value given as argument, divides it by the value of the *simTimeUnit* variable, and then returns the result rounded to the nearest integer. If *simTimeUnit* is set to `nil`, the value of the property is returned, again rounded to the nearest integer. The property name given as the second argument is used for error messages if the value does not evaluate to a number. If the property is of the string type, the property value is evaluated before it is scaled.

Arguments

g_propVal	Value of the property.
t_propName	Name of the property.

Value Returned

value	The result of the first argument divided by the *simTimeUnit* variable. The returned value is rounded to the nearest integer. If *simTimeUnit* is `nil`, the value of the first argument is returned, again rounded to the nearest integer.

Examples

```
hnlScaleTimeUnit( 5 "l" )
hnlScaleTimeUnit( "5" "l" )
```

Related Topics

OSS Functions

hnlSetCellFiles

```
hnlSetCellFiles(
      )
      => t / nil
```

Description

Opens all the files associated with `hnlCurrentCell` and reinitializes name-mapping functions.

Arguments

None

Value Returned

`t`	No error is encountered during processing.
`nil`	An error was encountered during processing.

Examples

```
hnlSetCellFiles()
```

Related Topics

OSS Functions

hnlSetDef

```
hnlSetDef(
     s_sVariable
     g_value
     )
     => t / nil
```

Description

Sets variables in HNL. By using this function, the netlister only sets *sVariable* if it is not already set, or the symbol *sVariable* evaluates to `null`, thus allowing user-override of netlister variables by loading user-supplied defaults before loading the Cadence netlister.

Source code for this procedure is available for CAD developers in the *install_dir*/`tools/dfII/src/hnl/hnl.il` file.

Arguments

s_sVariable	Symbol name of the variable whose value is set to the *g_value* in the second argument. The value is set only if it is not already set, or if the symbol *sVariable* evaluates to `null`.
g_value	Value assigned to *sVariable* if it is `null`.

Value Returned

`t`	The variable is set to the specified value.
`nil`	The variable is not set to the specified value.

Examples

```
hnlSetDef( 'simSimulator "spice" )
```

Related Topics

OSS Functions

hnlSetMappingType

```
hnlSetMappingType(
     t_string
     )
     => t / nil
```

Description

Sets namespace mapping type for nets, terminals, globals, and models to `tabularOnly`, `nmpOnly`, or `nmpWithTabular`.

- `tabularOnly`: The default OSS name mapping type.
- `nmpOnly`: Maps invalid names in a formatter namespace. The function returns `null`, if `nmp` fails.
- `nmpWithTabular`: Maps invalid names in a formatter namespace similar to `nmpOnly` mapping type. However, if `nmp` fails or there is a conflict with already mapped names, OSS reverts to the `tabularOnly` mapping type.

Cadence and third party formatters, which are plugged into OSS, use the `hnlSetMappingType` function.

Arguments

t_string	The mapping type.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
hnlSetMappingType("tabularOnly")
hnlSetMappingtype("nmpOnly")
hnlSetMappingType("nmpWithTabular")
```

Related Topics

OSS Functions

hnlSetPrintLinePrefix

```
hnlSetPrintLinePrefix(
     t_value
     )
     => t / nil
```

Description

Registers the line prefix to be used when printing individual subcircuits during auCdl netlisting to indicate the continuation of the text on the next line, when the line length exceeds the limit set by the HNL global variable *hnlSoftLineLength*.

Arguments

t_value	The line prefix set as the indicator of the continuation of subcircuit text. The valid prefixes are: ■ + ■ *.PININFO

Value Returned

t	The command is successful.
nil	The command was unsuccessful.

Examples

```
hnlSetPrintLinePrefix("+")
```

Related Topics

OSS Functions

hnlSetPseudoTermDir

```
hnlSetPseudoTermDir(
     t_string
     )
     => t / nil
```

Description

Sets the direction of pseudo ports, which are created to propagate inherited connections from a cellview to upper levels in the hierarchy. OSS-based formatter calls this function to set the direction of pseudo ports. If the formatter does not call it, you can specify a statement to call the function in the `.simrc` file.

Arguments

`t_string`	Direction of pseudo ports. The argument can have any of these values: ■ `input`: Indicates that the direction of pseudo ports is `input`. ■ `output`: Indicates that the direction of pseudo ports is `output`. ■ `inputOutput`: Indicates that the direction of pseudo ports is `inputOutput`. ■ `sameAsTermDir`: Indicates that the direction of pseudo port created for explicit inherited terminals, is same as explicit terminal. The direction of rest of the pseudo ports is `inputOutput`.

Value Returned

`t`	No error is encountered during netlisting.
`nil`	An error was encountered during netlisting.

Examples

```
hnlSetPseudoTermDir("inputOutput")
```

Related Topics

OSS Functions

hnlSetVars

```
hnlSetVars(
      )
      => t / nil
```

Description

Sets all global-required variables for netlisting. Depending on the target simulator, the required netlist formatting functions are loaded.

After that, the *`hnlViewList`* and *`hnlStopList`* global variables are set to be lists of strings from the user-entered values for the view switch list and the stopping view list. The last step is to load a user-supplied file of functions if specified. If netlisting is not to continue, this function returns `nil`. If there is an error, the *`hnlError`* variable is also set to `true`. When run from within SE, this function sets the appropriate `hnl` variables from the ones in the SE environment.

Source code for this procedure is available for CAD developers in the *`install_dir`*`/tools/dfII/src/hnl/hnl.il` file.

Arguments

None

Value Returned

`t`	No error is encountered during processing.
`nil`	An error was encountered during processing.

Examples

```
hnlSetVars()
```

Related Topics

OSS Functions

hnlSortTerms

```
hnlSortTerms(
     d_term1
     d_term2
     )
     => t / nil
```

Description

Determines net order in the sorted lists of net names. It takes as argument two terminals and does a comparison by the name field. It is used as argument to the `sort` SKILL function.

Arguments

d_term1	The `termId` of `term1`.
d_term2	The `termId` of `term2`.

Value Returned

`t`	The name of the net connected to `term1` precedes that of the net connected to `term2`.
`nil`	The name of the net connected to `term1` does not precede that of the net connected to `term2`.

Examples

```
hnlSortTerms( term1 term2 )
```

Related Topics

OSS Functions

hnlSortTermsToNets

```
hnlSortTermsToNets(
     l_terminals
     )
     => list
```

Description

Returns a list of lists by accepting a list of terminals as argument. The first element in each list is the original terminal, the second element is the name of the net attached to that terminal. The list is sorted alphanumerically by the name of the terminal attached to the net.

The *`hnlCurrentIteration`* global variable is used to construct the name of the net attached to the current iteration of the current instance being expanded. For this reason, this function can only be called from the output-formatting functions for a stopping cell.

For inherited terminals, the netlister-generated names of the signals attached to the terminal are returned.

Arguments

`l_terminals`	List of terminal identifications, which is the `termId` of terminals.

Value Returned

`list`	A list of lists. The first element in each list is the original terminal, and the second element is the name of the net attached to that terminal. The list is sorted alphanumerically by the name of the terminal attached to the net.

Examples

```
hnlSortTermsToNets( '( term1 term2 term3 term4 term5 ) )
```

Related Topics

OSS Functions

hnlStartNetlist

```
hnlStartNetlist(
    )
    => t / nil
```

Description

Opens the output files.

Source code for this procedure is available for CAD developers in the *install_dir*/tools/dfII/src/hnl/hnl.il file.

Arguments

None

Value Returned

t	No error is encountered during netlisting.
nil	An error was encountered during netlisting.

Examples

```
hnlStartNetlist()
```

Related Topics

OSS Functions

hnlStopNetlist

```
hnlStopNetlist(
     )
     => t
```

Description

Cleans up after netlisting, closes open files, closes all open cellviews, and resets all global variables to nil.

Source code for this procedure is available for CAD developers in the *install_dir*/tools/dfII/src/hnl/hnl.il file.

Arguments

None

Value Returned

t	Always returns t.

Examples

```
hnlStopNetlist()
```

Related Topics

OSS Functions

hnlStringToList

```
hnlStringToList(
     t_theString
     )
     => list
```

Description

Accepts a string of white-space-separated strings, and returns a list of strings, that is, the string - `silos schematic` - returns the list `(silos schematic)`.

Source code for this procedure is available for CAD developers in the *`install_dir`*`/tools/dfII/src/hnl/hnl.il` file.

Arguments

`t_theString`	A string of white-space-separated strings.

Value Returned

`list`	A list of strings.

Examples

```
hnlStringToList( "behavorial structural verilog schematic symbol" )
```

Related Topics

OSS Functions

hnlWriteBlockControlFile

```
hnlWriteBlockControlFile(
     d_cellView
     [ t_viewList ]
     [ t_pathName ]
     [ t_LibName ]
     [ t_cellName ]
     [ t_viewName ]
     [ t_isCellTopCell ]
     )
     => t / nil
```

Description

Creates the control file for the cellview. The control file records information for subsequent IHNL runs. The `hnlCloseCellFiles` function calls the `hnlWriteBlockControlFile` function.

Arguments

d_cellView	The `cellViewId` of the cellview. This argument must also be supplied if hierarchical configuration is used.
t_viewList	Effective view list. This argument must also be supplied if hierarchical configuration is used.
t_pathName	Path name string. This argument must also be supplied if hierarchical configuration is used.
t_LibName	Configuration library name. This argument must also be supplied if hierarchical configuration is used.
t_cellName	Cell name. This argument must also be supplied if hierarchical configuration is used.
t_viewName	View name. This argument must also be supplied if hierarchical configuration is used.
t_isCellTopCell	If this cell is top cell. This argument must also be supplied if hierarchical configuration is used.

Value Returned

`t`	No error is encountered during processing.
`nil`	An error is encountered during processing.

Examples

```
hnlWriteBlockControlFile( cellView )
```

Related Topics

OSS Functions

hnlWriteMap

```
hnlWriteMap(
      )
      => t / nil
```

Description

Saves the name map tables in a file specified in an earlier call to the `hnlInitMap` function. This function takes no arguments.

The syntax of the map file cannot be relied upon because it is subject to change without notice. Always use Cadence-supplied access functions to translate names.

With the Incremental Hierarchical Netlister (IHNL), you can design a formatter that netlists incrementally. An incremental formatter checks each cellview in the design and netlists only the cellviews that the designer has modified since the previous netlisting. IHNL is only an option to HNL; it is not a separate netlister. It is important that you read the basic HNL documentation first, before reading this section, which describes only additional functionality to allow your formatter to be used in the incremental netlisting mode. For simplicity, IHNL will be referred to as the hierarchical netlister running with the incremental feature.

Arguments

None

Value Returned

`t`	No error is encountered during processing.
`nil`	An error was encountered and printed during processing.

Examples

```
hnlWriteMap()
```

Related Topics

OSS Functions

iseCloseSchWindow

```
iseCloseSchWindow(
     )
     => t / nil
```

Description

Closes the schematic window.

Arguments

None

Value Returned

`t`	The schematic window is closed.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
iseCloseSchWindow()
```

Related Topics

OSS Functions

iseCloseSimWindow

```
iseCloseSimWindow(
     )
     => t / nil
```

Description

Closes the simulation window.

Arguments

None

Value Returned

`t`	The simulation window is closed.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
iseCloseSimWindow()
```

Related Topics

OSS Functions

iseCommToSimulator

```
iseCommToSimulator(
     t_command
     )
     => t / nil
```

Description

Sends the command specified as the text parameter to the simulator. This function should be used in conjunction with input filtering, if input filtering is in effect.

Arguments

t_command	Name of the command.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
iseCommToSimulator( 'myfunc )
```

Related Topics

OSS Functions

iseCompleteInteractive

```
iseCompleteInteractive(
     )
     => t / nil
```

Description

Completes an interactive simulation session by closing all windows opened for interactive simulation, and updates the ISE state machine so that ISE is aware that the interactive session has ended.

Arguments

None

Value Returned

`t`	All the windows opened during interactive simulation are closed.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
iseCompleteInteractive()
```

Related Topics

OSS Functions

iseEnterNodeNamesList

```
iseEnterNodeNamesList(
     iseEnterPointFunc
     )
     => t / nil
```

Description

Prompts the user to enter node names into a form and returns the netlister-assigned names as a SKILL list of strings corresponding to the schematic names entered.

If nothing is specified in the form, the function returns `nil`.

Arguments

`iseEnterPointFunc`	Valid SKILL function to be used as a callback and registered by the `iseEnterNodeNamesList` function.

Value Returned

`t`	The selection of node names was successful.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
iseEnterNodeNamesList( 'myfunc )
```

Related Topics

OSS Functions

iseExitSimulator

```
iseExitSimulator(
     )
     => t / nil
```

Description

Terminates the simulation using the command string that is defined using the ISE variable *iseExitSimulatorCommand*. If the simulation is remote, this routine copies every file from the remote host back to the local simulation run directory. To use the `iseExitSimulator` function properly, write a function in which you call `iseExitSimulator`. After this function returns, you can do post-processing such as translating netlister-assigned names.

If there is no simulator, or it is determined not to be running, this function returns `nil`. This function also returns `nil` if it is a remote simulation, but files are not successfully copied back.

Arguments

None

Value Returned

`t`	The simulator is successfully terminated.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
iseExitSimulator()
```

Related Topics

OSS Functions

iseGetExtName

```
iseGetExtName(
     iseEnterPointsFunc
     )
     => t / nil
```

Description

Directs the enter points function to use the specified callback function. The `iseGetExtName` function returns the netlister-assigned name of the object being pointed at in the schematic window.

Arguments

`iseEnterPointsFunc`	Valid SKILL function to be used as a callBack.This function is registered by the `iseGetExtName()` function.

Value Returned

`t`	The selection is a success.
`nil`	The command was unsuccessful because a net, terminal, or instance is not selected, or the name of the object could not be translated, and an error message was printed in the log file.

Examples

```
iseGetExtName( 'myfunc )
```

Related Topics

OSS Functions

iseGetInputFromEncapWindow

```
iseGetInputFromEncapWindow (
     t_tmpStream
     )
     => t / nil
```

Description

This procedure is registered to be called whenever something is typed into the input window of the encapsulation window.The *tmpStream* argument is passed into the procedure to be packaged before it is sent to the simulator. This function also checks if the user has registered a function to be called so that the simulator inputs can be passed to this user-registered function for filtering. After filtering, the user can call the `isePrintSimulatorCommand` function such that the data will be sent to the simulator. All inputs typed into the encapsulation window are automatically reflected in the history section of the encapsulation window. If no user-function is registered then the inputs are sent directly to the simulator without filtering.

Arguments

t_tmpStream	Valid SKILL string which consists of commands that must be packaged and sent to the simulator.

Value Returned

`t`	The commands were successfully sent to the simulator.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
iseGetInputFromEncapWindow( "-v file.v -l run.log " )
```

Related Topics

OSS Functions

iseGetMappedProbeList

```
iseGetMappedProbeList(
     )
     => t / nil
```

Description

Returns a SKILL list of the netlister-assigned names of all of the nets currently probed in the schematic window.

Arguments

None

Value Returned

`t`	A list of internally mapped names of current probe is returned.
`nil`	The command was unsuccessful.

Examples

```
iseGetMappedProbeList()
```

Related Topics

OSS Functions

iseGetProbeList

```
iseGetProbeList(
     )
     => t / nil
```

Description

Returns a SKILL list of the user-assigned names of all nets currently probed in the schematic window.

Arguments

None

Value Returned

`t`	A list of internally mapped names of the current probe.
`nil`	The command was unsuccessful.

Examples

```
iseGetProbeList()
```

Related Topics

OSS Functions

iseInitSchematicWindow

```
iseInitSchematicWindow(
      )
      => t / nil
```

Description

Reads in the design specified by the *simLibName*, *simLibConfigName*, *simCellName*, *simViewName*, and *simVersionName* variables in the schematic window maintained by ISE.

Arguments

None

Value Returned

`t`	The simulator window was initialized for the current session.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
iseInitSchematicWindow()
```

Related Topics

OSS Functions

iseInitSimWindow

```
iseInitSimWindow(
      )
      => t / nil
```

Description

Initializes the simulator window.

First, the current window is set as the simulation window. Next, the menu for that window is set as the menu whose menu handle is specified by the *`iseSimulatorMenuHandle`* variable. No simulator menu will appear in the window unless this variable is set. Then, a check is performed to see if there is a netlist in the simulation run directory. If there is none, you are prompted to choose whether you want to netlist. A new netlist is generated and the simulator is invoked if you specify `Yes`. If you specify `No`, no netlist is generated, and the simulator is not invoked but the process remains inside the interactive simulation environment. This function returns `nil` if the simulator window is not accessible, if a simulator is running, or if the simulator is not invoked successfully; otherwise, this function returns `t`.

Arguments

None

Value Returned

`t`	The simulator window was initialized for the current session.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
iseInitSimWindow()
```

Related Topics

OSS Functions

iseInterruptSimulator

```
iseInterruptSimulator(
     )
     => t / nil
```

Description

Issues a soft interrupt to the simulator.

Arguments

None

Value Returned

`t`	A simulator window is accessible.
`nil`	A simulator window is inaccessible or the command was unsuccessful.

Example

```
iseInterruptSimulator()
```

Related Topics

OSS Functions

iseNetExtNameCdsName

```
iseNetExtNameCdsName(
      )
      => t / nil
```

Description

Displays a form for the user to enter a netlister-assigned node name. A second form appears which displays both this name and its corresponding user-assigned name.

Arguments

None

Value Returned

`t`	The form is successfully opened for the user.
`nil`	The form could not be opened or the command was unsuccessful.

Examples

```
iseNetExtNameCdsName()
```

Related Topics

OSS Functions

iseOpenWindows

```
iseOpenWindows(
      )
      => t / nil
```

Description

Opens three new windows and updates internal ISE structures required to keep track of the identifications of these windows. The first covers the bottom right quarter of the screen and is set to be the schematic window. The second occupies the upper half of the screen and is the waveform window. The third window occupies the lower left quarter of the screen and is used to run the simulator. The original windows are not altered, but instead are overlaid with the new windows. This is the default window configuration.

If the *iseDontOpenSchematicWindowIfOneExists* variable is set to `t`, ISE searches all windows and makes the ISE schematic window the first window it finds with the appropriate design that was opened in append mode. If no such window exists, ISE searches all windows and makes the ISE schematic window the first window it finds with the appropriate design that was opened in read or write mode. If no such window is found, ISE opens a new schematic window. If the `iseNoWaveformWindow` is set to `t`, ISE does not open a waveform window. In this case, the user must open a waveform window if it is needed.

Arguments

None

Value Returned

`t`	All the windows are successfully opened.
`nil`	The command was unsuccessful.

Examples

```
iseOpenWindows()
```

Related Topics

OSS Functions

isePrintName

```
isePrintName(
    )
    => t / nil
```

Description

Prints the netlister-assigned name of the object being pointed at in the schematic window to the input portion of the simulation window, starting from the current cursor position.

Arguments

None

Value Returned

`t`	The window is accessible.
`nil`	The window is inaccessible.

Examples

```
isePrintName()
```

Related Topics

OSS Functions

isePrintNameCB

```
isePrintNameCB(
    t_tempList
    )
    => t / nil
```

Description

The callback function of `isePrintName` which is to be called by the enter points function in `iseGetExtName`.

After receiving the mapped names, this function prints the names of the object selected from the schematic to the command line in the input section of the encapsulation window.

Arguments

t_tempList	The list of strings which are mapped names of the objects selected from the schematic window.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
isePrintNameCB( '("a" "b" "c" ) )
```

Related Topics

OSS Functions

isePrintSimulatorCommand

```
isePrintSimulatorCommand(
     [ commandArg | nil ]
     )
     => t / nil
```

Description

Used when you create menu commands that interface with both the design and simulator.

For example, it makes it possible to instruct the simulator to set a node by pointing at the node in the design. The SKILL function underlying the menu entry determines the name of the node pointed in the design, translates the name to the name assigned by the netlister, and issues the appropriate command to set the node with the determined name to the simulator.

Arguments

commandArg	If no argument is passed, then the function opens a form for the user to enter the command to be passed to the simulator.

Value Returned

`t`	The window is accessible.
`nil`	The command was unsuccessful.

Examples

```
isePrintSimulatorCommand( "-f test -v verilog.v" )
```

Related Topics

OSS Functions

iseReleaseNodeFrom

```
iseReleaseNodeFrom(
     )
     => t / nil
```

Description

After forcing a node to a preferred value, this function calls the function specified by the *`iseReleaseFunc`* variable to release the node from that value.

Arguments

None

Value Returned

`t`	The command is successfully sent to the simulator.
`nil`	The operation was unsuccessful.

Examples

```
iseReleaseNodeFrom()
```

Related Topics

OSS Functions

iseSearchForASchWindow

```
iseSearchForASchWindow(
      )
      => t / nil
```

Description

If *iseDontOpenSchematicWindowIfOneExists* is set to `t`, this function searches for the first existing schematic window with the correct design opened for edit and uses that window as the ISE schematic window.

If not available, this function searches for a window with the same design but opened for read only. If such a window is found, then the window is used as the ISE schematic window. If no such window is found, then it returns `nil`.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

```
iseSearchForASchWindow()
```

Related Topics

OSS Functions

iseSendOutputToEncapHistory

```
iseSendOutputToEncapHistory(
     t_myText
     )
     => t / nil
```

Description

Writes the string passed in as a parameter to the history section of the encapsulation window.

Arguments

t_myText	A valid SKILL string, integer or a float value. The function checks for the type of the argument passed and appends it to the history section of the encapsulation window.

Value Returned

`t`	The text is appendedto the history section of the encapsulation window.
`nil`	The command was unsuccessful and an error message was printed in the log file.

Examples

- For a string value

  ```
  iseSendOutputToEncapHistory( "Hello World")
  ```

- For an integer value

  ```
  iseSendOutputToEncapHistory( 10 )
  ```

- For a float value

  ```
  iseSendOutputToEncapHistory( 4.5 )
  ```

Related Topics

OSS Functions

iseSetEncapBindKeys

```
iseSetEncapBindKeys(
    )
```

Description

Sets the bindkeys for the encapsulation window.

Ensure that the bindkeys are set before opening the window. This is because setting bindkeys after the window is opened can adversely impact memory and time.

Arguments

None

Value Returned

None

Examples

```
iseSetEncapBindKeys()
```

Related Topics

OSS Functions

iseSetNodeTo

```
iseSetNodeTo(
    )
    => t / nil
```

Description

Replaces the `iseSet` function. It calls the function specified by the *`iseSetFunc`* variable.

Arguments

None

Value Returned

`t`	The function specified by *`iseSetFunc`* variable is set and a simulation window is accessible.
`nil`	The command was unsuccessful.

Examples

```
iseSetNodeTo()
```

Related Topics

OSS Functions

iseSimulate

```
iseSimulate(
      )
      => t / nil
```

Description

Calls the function specified by the *iseSimulateFunc* variable.

Arguments

None

Value Returned

`t`	The function set by the *iseSimulateFunc* variable is called and the simulation window exists.
`nil`	The command was unsuccessful.

Examples

```
iseSimulate()
```

Related Topics

OSS Functions

iseStartInteractive

```
iseStartInteractive(
    )
    => t / nil
```

Description

Sets up the default interactive simulation environment.

Before you start an interactive simulation session, you must have initialized the simulation environment by running the *Initialize* command, which can be found in the *Simulation* menu.

Three new windows are created which overlay any windows currently displayed. The creation and size of each window is controlled by global SKILL variables. The opening of windows is performed by executing the function specified by the *iseOpenWindowsFunc* variable.

One window displays waveforms. If a waveform file already exists in the run directory, the waveforms are read in, and the menu for that window is set to be the *Waveform* menu. Initialization of the waveform window is performed by executing the function specified by the *iseInitWaveWindowFunc* variable.

The second window displays the design. The top level of the design, as specified by the variables *simLibName*, *simLibConfigName*, *simCellName*, *simViewName*, and *simVersionName*, is automatically displayed.

The third window lets you interact with the simulator. To initialize this window, the function specified by the *iseInitSimWindowFunc* variable is run. The default for this variable is the `iseInitSimWindow` function. The simulator is automatically started and associated with this window. First, the function specified by *iseStartSimulatorFunc* is executed. The default value for this variable is the `iseStartSimulator` function. This function ensures that the simulation window is available and evaluates the *iseInvokeSimulatorFunc* variable to invoke the simulator. If it is `nil`, whichindicates that you have not defined it, the variable *iseRunSimulatorCommand* must be defined (the command string to invoke the simulator) so ISE can invoke your simulator. If the variable *iseInvokeSimulatorFunc* is set to the name of a function, that function is called so you can do any required preprocessing. Before the preprocessing function is called, the variable *iseRunSimulatorCommand* may or may not be defined. If it is not defined, it is expected to be defined before the preprocessing routine returns. The command string in *iseRunSimulatorCommand* is used by ISE to invoke your simulator.

The `iseInvokeSimulatorFunc` function takes precedence over `iseRunSimulatorCommand`. If both are defined at the beginning of the routine `iseStartSimulator`, the variable *`iseInvokeSimulatorFunc`* is always evaluated first; that is, the preprocessing routine will be called. If the variable *`iseInvokeSimulatorFunc`* is undefined (with value `nil`) and the variable *`iseRunSimulatorCommand`* is defined, the command string in *`iseRunSimulatorCommand`* is used directly to invoke the simulator. If both are undefined, or the preprocessing function evaluates to `nil`, this function will return `nil` and the simulator will not be invoked.

The menu for this window is set to the menu whose menu handle is specified by the *`iseSimulatorMenuHandle`* variable.

Because this default initialization sequence may not suit the needs of every user, many features can be separately parameterized with global SKILL variables.

Arguments

None

Value Returned

`t`	The interactive mode simulation was started.
`nil`	The command was unsuccessful.

Examples

```
iseStartInteractive()
```

Related Topics

OSS Functions

iseStartSimulator

```
iseStartSimulator(
    )
    => t / nil
```

Description

Issues the command to start the simulator in the simulator window by first evaluating the *iseInvokeSimulatorFunc* variable.

If it is undefined, the variable *iseRunSimulatorCommand* must be defined (the command string to invoke the simulator) so that ISE can invoke your simulator. If the variable *iseInvokeSimulatorFunc* is set to the name of a user-defined function, that function is called so that you can do any preprocessing needed. Before the pre-processing function is called, the variable *iseRunSimulatorCommand* may or may not be defined. If it is not, it is expected to be defined before your pre-processing function returns. The command string in *iseRunSimulatorCommand* is used by ISE to invoke your simulator. The `iseInvokeSimulatorFunc` function takes precedence over `iseRunSimulatorCommand`. If both are defined going into the routine `iseStartSimulator`, the variable *iseInvokeSimulatorFunc* is always evaluated first, that is, your pre-processing routine will be called. If the variable *iseInvokeSimulatorFunc* is undefined (with value `nil`) and the variable *iseRunSimulatorCommand* is defined, the command string in *iseRunSimulatorCommand* is used directly to invoke the simulator. If both are undefined or your pre-processing function evaluates to `nil`, this function will return `nil` and your simulator will not be invoked.

Arguments

None

Value Returned

`t`	The simulator was started.
`nil`	The command was unsuccessful.

Examples

```
iseStartSimulator()
```

Related Topics

OSS Functions

iseUpdateNetlist

```
iseUpdateNetlist(
      )
      => t / nil
```

Description

Generates a new netlist by calling the SE `netlist` function.

If there were no errors, the command specified by the `iseInputNetlistCommand` variable is issued to the simulator if a simulator window exists and the command is not `nil`.

Arguments

None

Value Returned

`t`	The netlist was generated.
`nil`	The command was unsuccessful.

Examples

```
iseUpdateNetlist()
```

Related Topics

OSS Functions

iseUpdateStimulus

```
iseUpdateStimulus(
      )
      => t / nil
```

Description

Runs the input name translation `simin` function.

It then issues the command specified by the *`iseInputStimulusCommand`* variable to the simulator if a simulator window exists and the command is not `nil`.

Arguments

None

Value Returned

`t`	The `simin` function was run successfully.
`nil`	The command was unsuccessful.

Examples

```
iseUpdateStimulus()
```

Related Topics

OSS Functions

netlist

```
netlist(
    )
    => t / nil
```

Description

Performs all the steps needed to generate a netlist. If the *simDoNetlist* variable is not set to `t`, the function returns `t` and does not run.

If the *simNetlistHier* variable is set, the `hnlRunNetlister` function is called to generate a hierarchical netlist; otherwise, the values of netlister control variables are printed and the `simNetlistWithArgs` function is called with the correct arguments to generate a flat netlist. The following variables are passed to the `simNetlistWithArgs` function for use by the flat netlister (FNL) to produce the netlist:

simLibName
simCellName
simViewName
simRunDir
simNlpGlobalLibName
simNlpGlobalCellName
simNlpGlobalViewName
simViewList
simStopList
simGlobalErrFileName
simProbeFileName
simSimulator
simTimeUnit
simCapUnit
simNetlistFileName

You must set the above variables correctly before calling this function. The function is defined in `/etc/skill/si/caplib/netlist.ile`. You can modify this function.

Arguments

None

Value Returned

`t`	The operation is successful.
`nil`	The operation was unsuccessful.

Examples

```
netlist()
```

Related Topics

OSS Functions

runsim

```
runsim(
    )
    => t / nil
```

Description

Runs the SKILL instructions stored in the *simCommand* variable, which is used to run the simulator for this simulation. The function then calls `simOutWithArgs` with the correct arguments to translate the textual simulator output stored in the `simout.tmp` file and generate the `si.out` file.

The following variables are used as arguments to the `simOutWithArgs` function and must be correctly set before calling this function.

simSedFile
simRunDir
simLibName
simCellName
simViewName

For more information on this translation process, refer to the `simOutWithArgs` or `simout` functions. The `runsim` function returns the return value of the *simCommand* variable. The function is defined in `/etc/skill/si/caplib/simulate.ile`. You can modify this function.

Arguments

None

Value Returned

`t`	The operation is successful.
`nil`	The operation was unsuccessful.

Examples

```
runsim()
```

Related Topics

OSS Functions

sim

```
sim(
    )
    => t / nil
```

Description

If this simulation is not being run in batch mode, as determined by the *simBatchFlag* variable being set to `nil`, the `simInitSimulator` function is called. The `simInitSimulator` function is called when SE first starts executing during its initialization phase. The function must be called again if run interactively because the user might have manually set simulator-specific variables such as *silosSimViewList*, which must replace the value of `simViewList` for it to affect netlisting. Calling `simInitSimulator` again ensures correct setting of global variables by simulator-specific variables.

Next, the *simActions* variable is set to the following if it has not already been set:

```
'( simCheckVariables()
   simInitRunDir()
   netlist()
   simin()
   runsim()
)
```

This is the default list of functions, to be executed in order, to run a complete simulation. If these functions do not provide the proper sequence of steps to be performed for a particular simulator, the variable can be set in the simulator-specific file stored in the `local/si/caplib` directory with the same name as the simulator with the `.ile suffix`. The variable can be set inside the file outside of any function or in the function of the same name as the simulator.

Next, each function specified in the *simActions* list is called in order. As soon as one of these functions returns a value other than `t`, the simulation is stopped, the string stored in the *simFailedMessage* variable is printed, and `nil` is returned. If all of the functions return `t`, the string stored in the *simCompleteMessage* variable is printed and `t` is returned. If the *simCompleteMessage* variable or *simFailedMessage* is `nil`, no message is printed.

The function is defined in `/etc/skill/si/caplib/simulate.ile`. You can modify this function.

Arguments

None

Value Returned

`t`	The operation is successful.
`nil`	The operation was unsuccessful.

Examples

```
sim()
```

Related Topics

OSS Functions

simAddProbeCapByName

```
simAddProbeCapByName(
      t_netName
      )
      => value
```

Description

Displays the net capacitance of the supplied net name. This function is to be used in conjunction with the function `simReadNetCapFile`, which reads in the `sim.cap` file, initializing all the net names in the design with the associated capacitance values. This function refreshes the active display window by displaying the capacitance value of the given net name.

Arguments

t_netName	The name of the net for which the capacitance value is required.

Value Returned

value	If the net name is valid, the capacitance value is placed on the net, otherwise, capacitance values for all the nets in the current design window are displayed.

Examples

```
simAddProbeCapByName( "net19" )
```

Related Topics

OSS Functions

simAddProbeCapByScreen

```
simAddProbeCapByScreen(
     )
```

Description

Displays all the nets capacitance on the current screen.

Arguments

None

Value Returned

None

Examples

```
simAddProbeCapByScreen()
```

Related Topics

OSS Functions

simAddProbeCapForBusBit

```
simAddProbeCapForBusBit(
      )
```

Description

Displays net capacitance on each bit of a bus.

This function is similar to the `simAddProbeCapByName` function, except that it places the capacitance values on each bit of the buses in the design.

Arguments

None

Value Returned

None

Examples

```
simAddProbeCapForBusBit()
```

Related Topics

OSS Functions

simCheckExist

```
simCheckExist (
     l_variableNames
     )
     => t / nil
```

Description

Checks whether all of the variables in the *`variableNames`* list argument are defined and are not set to `nil`. Otherwise, prints an error message.

The function is defined in `/etc/skill/si/simcap.ile`. You can modify this function.

Arguments

`l_variableNames`	List of variable names.

Value Returned

`t`	All variables in the list of variable names are defined.
`nil`	The command was unsuccessful and an error message is displayed.

Examples

```
simCheckExist('(simSimulator simRunDir) )
```

Related Topics

OSS Functions

simCheckHeader

```
simCheckHeader(
    p_inf
    )
    => t / nil
```

Description

Checks the header stamp of the net capacitance file pointer,`sim.cap`. The header stamp must be the first word on the first line of the `sim.cap` file, and it must be equal to `Net_Capacitance_File`. Use the `infile` function to open the `sim.cap` file before calling this function.

Arguments

`p_inf`	The valid SKILL `portId` of the input capacitance `sim.cap` file. There is no default value.

Value Returned

`t`	The file is a valid net capacitance file.
`nil`	The operation was unsuccessful.

Examples

```
simCheckHeader( infile("./sim.cap") )
```

Related Topics

OSS Functions

simCheckVariables

```
simCheckVariables(
     )
     => t / nil
```

Description

Checks whether the variables *simSimulator*, *simCellName*, *simLibName*, *simViewName*, *simRunDir*, *simViewList*, *simStopList*, *simSedFile*, *simCommand*, *simNlpGlobalLibName*, and *simNlpGlobalCellName* have been set and their value is `t`. It does so by calling the `simCheckExist` function.

The function is defined in `/etc/skill/si/simcap.ile`. You can modify this function.

Arguments

None

Value Returned

`t`	All of the variables have been set and are not `nil`.
`nil`	Prints an error message because one or more variables are not set or their value is `nil`.

Examples

```
simCheckVariables()
```

Related Topics

OSS Functions

simCheckViewConfig

```
simCheckViewConfig(
     t_libName
     t_cellName
     t_viewName
     )
     => t / nil
```

Description

Checks if the design you want to netlist is an HDB configuration.

Arguments

t_libName	Name of the library containing the design.
t_cellName	Cell name of the design.
t_viewName	View name of the design.

Value Returned

t	The given design is an HDB configuration.
nil	The given design is not an HDB configuration.

Examples

```
simCheckViewConfig( t_libName, t_cellName, t_viewName )
=> t
```

Related Topics

OSS Functions

simCleanRun

```
simCleanRun(
    )
    => t / nil
```

Description

Deletes files created by both SE and the analysis tool being used from the simulation run directory. The function deletes only files that can be recreated by renetlisting or resimulating. Files which are required to rerun the simulation, such as the `si.env` and `control` files, are not deleted. The function displays a dialog box so you can confirm that you intend to delete the information. If the deletion is confirmed, the files that SE creates are deleted along with the files specified by the *`simCleanFileList`* variable. The *`simCleanFileList`* variable is set differently by each application integrated into SE.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simCleanRun( )
```

Related Topics

OSS Functions

simDateStamp

```
simDateStamp(
      t_string
      )
      => t / nil
```

Description

Prints the value contained in *t_string* argument followed by the date to standard output.

The function is defined in `/etc/skill/si/caplib/util.ile`. You can modify this function.

Arguments

t_string	A string value.

Value Returned

`t`	The operation was successful.
`nil`	The operation was unsuccessful.

Examples

```
simDateStamp( "Begin netlisting:" )
=> t
```

Related Topics

OSS Functions

simDeleteRunDirFile

```
simDeleteRunDirFile(
     t_fileName
     )
     => t / nil
```

Description

Deletes the specified file in the simulation run directory.

The function is defined in `/etc/skill/si/caplib/util.ile`. You can modify this function.

Arguments

t_fileName	Name of the file to be deleted in the run directory.

Value Returned

t	The file deletion is successful.
nil	The file deletion was unsuccessful.

Examples

```
simDeleteRunDirFile( "raw/waves" )
```

Related Topics

OSS Functions

simDeRegPostNetlistTrigger

```
simDeRegPostNetlistTrigger(
     S_triggerFunc
     )
     => t / nil
```

Description

Deregisters a user-defined SKILL trigger that has been registered using `simRegPostNetlistTrigger`. For backward compatibility, `hnlDeRegPostNetlistTrigger` can be used as an alias for `simDeRegPostNetlistTrigger`.

The `simDeRegPostNetlistTrigger` function can be called in the `.simrc` or the `libinit.il` file and *`S_triggerFunc`* can be defined in any one of these files.

Arguments

`S_triggerFunc`	Name of SKILL function symbol.

Value Returned

`t`	The specified SKILL trigger was deregistered.
`nil`	The specified SKILL trigger could not be deregistered.

Examples

```
simDeRegPostNetlistTrigger( 'S_triggerFunc )
```

Related Topics

OSS Functions

simDeRegPreNetlistTrigger

```
simDeRegPreNetlistTrigger(
     S_triggerFunc
     )
     => t / nil
```

Description

Deregisters a user-defined SKILL trigger that has been registered using `simRegPreNetlistTrigger`. For backward compatibility, `hnlDeRegPreNetlistTrigger` can be used as an alias for `simDeRegPreNetlistTrigger`.

The `simDeRegPreNetlistTrigger` function must be called in the `.simrc` file and *`S_triggerFunc`* must be defined in the `.simrc` file.

Arguments

`S_triggerFunc`	Name of SKILL function symbol.

Value Returned

`t`	The specified SKILL trigger was deregistered.
`nil`	The specified SKILL trigger could not be deregistered.

Examples

```
simDeRegPreNetlistTrigger( 'S_triggerFunc )
```

Related Topics

OSS Functions

simDesignVarCdsNameExtName

```
simDesignVarCdsNameExtName(
     t_cdsDesignVarName
     )
     => t_string
```

Description

Accepts the user-assigned variable name, *cdsDesignVarName*, as an argument and returns the netlister-assigned name for it as it appears in the netlist. The function reads the mapping information from the `simRunDir` directory using the design information set through the *simCellName*, *simLibName*, *simViewName* SKILL variables. Therefore, all these variables must be set before calling this function.

Arguments

t_cdsDesignVarName

The user-assigned name of the design variable.

Value Returned

t_string	Netlister-assigned design variable name as it appears in the netlist.

Examples

```
simDesignVarCdsNameExtName( "scale" )
=> "_qpar0"
```

Related Topics

OSS Functions

simDesignVarExtNameCdsName

```
simDesignVarExtNameCdsName(
     t_extDesignVarName
     )
     => t_string
```

Description

Accepts the netlister-assigned variable name, *extDesignVarName*, as an argument and returns the user-assigned name for it as it appears on the schematic. The function reads the mapping information from the `simRunDir` directory using the design information set through the *simCellName*, *simLibName*, *simViewName* SKILL variables. Therefore, all these variables must be set before calling this function.

Arguments

t_extDesignVarName

Name of the netlister-assigned design variable name.

Value Returned

t_string	User-assigned name of design variable, as it appears on the schematic.

Examples

```
simDesignVarExtNameCdsName( "_qpar0" )
=>  "scale"
```

Related Topics

OSS Functions

simDrain

```
simDrain(
      )
      => t
```

Description

Performs the equivalent of `drain( stdout )` when run in the non-graphic environment without accepting any arguments. No action is taken or required when this function is run in the Cadence graphics environment.

This is a replacement for the `drain` SKILL function. Replace all references to `drain( stdout )` with calls to this function.

The function is defined in `/bin/si`. You cannot modify this function.

Arguments

None

Value Returned

`t`	Always returns `t`.

Examples

```
simDrain()
```

Related Topics

OSS Functions

simEditFileWithName

```
simEditFileWithName(
     t_fileName
     )
     => ipcId / nil
```

Description

Calls the `edit` function internally to display the specified file.

Arguments

t_fileName	Name of the file.

Value Returned

ipcId	The Id of the process initiated by call to MPS to view the file in VI.
nil	The command was unsuccessful.

Examples

```
simEditFileWithName("/tmp/skill.il")
```

Related Topics

OSS Functions

simEnvExecute

```
simExecute(
     t_command
     )
     => t / nil
```

Description

Runs the UNIX command or program provided as its single argument.

The function is defined in `/etc/skill/si/simcap.ile`. You can modify this function.

Arguments

`t_command`	Name of the UNIX command or program.

Value Returned

`t`	The command is successful and the UNIX exit status is `0`.
`nil`	The command is unsuccessful.

Examples

```
simExecute("exec cat si.env")
```

Related Topics

OSS Functions

simFindFile

```
simFindFile(
     t_filename
     )
     => t_filename / nil
```

Description

Finds the full file system pathname to `t_filename` in the install hierarchy, if it exists. The pathname this function searches for is specified by the `prependInstallPath` function.

The function is defined in `/bin/si`.You cannot modify this function.

Arguments

t_filename	Name of the file to be searched.

Value Returned

t_filename	The full file system pathname of the install hierarchy if it exists.
nil	The pathname cannot be returned.

Examples

```
simFindFile("defaults.il")
```

Related Topics

OSS Functions

simFlattenWithArgs

```
simFlattenWithArgs(
     t_simLibName
     t_simCellName
     t_simViewName
     t_simRunDir
     t_simNlpGlobalLibName
     t_simNlpGlobalCellName
     t_simNlpGlobalViewName
     l_simViewList
     l_simStopList
     t_simGlobalErrFileName
     t_simProbeFileName
     t_simSimulator
     f_simTimeUnit
     f_simCapUnit
     t_simFlatLibName
     t_simFlatCellName
     t_simFlatViewName
     t_simFlatViewTypeName
     )
     => t / nil
```

Description

Runs the `prFlatten` tool. The design hierarchy is specified by the *simLibName*, *simCellName*, and *simViewName* variables. The global formatting properties are defined in `simNlpGlobalLibName`, `simNlpGlobalCellName` and `simNlpGlobalViewName`. The function is defined in `/bin/si` and also in the Cadence graphics program.

Arguments

t_simLibName	Name of the library containing the top-level cellview of the design.
t_simCellName	Cell name of the top-level cellview (design) to be netlisted.
t_simViewName	View name of the top-level cellview of the design.
t_simRunDir	Full file system pathname to the simulation run directory. This pathname must be the same as the *simRunDir* global SE variable, for example, `"/mnt2/user1/simulations/silos1"`

t_simNlpGlobalLibName

The name of the library that contains the global cellview. The global cellview defines global format strings. Global format strings, in turn, define the netlist syntax. The library name be the same as the *simNlpGlobalLibName* global SE variable, for example,

```
"basic"
```

t_simNlpGlobalCellName

The cell name of the global cellview that contains the global format strings that define the netlist syntax. This cell name must be the same as the *simNlpGlobalCellName* global SE variable, for example,

```
"nlpglobals"
```

t_simNlpGlobalViewName

The view name of the global cellview that contains the global format strings that define the netlist syntax. This view name must be the same as the *simNlpGlobalViewName* global SE variable, for example,

```
"verilog"
```

l_simViewList

View switch list used to determine which view to switch into when a cell has been found. This list must be the same as the *simViewList* global SE variable, for example,

```
list("verilog" "schematic")
```

l_simStopList

Stopping view list used to determine when to halt the expansion of the hierarchy. This list must be the same as the *simStopList* global SE variable, for example,

```
list("verilog")
```

t_simGlobalErrFileName

Name of the file that stores global error messages.

t_simProbeFileName

Name of the probe file that stores errors, usually `probe.err`.

t_simSimulator Simulator for which the netlist syntax is intended. This simulator name must be the same as the *simSimulator* SE global environment variable, for example,

```
"verilog"
```

f_simTimeUnit Floating-point time unit factor, for example, 1.0e-9.

f_simCapUnit Floating-point capacitance unit factor, for example, 1.0e-15.

t_simFlatLibName

Name of the library which contains the flattened cellview.

t_simFlatCellName

Cell name of the flattened cellview.

t_simFlatViewName

View name of the flattened cellview.

t_simFlatViewTypeName

Name of the view type of the flattened cellview.

Value Returned

t	The command is successful.
nil	The command is unsuccessful.

Examples

```
simFlattenWithArgs( simLibName
   simCellName
   simViewName
   "mnt/dave/chip1/spice1.run"
   simNlpGlobalLibName
   simNlpGlobalCellName
   simNlpGlobalViewName
   simViewList
   simStopList
   "global.err"
   "probe.err"
   simSimulator
   float(simTimeUnit)
   float(simCapUnit)
   "flat"
   "test"
   "autoLayout"
   "maskLayout"
)
```

Related Topics

OSS Functions

simGetLoginName

```
simGetLoginName(
      )
      => t_string
```

Description

Returns the string-valued login name of the current user.

The function is defined in `/bin/si`. You cannot modify this function.

Arguments

None

Value Returned

t_string	The login name of the user.

Examples

```
simGetLoginName()
= >  "user1"
```

Related Topics

OSS Functions

simGetTermList

```
simGetTermList(
     t_libName
     t_cellName
     t_viewName
     )
     => l_list / nil
```

Description

Collects all the terminals for a block and returns them as a list.

Arguments

t_libName	Name of the library containing the design.
t_cellName	Cell name of the design.
t_viewName	View name of the design.

Value Returned

l_list	SKILL list of the terminals in the specified cellview.
nil	The command was unsuccessful.

Examples

```
simGetTermList( "top" "cell" "schematic" )
```

Related Topics

OSS Functions

simIfNoProcedure

```
simIfNoProcedure(
     procedureDefinition
     )
     => t
```

Description

Same as the SKILL keyword `procedure`, except the procedure definition given as an argument is only defined if it is not currently defined. This function is used instead of the SKILL `procedure` to permit the overriding of procedures defined in SE by procedures in `.simrc`.

The function is defined in `/etc/skill/si/caplib/util.ile`. You can modify this function.

Arguments

`procedureDefinition`

SKILL procedure definition passed as an argument.

Value Returned

`t`	The command is successful.

Examples

```
simIfNoProcedure( cat(file)
   let( (cmd status)
      sprintf( cmd "exec cat %s" file )
      status = simExecute(cmd)
      status
   )
)
```

Related Topics

OSS Functions

simIlSleep

```
simIlSleep(
     x_seconds
     )
     => t / nil
```

Description

Suspends the current process by the number of seconds specified as an argument. The current process in a replay file launches `si` in batch mode and waits for `si` to return before it runs the next command in the file.

Arguments

x_seconds	Number of seconds by which the run of a process is to be delayed.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simIlSleep(4)
```

Related Topics

OSS Functions

simin

```
simin(
    )
    => t / nil
```

Description

Translates the user-assigned names for nets and instances in the `control` file to the corresponding netlister-assigned names and creates the `si.inp` file for input to the simulator. A message is printed stating that `simin` is being run, and then the `simInWithArgs` function is called with the correct arguments to do the translation.

The following variables are used as arguments to the `simInWithArgs` function and must be correctly set before calling this function:

simRunDir
simLibName
simCellName
simViewName

All text in the control file is copied to the `si.inp` file unless the text is surrounded by square brackets (`[ ]`). The opening square bracket (`[`) indicates that the following text up to the closing square bracket (`]`) is to be interpreted. The entire expression is replaced by the resulting interpreted value. Following the opening square bracket (`[`) should be one of the following command characters:

`#`	`[#netname]`	Replace the [`#netname`] expression with the netlister-assigned net name for `netname`.
`$`	`[$instname]`	Replace the [`$instname`] expression with the netlister-assigned instance name for `instname`.
`!`	`[!filename]`	Replace the [`!filename`] expression with the contents of the `filename` file. If you use a relative pathname, the system will look for the file in the simulation run directory. To access a file outside the simulation run directory, use a full file system pathname. If the file does not exist, an error is generated.
`?`	`[?filename]`	Replace the [`?filename`] expression with the contents of the `filename` file. If you use a relative pathname, the system will look for the file in the simulation run directory. To access a file outside of the simulation run directory, use a full file system pathname. If the file does not exist, no error is generated.

`n!`	`[n!filename]`	Same as `[! filename]`, except that the contents of the new file are not parsed, and square-bracketed expressions are not interpreted.
`n?`	`[n?filename]`	Same as `[? filename]`, except that the contents of the new file are not parsed, and square-bracketed expressions are not interpreted.

The function is defined in `/etc/skill/si/caplib/siminout.ile`. You can modify this function.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command is unsuccessful.

Examples

```
simin()
```

Related Topics

OSS Functions

simInitControl

```
simInitControl(
    )
    => t / nil
```

Description

Checks whether a file called `control` exists in the simulation run directory.

The following checks are done:

1. If the *simControlFile* variable is `nil`, it locates the files specified by the *simDefaultControl* variable in the `local/si` directory.
2. If not found, it locates the files in the `etc/si` directory using the `prependInstallPath` function and sets *simControlFile* to the resulting full pathname.
3. Next, the function copies the file specified by the *simControlFile* variable to the run directory and names it `control`.

The function is defined in `/etc/skill/si/caplib/init.ile`. You can modify this function.

Arguments

None

Value Returned

`t`	The control file is found.
`nil`	The control file was not found.

Examples

```
simInitControl()
```

Related Topics

OSS Functions

simInitEnv

```
simInitEnv(
     )
     => t / nil
```

Description

Initializes the simulation environment within the Cadence graphics environment.

In addition to defining the SKILL environment needed for SE and the target application, the run directory is created and initialized as needed, using the `simInitRunDir` function. When `simInitEnv` is invoked, a form appears, prompting you for the simulation run directory name. You can specify either a relative or a full file system path name. If you specify a relative path name, the name is prepended with the full file system path name to the directory from which the program was invoked. If the specified run directory does not exist, a second form appears and prompts you for the following:

- Simulation run directory
- Simulator name
- Library name
- Cell name
- View name

Using this information, the run directory is created and initialized. If the run directory already exists, the environment specified within it is restored, and the second form is not displayed.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simInitEnv( )
```

Related Topics

OSS Functions

simInitEnvWithArgs

```
simInitEnvWithArgs(
     t_runDirName
     t_libName | nil
     t_cellName | nil
     t_viewName | nil
     t_simulatorName | nil
     g_forceInit
     )
     => t / nil
```

Description

Initializes the simulation environment within the Cadence graphics environment. In addition to defining the SKILL environment needed for SE and the target application, the run directory is created and initialized as needed.

Arguments

t_runDirName	The name of the simulation run directory to use.
t_libName	The name of the library containing the top-level cellview of the design to be analyzed.
t_cellName	The cell name of the top-level cellview of the design to be analyzed.
t_viewName	The view name of the top-level cellview of the design to be analyzed.
t_simulatorName	The name of the analysis tool to use.
g_forceInit	If the function returns `t`, and the specified run directory exists but is not initialized, the run directory is initialized with the specified parameters. If the function returns `nil`, there is an error.

The arguments can be used to overwrite the simulation environment variables. If the run directory exists, all arguments except for *t_runDirName* can be `nil`. The contents of the run directory are then used to initialize the environment.

The `si.env` file is used to initialize a run directory for storing the values of the simulation environment variables. The arguments above can be used to change the simulation environment variables before the run directory is initialized. The arguments are processed according to the following rules:

- If the specified run directory does not already exist, the argument is not used, and the current graphics environment library path is stored in the new run directory.
- If *`t_simulatorName`* is not `nil`, assign it to the *`simSimulator`* environment variable.
- If *`t_libName`* is not `nil`, assign it to the *`simLibName`* environment variable.
- If *`t_cellName`* and *`t_viewName`* are not `nil`, assign them to the *`simCellName`* and *`simViewName`* environment variables, respectively.

Therefore, the *`t_libName`* argument cannot be `nil` if the *`t_cellName`* or *`t_viewName`* arguments are used (not `nil`).

If the `si.env` file exists in the run directory, it is loaded first, followed by the steps above. This will override what is stored in the run directory. Otherwise, the steps above are applied first, and the `si.env` file is created. Therefore, it is important that you determine the existence of the run directory before calling the `simInitEnvWithArgs` function. Parameters should be passed in if the directory does not exist, but should normally not be passed if it does exist.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simInitEnvWithArgs( "/mnt/dave/chip1/spice.run1"
"myLib" "fast_mux" "schematic"
"spice" nil )
```

```
simInitEnvWithArgs(    "/mnt/user1/chip1/spice.run1"
nil nil nil nil nil )
```

Related Topics

OSS Functions

simInitRaw

```
simInitRaw(
      )
      => t / nil
```

Description

Creates the `raw` directory in the simulation run directory that stores the waveform file.

The function is defined in `/etc/skill/si/caplib/init.ile`. You can modify this function.

Note: WSF is no longer supported, therefore, this function will be removed in the future release of the product.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simInitRaw()
```

Related Topics

OSS Functions

simInitRunDir

```
simInitRunDir(
     )
     => t / nil
```

Description

Executes the list of functions specified by the variable *`simInitRunActions`*. If this variable is not set then `simInitControl` and `simInitRaw` are the default list of functions to be executed in this order, which initializes a simulation run directory.

If these functions do not provide the correct sequence of steps to be performed for a particular simulator, the variable can be set in the simulator-specific file.

If the variable *`simInitRunActions`* has not already been set, it is set as follows:

```
'(simInitControl ( )
   simInitRaw( )
)
```

The file is stored in the directory `local/si/caplib` and has the same name as the simulator with the `.ile` suffix, either inside the file outside of any function, or in the function of the same name as the simulator.

Next, each function specified in the list *`simInitRunActions`* is called in the given order. As soon as one of these functions returns a value other than `t`, the initialization is stopped, and `nil` is returned. If all the functions return `t`, `t` is returned.

The function is defined in `/etc/skill/si/caplib/init.ile`.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simInitRunDir()
```

Related Topics

OSS Functions

simInitSimulator

```
simInitSimulator(
     )
     => t / nil
```

Description

Calls the function with the same name as the simulator name specified by the `simSimulator` variable, for example, `silos`. This function must be defined in a file with the same name with either the `.il` or the `.ile` suffix added and stored in the `local/si/caplib` directory.

The function with the same name as the simulator must set the following variables, as well as any simulator-specific variables.

simDefaultControl
simViewList
simStopList
simNlpGlobalViewName
simSedFile
simCommand

The function is defined in `/etc/skill/si/caplib/init.ile`. You can modify this function.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simInitSimulator()
```

Related Topics

OSS Functions

simInstCdsNameExtName

```
simInstCdsNameExtName(
     t_cdsInstName
     )
     => t_string
```

Description

Accepts the user-assigned instance name as an argument and returns the netlister-assigned instance name for *cdsInstName* as it appears in the netlist. The exact name returned depends on the design.

The function is defined in `/bin/si` and also in the Cadence graphics program. You cannot modify this function.

Arguments

t_cdsInstName	User-assigned instance name.

Value Returned

t_string	The netlister-assigned instance name as it appears in the netlist.

Examples

```
simInstCdsNameExtName( "/adder1/and1" )
=>  "I34"
```

Related Topics

OSS Functions

simInstExtNameCdsName

```
simInstExtNameCdsName(
     t_extInstName
     )
     => t_string
```

Description

Accepts the netlister-assigned instance name as an argument `extInstName` and returns the user-assigned instance name as it appears in the schematic.

The function is defined in `/bin/si` and also in the Cadence graphics program. You cannot modify this function.

Arguments

t_extInstName	Netlister-assigned instance name.

Value Returned

t_string	The netlister-assigned instance name *extInstName* specified as an argument. The exact name returned depends on the design.

Examples

```
simInstExtNameCdsName( "I34" )
=>  "/adder1/and1"
```

Related Topics

OSS Functions

simInWithArgs

```
simInWithArgs(
     l_fileList
     t_runDirName
     t_libName
     t_cellName
     t_viewName
     )
     => t / nil
```

Description

Translates the designer-assigned names for nets and instances in the input files into the corresponding netlister-assigned names and creates the files for input into the simulator.

When the input arguments are processed the following conditions apply:

- If *`libName`* is `nil`, then the *`simLibName`*, *`simCellName`*, *`simViewName`* environment variables are used.
- Otherwise, if *`cellName`* is `nil`, then the *`simCellName`* environment variable is used.
- If *`viewName`* is `nil`, then the *`simViewName`* environment variable is used.

Therefore, the *`libName`* argument must not be `nil` if the *`cellName`* and *`viewName`* arguments are used.

All text in the `inputFileName` file is copied to the `outputFileName` file, unless it is surrounded by square brackets (`[ ]`). The opening square bracket (`[`) specifies that the following text up to the closing square bracket (`]`) must be interpreted. The entire expression is replaced by the interpreted value.

One of the command characters below must follow the opening square bracket (`[`):

`#`	`[#netname]`	Replaces the `[#netname]` expression with the netlister-assigned node name for `netname`.
`$`	`[$instname]`	Replaces the `[$instname]` expression with the netlister-assigned instance name for `instname`.
`!`	`[!filename]`	Replaces the `[!filename]` expression with the contents of the filename file. If you use a relative pathname, the system looks for the filename file in the simulation run directory. To access a file outside the simulation run directory, use a full file system pathname. If the filename file does not exist, an error is generated.

`?`	`[?filename]`	Replaces the `[?filename]` expression with the contents of the `filename` file. If you use a relative pathname, the system looks for the `filename` file in the simulation run directory. To access a file outside the simulation run directory, use a full file system pathname. If the `filename` file does not exist, *no* error is generated.
`n!`	`[n!filename]`	Same as `[! filename]`, except that the contents of the new file are not parsed, and square-bracketed expressions are not interpreted.
`n?`	`[n?filename]`	Same as `[? filename]`, except that the contents of the new file are not parsed, and square-bracketed expressions are not interpreted.

The function is defined in `/bin/si`. You cannot modify this function.

Arguments

`l_filelist`	A list of lists for the input and output file names.
`t_runDirName`	The name of the simulation run directory. It must be a string name.
`t_libName`	The name of the library containing the top-level cellview of the design.
`t_cellName`	The cell name of the top-level cellview.
`t_viewName`	The view name of the top-level cellview.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simInWithArgs( list("control" "si.inp") simRunDir
       simLibName simCellName simViewName )

simInWithArgs( '((infile1 outfile1)
   (infile2 outfile2))
   simRunDir simLibName simCellName
   simViewName )
```

Related Topics

OSS Functions

simJobMonitor

```
simJobMonitor(
      )
      => t / nil
```

Description

Displays a form listing the analysis jobs invoked in the background using the `simRunNetAndSim` and `simRunNetAndSimWithArgs` functions. The analysis job is listed on the form, along with its current status, time of invocation, and execution priority. Using this form, you can view the run log of a job, terminate the execution of an active job, suspend the execution of an active job, change the execution priority of a job, or delete a job from the form.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simJobMonitor( )
```

Related Topics

OSS Functions

simLoadNetlisterFiles

```
simLoadNetlisterFiles(
     t_fileName
     )
     => t / nil
```

Description

Loads the file specified as an argument from the `local/fnl` directory. If the tool does not find the file in the `local/fnl` directory, it searches in the `etc/skill/fnl` directory in the install hierarchy. If it does not exist, no error is generated.

The function is defined in `/etc/skill/si/simcap.ile`. You can modify this function.

Arguments

t_fileName	Name of the file to be loaded.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simLoadNetlisterFiles( "silos.ile" )
```

Related Topics

OSS Functions

simLoadSimulatorFiles

```
simLoadSimulatorFiles(
    )
    => t / nil
```

Description

Loads the simulator-specific file from the `local/si/caplib` directory. If it does not find the file in `local/si/caplib`, it searches `etc/skill/si/caplib` directory in the install hierarchy. If the file does not exist, no error is generated.

The function is defined in `/etc/skill/si/simcap.ile`. You can modify this function.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simLoadSimulatorFiles()
```

Related Topics

OSS Functions

simNetCdsNameExtName

```
simNetCdsNameExtName(
     t_cdsNetName
     )
     => t_string
```

Description

Accepts the user-assigned net name as an argument for *cdsNetName* and returns the netlister-assigned net name as it appears in the netlist.

The function is defined in `/bin/si`. You cannot modify this function.

Arguments

t_cdsNetName	User-assigned net name in the schematic.

Value Returned

t_string	The netlister-assigned net name as it appears in the netlist. The exact name returned depends on the design.

Examples

```
simNetCdsNameExtName( "/adder1/in" )
=> "N34"
```

Related Topics

OSS Functions

simNetExtNameCdsName

```
simNetExtNameCdsName(
     t_extNetName
     )
     => t_string
```

Description

Accepts the netlister-assigned net name as an argument and returns the user-assigned net name for *extNetName* as it appears in the schematic.

The function is defined in `/bin/si` and also in the Cadence graphics program. You cannot modify this function.

Arguments

t_extNetName	Netlister-assigned net name

Value Returned

t_string	The user-assigned net name as it appears in the schematic. The exact name returned depends on the design.

Examples

```
simNetExtNameCdsName( "N34" )
=> "/adder1/in"
```

Related Topics

OSS Functions

simNetlistWithArgs

```
simNetlistWithArgs(
     t_simLibName
     t_simCellName
     t_simViewName
     t_simRunDir
     t_simNlpGlobalLibName
     t_simNlpGlobalCellName
     t_simNlpGlobalViewName
     l_simViewList
     l_simStopList
     t_simGlobalErrorFileName
     t_simProbeFileName
     t_simSimulator
     f_simTimeUnit
     f_simCapUnit
     t_simNetlistFileName
     t_simFlatViewTypeName
     )
     => t / nil
```

Description

Generates a flattened description of the design hierarchy specified by *simLibName*, *simCellName*, and *simViewName* in the syntax described by the global formatting properties in *simNlpGlobalLibName*, *simNlpGlobalCellName* and *simNlpGlobalViewName*.

In addition to specifying the input parameters, you must set the following variables from the SE global environment before calling this function:

simNetNamePrefix
simInstNamePrefix
simModelNamePrefix

These variables are used by the `simNetlistWithArgs` SKILL function.

The variables generate unique names during netlisting. The netlister uses these string prefixes and adds a unique number as a suffix to each of them to create a unique name. If you set these variables to `nil`, the netlister generates the unique number as output but does not add a string prefix to the number.

The function is defined in `/bin/si` and also in the Cadence graphics program. You cannot modify this function.

Arguments

t_simLibName The name of the library containing the top-level cellview of the design.

t_simCellName The cell name of the top-level cellview (design) to be netlisted. For example:

`"alu"`

This cell name must be the *simCellName* global SE variable.

t_simViewName The view name of the top-level cellview of the design.

t_simRunDir The full file system path name to the simulation run directory. This path name must be the same as the *simRunDir* global SE variable. For example:

`"/mnt2/user1/simulations/silos1"`

t_simNlpGlobalLibName

The name of the library which contains the global cellview that defines the global format strings. The format strings define the netlist syntax. This library name must be the same as the value of the *simNlpGlobalLibName* global SE variable. For example:

`"basic"`

t_simNlpGlobalCellName

The cell name of the global cellview which contains the global format strings that define the netlist syntax. This cell name must be the same as the value of the *simNlpGlobalCellName* global SE variable. For example:

`"nlpglobals"`

t_simNlpGlobalViewName

The view name of the global cellview which contains the global format strings that define the netlist syntax. This view name must be the same as the value of the *simNlpGlobalViewName* global SE variable. For example:

`"verilog"`

l_simViewList The view switch list that determines which view to switch into when a cell has been found. This list must be the same as the value of the *simViewList* global SE variable. For example:

`list("verilog" "schematic")`

l_simStopList The stopping view list that determines when expansion of the hierarchy is halted. This list must be the same as the value of the *simStopList* global SE variable. For example:

```
list("verilog")
```

t_simGlobalErrorFile

The name of the file for storing global error messages.

t_simProbeFileName

The name of the probe file for storing errors, usually `probe.err`.

t_simSimulator The simulator for which the netlist syntax is intended. This simulator must be the same as the value of the *simSimulator* SE global environment variable. For example:

```
"verilog"
```

f_simTimeUnit The floating-point time unit factor. For example, `1.0e-9`.

f_simCapUnit The floating-point capacitance unit factor. For example, `1.0e-15`.

t_simNetlistFileName

The name of the file in which the netlist is stored. For example:

```
"/mnt/user1/simulations/silos1/netlist"
```

Value Returned

`t`	The netlist process does not detect any errors.
`nil`	The netlist process detected an error.

Examples

```
simNetlistWithArgs(
   simLibName
   simCellName
   simViewName
   simRunDir
   simNlpGlobalLibName
   simNlpGlobalCellName
   simNlpGlobalViewName
   simViewList
   simStopList
   "global.err"
   "probe.err"
   simSimulator
   float(simTimeUnit)
   float(simCapUnit)
```

```
    "/mnt/user1/simulations/silos1/netlist
)
```

Related Topics

OSS Functions

simNoNetlist

```
simNoNetlist(
      )
      => t / nil
```

Description

Sets the *`simNoNetlist`* variable to `t` to specify that no netlist is generated, and then callsthe `sim` function to perform a complete simulation, except for netlist generation.

The function is defined in `/etc/skill/si/caplib/simulate.ile`. You can modify this function.

Value Returned

`t`	The sim function completed running the simulation.
`nil`	The command was unsuccessful. In this case, an error message is printed out.

Examples

```
simNoNetlist()
```

Related Topics

OSS Functions

simout

```
simout(
    )
    => t / nil
```

Description

Translates the netlister-assigned names for nets and instances in the `simout.tmp` file to the corresponding user-assigned names and produces the `si.out` file. The `simout.tmp` file is normally the simulator text output. A message is printed stating that `simout` is being run, and then the `simOutWithArgs` function is called with the correct arguments to do the translation.

The following variables are used as arguments to the `simOutWithArgs` function. You must set them correctly before calling this function:

simSedFile
simRunDir
simLibName
simCellName
simViewName

All text in the `simout.tmp` file is copied to the `si.out` file, unless the text is surrounded by square brackets (`[ ]`). The opening square bracket (`[`) signals that the following text up to the closing square bracket (`]`) is to be interpreted. The entire expression is replaced by the resulting interpreted value. Following the opening square bracket (`[`) should be one of these command characters:

`#`	`[#netname]`	Replaces the `[#netname]` expression with the user-assigned net name for `netname`.
`$`	`[$instname}`	Replaces the `[$instname]` expression with the user-assigned instance name for `instname`.

Because names requiring translation are not output by the simulator surrounded by square brackets (`[ ]`), the interface developer must provide a `sed` input script for each simulator that surrounds each name requiring translation with square brackets (`[ ]`) and inserts the correct command character after the opening square bracket (`[`). For example, if the name `netname` needs to be translated back to the user-assigned name for the net, the `sed` script needs to replace the word `netname` with `[#netname]` so that this function translates it.

The function is defined in `/etc/skill/si/caplib/siminout.ile`. You can modify this function.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simout()
```

Related Topics

OSS Functions

simOutWithArgs

```
simOutWithArgs(
     l_fileList
     t_runDirName
     t_libName
     t_cellName
     t_viewName
     )
     => t / nil
```

Description

Translates the netlister-assigned names for nets and instances in the input files specified in `fileList` to the corresponding user-assigned names, and produces the output files specified in `fileList`.

Because names requiring translation are not output by the simulator surrounded by square brackets (`[ ]`), you must provide a `sed` input script for each simulator that surrounds each name to be translated with square brackets (`[ ]`). You must also insert the correct command character after the opening square bracket (`[`).

When the input arguments are processed the following conditions apply:

- If *`libName`* is `nil`, then the *`simLibName`*, *`simCellName`*, *`simViewName`* environment variables are used.
- Otherwise, if *`cellName`* is `nil`, then the *`simCellName`* environment variable is used.
- If *`viewName`* is `nil`, then the *`simViewName`* environment variable is used.

Therefore, the *`libName`* argument must not be `nil` if the *`cellName`* and *`viewName`* arguments are used (not `nil`).

All text in the `inputFileName` file is copied to the `outputFileName` file, unless it is surrounded by square brackets (`[ ]`). The opening square bracket (`[`) specifies that the following text up to the closing square bracket (`]`) must be interpreted. The entire expression is replaced by the interpreted value.

One of the command characters below must follow the opening square bracket (`[`):

`#`	`[#netname]`	Replaces the `[#netname]` expression with the user-assigned net name for `netname`.
`$`	`[$instname]`	Replaces the `[$instname]` expression with the user-assigned instance name for `instname`.

The function is defined in `/bin/si` and also in the Cadence graphics program. You cannot modify this function.

Arguments

l_fileList	A list of lists for the input, output, and `sed` file names. The `sed` file name can be `nil` and a single list can be used for only one input, one output, and one `sed` file.
t_runDirName	The name of the simulation run directory. This must be a string name.
t_libName	The library name of the top-level design.
t_cellName	The name of the top-level cell.
t_viewName	The name of the top-level view.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simOutWithArgs(
   list ( infile outfile simSedFile)
   simRunDir
   simLibName
   simCellName
   simViewName
)
```

```
simOutWithArgs(
   '(( infile1 outfile1 simSedFile)
   ( infile2 outfile2 simSedFile))
   simRunDir
   simLibName
   simCellName
   simViewName
)
```

Related Topics

OSS Functions

simPostNameConvert

```
simPostNameConvert(
     )
     => t / nil
```

Description

Clears all allocated storage and marks the map structure as invalid. Since this function can be called using SKILL, the parameter has to be maintained as a list.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simPostNameConvert()
```

Related Topics

OSS Functions

simPostNetlistTriggerList

```
simPostNetlistTriggerList(
     )
     => l_triggerFunc / nil
```

Description

Returns the list of functions registered through `simRegPostNetlistTrigger` when this function is specified in the `.simrc` file or the CIW. For backward compatibility, `hnlPostNetlistTriggerList` can be used as an alias for `simPostNetlistTriggerList`.

Arguments

None

Value Returned

l_triggerFunc	List of functions registered using `simRegPostNetlistTrigger`.
`nil`	There are no functions registered using `simPostNetlistTriggerList`.

Examples

```
simPostNetlistTriggerList()
```

Related Topics

OSS Functions

simPreNameConvert

```
simPreNameConvert(
     )
     => t / nil
```

Description

Sets up the variables needed for name conversion routines. If no arguments are passed, it does the setup by doing a lookup of the simulator environment settings.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was not successful.

Examples

```
simPreNameConvert()
```

Related Topics

OSS Functions

simPreNetlistTriggerList

```
simPreNetlistTriggerList(
    )
    => l_triggerFunc / nil
```

Description

Returns the list of functions registered through `simRegPreNetlistTrigger` when this function is specified in the `.simrc` file or the CIW. For backward compatibility, `hnlPreNetlistTriggerList` can be used as an alias for `simPreNetlistTriggerList`.

Arguments

None

Value Returned

l_triggerFunc	List of functions registered using `simRegPostNetlistTrigger`.
`nil`	There are no functions registered using `simPostNetlistTriggerList`.

Examples

```
simPreNetlistTriggerList()
```

Related Topics

OSS Functions

simPrintEnvironment

```
simPrintEnvironment(
    )
    => t / nil
```

Description

Writes the primary simulation control variables and their values to the `si.env` file in the simulation run directory.

The following variables must be defined and are written to the file:

simLibName
simCellName
simViewName
simSimulator
simNotIncremental
simReNetlistAll

In addition, the following variables are written to the `si.env` file if they are defined:

simViewList
simStopList
simOtherInfo
simHost

The following variables are written to the `si.env` file if they are not `nil`:

simNetlistHier
simHostDiffers
simNoSimDiff

The variables specified by the *simSimulatorSaveVars* variable are also written to the file.

The function is defined in `/etc/skill/si/caplib/init.ile`. You can modify this function.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful. In this case, an error message is also printed.

Examples

```
simPrintEnvironment()
```

Related Topics

OSS Functions

simPrintError

```
simPrintError(
     t_text
     )
     => t
```

Description

Prints the `text` argument to the *`stderr`* port if executed in the Cadence nongraphic environment. If the function is called from within the Cadence graphics environment, the output is written to the CIW window.

You can use this function instead of the following `fprintf` function:

```
fprintf( stderr text )
```

The function is defined in `/bin/si` and also in the Cadence graphics program. You cannot modify this function.

Arguments

`t_text`	A text string.

Value Returned

`t`	The command run was completed.

Examples

```
simPrintError( "Can't open file simout.tmp\n" )
```

Related Topics

OSS Functions

simPrintErrorLine

```
simPrintErrorLine(
     g_listArg
     )
     => t
```

Description

Prints the errors for a cell and writes them out to a file.

Arguments

g_listArg	The list contains the following components: ■ *t_libName*: Name of the library containing the design. ■ *t_cellName*: Cell name of the design. ■ *t_viewName*: View name of the design. ■ *t_fileName*: `t` for `stdout` , otherwise, the file name.

Value Returned

`t`	The errors were printed.

Examples

```
simPrintErrorLine(l_listArg)
```

Related Topics

OSS Functions

simPrintMessage

```
simPrintMessage(
     t_text
     )
     => t
```

Description

Prints the `text` argument to the `stdout` port if executed in the Cadence non-graphic environment. If the function is called from within the Cadence graphics environment, the output is written to the CIW window.

You can use this function instead of the following `fprintf` function:

```
fprintf( stdout text )
```

The function is defined in `/bin/si` and also in the Cadence graphics program. You cannot modify this function.

Arguments

t_text	A text string.

Value Returned

`t`	The command is successful.

Examples

```
simPrintMessage( "Can’t open file simout.tmp\n" )
```

Related Topics

OSS Functions

simPrintTermList

```
simPrintTermList(
     t_libName
     t_cellName
     t_viewName
     t_fileName
     )
     => t
```

Description

Lists the terminals for a cell and writes them out to a file.

Arguments

t_libName	Name of the library containing the design.
t_cellName	Cell name of the design.
t_viewName	View name of the design.
t_fileName	`t` for `stdout` , otherwise, the file name.

Value Returned

`t`	The command run was completed.

Examples

```
simPrintTermList("basic" "VDD" "schematic" "fileName")
```

Related Topics

OSS Functions

simReadNetCapFile

```
simReadNetCapFile(
     t_filename
     )
     => t / nil
```

Description

Reads in the net capacitance file and initializes the *simAllNets* global variable. This contains the list of all nets in the design and their associated capacitance values, which are present in the capacitance file.

Arguments

t_filename	The filename, if the file is in the current directory or the full path to the file which contains the information about the nets and their associated capacitance. The filename or the full path to the file must be valid SKILL strings. The default value is `<rundir>/lperun/sim.cap`.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simReadNetCapFile( "./mydir/dir2/sim.cap" )
```

Related Topics

OSS Functions

simRegPostNetlistTrigger

```
simRegPostNetlistTrigger(
     S_triggerFunc
     )
     => t / nil
```

Description

Registers a function to be called after the netlist has been generated. This is applicable for both hierarchical and flat netlisting. For backward compatibility, `hnlRegPostNetlistTrigger` can be used as an alias for `simRegPostNetlistTrigger`.

The `simRegPostNetlistTrigger` function can be called in the `.simrc` or the `libinit.il` file and *`S_triggerFunc`* can be defined in any one of these files.

Arguments

`S_triggerFunc`	Name of SKILL function symbol.

Value Returned

`t`	The function was registered.
`nil`	The function could not be registered.

Examples

```
procedure(postNetlistTrigger()
   printf("\n\npostNetlistTrigger is called\n\n")
)
simRegPostNetlistTrigger('postNetlistTrigger)
```

Related Topics

OSS Functions

simRegPreNetlistTrigger

```
simRegPreNetlistTrigger(
     S_triggerFunc
     )
     => t / nil
```

Description

Registers a function to be called before the netlisting begins. This is applicable for both hierarchical and flat netlisting. For backward compatibility, `hnlRegPreNetlistTrigger` can be used as an alias for `simRegPreNetlistTrigger`.

The `simRegPreNetlistTrigger` function is called after the `.simrc` is loaded. It must be called in the `.simrc` file and *`S_triggerFunc`* must be defined in the `.simrc` file.

Arguments

`S_triggerFunc`	Name of SKILL function symbol.

Value Returned

`t`	The function was registered.
`nil`	The function could not be registered.

Examples

The following example shows how to set `useMfactorToIterateInstances` using `simRegPreNetlistTrigger`:

```
procedure(setUseMfactorFlag()
  fnlMFactorPropertyName="m"
  useMfactorToIterateInstances = "OnSubckt"
)
simRegPreNetlistTrigger('setUseMfactorFlag)
```

Related Topics

OSS Functions

simRunDirInfile

```
simRunDirInfile(
     t_fileName
     )
     => t / nil
```

Description

Opens the file specified using the *fileName* argument in the simulation run directory for reading. It creates a full file system path name to the file in the simulation run directory and then passes the file as an argument to the SKILL `infile` function. The function returns a SKILL `port`.

The function is defined in `/etc/skill/si/caplib/util.ile`. You can modify this function.

Arguments

t_fileName	Name of the file opened for reading.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simRunDirInfile( "tmp.in" )
```

Related Topics

OSS Functions

simRunDirLoad

```
simRunDirLoad(
     t_fileName
     )
     => t / nil
```

Description

Loads the file specified using the *t_fileName* argument in the simulation run directory. It creates a full file system path to the file in the simulation run directory and then passes the file as an argument to the SKILL `load` function.

The function is defined in `/etc/skill/si/caplib/util.ile`. You can modify this function.

Arguments

t_fileName	Name of the file to be loaded.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simRunDirLoad( "comp.env" )
```

Related Topics

OSS Functions

simRunDirOutfile

```
simRunDirOutfile(
     t_fileName
     )
     => t / nil
```

Description

Opens the file specified using the `fileName` argument in the simulation run directory for writing. It creates a full file system path name to the file in the simulation run directory and then passes the file as an argument to the SKILL `outfile` function. The function returns a SKILL `port`.

The function is defined in `/etc/skill/si/caplib/util.ile`. You can modify this function.

Arguments

t_fileName	Name of the file to be opened.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simRunDirOutfile( "tmp.out" )
```

Related Topics

OSS Functions

simRunNetAndSim

```
simRunNetAndSim(
    )
    => t / nil
```

Description

Starts an analysis job in either foreground or background mode.

The simulation environment must be initialized before `simRunNetAndSim` is called. That is, the `simInitEnv` or `simInitEnvWithArgs` function must have been called. When invoked, `simRunNetAndSim` displays a form, prompting for the following:

- Simulation run directory (read only field)
- Library name
- Cell name
- View name
- Simulator name
- Whether to run the netlister
- Whether to run the simulator
- Whether to run in background/foreground
- Priority of a background simulation (read only if foreground)

The `simRunNetAndSim` function applies the form settings and invokes the simulation.

Arguments

None

Value Returned

`t`	The form prompting for run options was successfully displayed.
`nil`	The command was unsuccessful.

Examples

```
simRunNetAndSim( )
```

Related Topics

OSS Functions

simRunNetAndSimWithArgs

```
simRunNetAndSimWithArgs(
     t_libName | nil
     t_cellName | nil
     t_viewName | nil
     t_simulatorName | nil
     g_doNetlist
     g_doSimulation
     g_runBackground
     x_jobPriority
     )
     => t / nil
```

Description

Starts an analysis job in either foreground or background mode.

The simulation environment must be initialized before the `simRunNetAndSim` function is called. Specifically, you must call the `simInitEnv` or `simInitEnvWithArgs` function before calling `simRunNetAndSim`. The *simRunDir* global variable specifies the current run directory. It is set when the simulation environment is initialized.

Arguments

t_libName	The name of the library containing the top-level cellview of the design to be analyzed.
t_cellName	The cell name of the top-level cellview of the design to be analyzed.
t_viewName	The view name of the top-level cellview of the design to be analyzed.
t_simulatorName	The name of the simulator that runs the analysis.
g_doNetlist	Boolean value, which if set to `t`, the netlist for the design is generated.
g_doSimulation	Boolean value, which if set to `t`, the design is simulated. The simulator and the name translation functions are invoked. This invokes the same steps and functions as the `sim` function.
g_runBackground	Boolean value, which if set to `t`, the background process invokes the `bin/si` program to perform the simulation.

x_jobPriority	Priority of the background job (0 to 20). This is the UNIX priority that invokes the process; therefore, the lower the number, the higher the priority.

The *t_libName*, *t_cellName*, *t_viewName*, and *t_simulatorName* arguments overwrite the corresponding simulation environment variables. Specifying these arguments redefines the global environment and changes the values in the simulation run directory. If you want to use the current global environment, use `nil` as the value for these parameters. The arguments are processed according to the following rules:

- If *t_libName* is not `nil`, assign it to the *simLibName* environment variable.
- If *t_simulatorName* is not `nil`, assign it to the *simSimulator* environment variable.
- If *t_cellName* or *t_viewName* are not `nil`, assign them to the *simCellName* and *simViewName* environment variables, respectively.

Therefore, the *t_libName* argument must not be `nil` if the *t_cellName* and/or *t_viewName* arguments are used (not `nil`).

Value Returned

`t`	The background process was successfully invoked. If the analysis is run in the foreground, `t` is returned if the analysis completed. A return value of `t` does *not* necessarily mean that the analysis was completed successfully.
`nil`	The background process or foreground analysis failed.

Examples

```
simRunNetAndSimWithArgs( "myLib" "fast_mux"
"schematic" "spice" t t t 10)
   simRunNetAndSimWithArgs( nil nil nil nil
   t t t 10)
```

Related Topics

OSS Functions

simRunNetAndSimWithCmd

```
simRunNetAndSimWithCmd(
     t_libName | nil
     t_cellName | nil
     t_viewName | nil
     t_simulatorName | nil
     t_cmdToBeExecuted
     g_runBackground
     x_jobPriority
     )
     => t / nil
```

Description

Runs a command that you specify in either foreground or background mode. The simulation environment *must be* initialized before `simRunNetAndSimWithCmd` is called. The *simRunDir* global variable specifies the current run directory. It is set when the simulation environment is initialized.

Arguments

t_libName	The name of the library containing the top-level cellview of the design to be analyzed.
t_cellName	The cell name of the top-level cellview of the design to be analyzed.
t_viewName	The view name of the top-level cellview of the design to be analyzed.
t_simulatorName	The name of the analysis tool.
t_cmdToBeExecuted	The string specifying the name of the function to be executed.
g_runBackground	Boolean value, which if set to `t`, the background process performs the specified command.
x_jobPriority	Priority of the background job.

The *t_libName*, *t_cellName*, *t_viewName* and *t_simulatorName* arguments overwrite the corresponding simulation environment variables. Specifying these arguments redefines the global environment *and* changes the values in the simulation run directory. If you want to use the current global environment, use `nil` as the value for these parameters. The arguments are processed according to the following rules:

- If *t_libName* is not `nil`, assign it to the *simLibName* environment variable.
- If *t_simulatorName* is not `nil`, assign it to the *simSimulator* environment variable.
- If *t_cellName* and/or *t_viewName* are not `nil`, assign them to the *simCellName* and *simViewName* environment variables, respectively.
- Thus, the *t_libName* argument must not be `nil` if the *t_cellName* and *t_viewName* arguments are used (not `nil`).

Value Returned

`t`	The background process was successfully invoked. If the analysis is run in the foreground, `t` is returned if the analysis completed. A return value of `t` does not necessarily mean that the analysis was completed successfully.
`nil`	The background process or foreground analysis failed.

Examples

```
simRunNetAndSimWithCmd( "myLib"
   "fast_mux" "schematic"
   "spice" "simin" t 10
   )

simRunNetAndSimWithCmd( nil nil nil nil
   "my_SKILL_function" nil 10
   )
```

Related Topics

OSS Functions

simSetDef

```
simSetDef(
     g_variableName
     g_value
     )
     => t
```

Description

Sets the *g_variableName* variable to *g_value* only if the variable is not yet set or if the symbol *variableName* evaluates to `nil`. If the variable was set prior to calling this function, and the *simGenWarnings* variable is not `nil`, a warning message is printed to `stdout` that the value of the *variableName* variable is overridden.

The function is defined in `/etc/skill/si/caplib/util.ile`. You can modify this function.

Arguments

g_variableName	Name of the SE variable.
g_value	Value to be assigned to the SE variable.

Value Returned

`t`	The command is successful.

Examples

```
simSetDef('simCapUnit 1.0e-15)
```

Related Topics

OSS Functions

simSetDefWithNoWarn

```
simSetDefWithNoWarn(
     g_variableName
     g_value
     )
     => t / nil
```

Description

Sets the *g_variableName* variable to *g_value*, same as the `simSetDef` function, except that no warning is generated if *variableName* is already set.

The function is defined in `/etc/skill/si/caplib/util.ile`. You can modify this function.

Arguments

g_variableName	Name of the SE variable.
g_value	Value to be assigned to the SE variable.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simSetDefWithNoWarn('simCapUnit 1.0e-15)
```

Related Topics

OSS Functions

simStringsToList

```
simStringsToList(
    t_stringArg
    )
    => list / nil
```

Description

Accepts a string of names separated by blanks and tab characters as an argument and returns a list of strings.

The function is defined in `/bin/si`. You cannot modify this function.

Arguments

t_stringArg	String of names separated by blanks and tab characters.

Value Returned

list	A list of strings.
nil	The command was unsuccessful.

Examples

```
simStringsToList( "@ . /cds/etc/sdalib/schema" )
returns:
'("@" "." "/cds/etc/sdalib/schema")
```

Related Topics

OSS Functions

simSubProbeCapByName

```
simSubProbeCapByName(
     t_netName
     )
```

Description

Displays net capacitance by name.

Arguments

t_netName	String value to be given to functions, which represents the name of the net for which the capacitance value is required. If the net name is valid, the capacitance value will be displayed on the net, else capacitance values for all the nets in the current design window will not be displayed.

Value Returned

None

Examples

```
simSubProbeCapByName( "net16" )
```

Related Topics

OSS Functions

simSubProbeCapByScreen

```
simSubProbeCapByScreen(
      )
```

Description

Displays all the net capacitance on the current screen.

Arguments

None

Value returned

None

Examples

```
simSubProbeCapByScreen()
```

Related Topics

OSS Functions

simVertToHoriz

```
simVertToHoriz(
     t_inputFileName
     t_outputFileName
     )
     => t / nil
```

Description

Reads in the `inputFileName` file, converts names in vertical table headers to horizontal, and copies the rest of the file.

For example, if the input lines in the *`t_inputFileName`* file are as follows:

```
N N
4N4
234
6 7
```

they are printed in the *`t_outputFileName`* output file as shown below:

```
N 4 2 6
| N 3 |
N 4 4 7
```

The function is defined in `/bin/si` and also in the Cadence graphics program. You cannot modify this function.

Note: This function is used only to generate SILOS and System HILO™ output files.

Arguments

`t_inputFileName`

Name of input file containing names for conversion.

`t_outputFileName`

Name of output file containing converted names.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simVertToHoriz( "simout2.tmp" "simout.tmp" )
```

Related Topics

OSS Functions

simViewFileWithArgs

```
simViewFileWithArgs(
     t_fileName
     l_windowSize
     [ t_windowTitle ]
     )
     => winID / nil
```

Description

Calls the `view` function with the file name as the first argument and the window size as the second argument. The `view` function creates a `viewFile` window where the file text is displayed with the banner as the value supplied through the optional *`windowTitle`* argument.

Arguments

`t_fileName`	Name of the file.
`l_windowSize`	Size of the window.
	Specify the list containing the window size and `maxWindowSize`.
`t_windowTitle`	Title of the window.

Value Returned

`winID`	The ID of the window in which the file is displayed.
`nil`	The command was unsuccessful.

Examples

```
simViewFileWithArgs( "/tmp/skill.il" list(680:800 hiGetMaxScreenCoords()) "SKILL
CODE" )
```

Related Topics

OSS Functions

simWaveOpen

```
simWaveOpen(
      )
      => t / nil
```

Description

Invokes a form displaying the current simulation run directory name, and prompts you for the name of the waveform file to display. The default waveform file is `raw/waves` in the simulation run directory. When the file name is specified, a new window is opened, displaying the waveform information.

Note: WSF is no longer supported, therefore, this function will be removed in a future release of the product.

Arguments

None

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Examples

```
simWaveOpen()
```

Related Topics

OSS Functions

3

VHDL Toolbox Functions

VHDL Toolbox is an integrated netlisting and simulation environment that you can use to generate structural VHDL (IEEE93/87) text netlists from hierarchical OpenAccess 2.2 schematics and run simulations in the Cadence NC environment. You can run VHDL netlister in standalone mode using Cadence Simulation Environment (SI) of Open Simulation System (OSS).

This section describes the SKILL functions associated with VHDL Toolbox.

- VHDL Functions

vhdlHiImport	vhdlHiInvokeToolBox	vhdlImport
vhdlPinListToVHDL	vhdlRegisterSimulator	vhdlPinListToVHDL
vosHiDisplayNetlist	vosLaunchIrunSimulation	

- VHDL AMS Functions

vhmsDefaultEdit	vhmsGetCellParameters	vhmsPinListToVHDLAMS
vhmsGetCellParameters	vhmsSymbolToPinListGen	vhmsToPinList
vhmsUpdateCellCDFParams		

Licensing Requirements

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

Related Topics

Introducing the VHDL Toolbox

About the VHDL Integration Environment

vhdlHiImport

```
vhdlHiImport(
     )
     => t / nil
```

Description

Builds and displays the VHDL Import form.

Arguments

None

Value Returned

`t`	The VHDL Import form is displayed.
`nil`	The VHDL Import form could not be displayed.

Examples

The following example displays the VHDL Import form.

```
vhdlHiImport()
=> t
```

Related Topics

VHDL Toolbox Functions

vhdlHiInvokeToolBox

```
vhdlHiInvokeToolBox(
     [ t_position ]
     )
     => t / nil
```

Description

Invokes the VHDL Toolbox window.

Arguments

t_position	The argument position specifies the initial position of toolbox on the screen. This value can be a SKILL list of any two integer elements. If you do not specify any value for this argument, the defaut value, `'(0 0)` is used.

Value Returned

`t`	The VHDL Toolbox window was invoked.
`nil`	The VHDL Toolbox window could not be invoked.

Examples

Displays the toolbox window at the x co-ordinate -9 and y co-ordinate 162 on the screen.

```
vhdlHiInvokeToolBox ('(-9 162))
=> t
```

Displays the toolbox window at the default location i.e. `'(0 0)`.

```
vhdlHiInvokeToolBox()
=> t
```

Related Topics

VHDL Toolbox Functions

vhdlImport

```
vhdlImport(
     t_libName
     l_srcFiles
     t_logName
     l_params
     [ g_runInBackground ]
     [ g_displayResults ]
     )
     => t / nil
```

Description

Runs `vhdlin` to import a list of VHDL source files into the specified library with the given parameters. The parameters are the names of the `vhdlin` parameters, passed in as a disembodied property list. Optionally, it can run `vhdlin` as a background process and display the results interactively.

Arguments

t_libName	Name of the target library where files are imported.
l_srcFiles	List of the VHDL text files to be imported by `vhdlin`.
t_logName	Name of the log file to be generated by `vhdlin`.
l_params	Disembodied Property List (DPL) defining the `vhdlin` parameters, where the members of the DPL match the names of the parameters used by `vhdlin`.
g_runInBackground	Boolean flag, specifying whether `vhdlin` is run in the foreground, blocking the current session, or as a background process. The default is `nil`, which means that `vhdlin` runs in the foreground.
g_displayResults	Boolean flag, specifying whether the results of the `vhdlin` run are displayed interactively using the VHDL Toolbox log or error viewing window. The default is `nil`, which means that `vhdlin` does not display results interactively.

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Related Topics

VHDL Toolbox Functions

vhdlPinListToVHDL

```
vhdlPinListToVHDL(
     t_libName
     t_cellName
     t_viewName
     l_pinList
     )
     => t / nil
```

Description

Allows the generation of VHDL views from an intermediate pin list format.

Arguments

t_libName	The name of the library to store the VHDL cellview.
t_cellName	The cell name of the translated VHDL cellview.
t_viewName	The view name of the translated VHDL cellview.
l_pinList	The pin list to be translated to the VHDL cellview.

The pinList has the following format:

```
( ports (nil name "termName" direction "termDir"
    pins ( (name "pinName" accessDir "pinAccessDir"
            pinPropName pinPropValue ...)
            .   .    .    .    .    .    .    .    .
            (name "pinName" accessDir "pinAccessDir"
               pinPropName pinPropValue ...)

                prop (termPropName termPropValue ...)

            .    .    .    .    .    .    .    .    .    .    .    .    .
         .    .    .    .    .    .    .    .    .    .    .    .    .
         .    .    .    .    .    .    .    .    .    .    .    .    .
         (nil name "termName" direction "termDir"
            pins ( (name "pinName" accessDir "pinAccessDir"
                     pinPropName pinPropValue ...)
                      .    .    .    .    .    .    .    .
                     (name "pinName" accessDir "pinAccessDir"
                     pinPropName pinPropValue ...)

            prop (termPropName termPropValue ...)

         prop (nil cvPropName cvPropValue ...)

    Returns
      t if successful
```

```
    else nil
    )
```

Value Returned

`t`	The command is successful.
`nil`	The command was unsuccessful.

Related Topics

VHDL Toolbox Functions

vhdlRegisterSimulator

```
vhdlRegisterSimulator(
     [ t_parserCallBack ]
     [ t_analyzerCallBack ]
     [ t_analyzerFileExt ]
     [ t_elaboratorCallBack ]
     [ t_simulatorCallBack ]
     [ t_dataDirCallBack ]
     [ t_dataFileCallBack ]
     [ t_workLibCallBack ]
     )
     => t / nil
```

Description

Lets you use non-Cadence VHDL tools by defining your own SKILL procedures and registering this information with the toolbox..

To register your callbacks, add the procedures for the callbacks in some file, forexample, `myfile.il`, that is in the `/home/xyz` directory and add the following lines to the `.cdsinit` file in your home directory:

```
(loadi "/home/xyz/myfile.il")
```

If you do not provide your own callback routines to invoke any of the non-Cadence tools, namely, the `parser/analyser/elaborator/simulator`, then by default, XM-VHDL tools such as the `parser/analyzer xmvhdl`, elaborator `xmelab` and simulator `xmsim` are run.

Arguments

t_parserCallBack This procedure takes the VHDL source file and the name of the library in which this file is contained and runs the parser on it.

t_analyzerCallBack

The procedure invokes the analyzer that analyzes the specified *sourceFileName* which exists in the specified directory `filePath`.

t_analyzerFileExt A string representing the name of the analyzed file.

t_elaboratorCallBack

This procedure invokes the elaborator to elaborate the VHDL design unit.

t_simulatorCallBack	
	The procedure invokes the simulator that simulates the specified simulation model.
t_dataDirCallBack	Given the library, cell, and view name, this procedure returns the physical directory where the VHDL text file is to be stored.
t_dataFileCallBack	
	Given the library, cell and view name, this procedure returns the physical file name under which the VHDL text file is to be stored.
t_workLibCallBack	This procedure returns the library that contains the compiled design unit information.

Value Returned

`t`	The operation was successful.
`nil`	The operation was unsuccessful.

Related Topics

VHDL Toolbox Functions

vhdlToPinList

```
vhdlToPinList(
     t_libName
     t_cellName
     t_viewName
     )
     => list / nil
```

Description

Translates a VHDL cellview into an intermediate pin list format.

Arguments

t_libName	The name of the library containing the VHDL cellview.
t_cellName	The cell name of the VHDL cellview to be translated.
t_viewName	The view name of the VHDL cellview to be created.

Value Returned

list	A pin list is returned.
nil	The command was not successful.

Observe the following for the returned pin list:

l_pinList	A DPL list describing the cellview ports, cellview properties, and port properties in the following format:
<l_pinList>	(nil ports <portList> [props <propList>]
<portList>	(<port> [<portList>])
<port>	(nil name "termName" direction "termDir"
	[prop <propList>] [pins <pinList>])
<propList>	(<prop> [<propList>])
<prop>	(nil s_propName t_propValue s_propName t_propValue)
<pinList>	(<pin> [<pinList>])
<pin>	(nil name "pinName" accessDir "pinAccessDir" [prop <propList>])

Examples

The following example shows the output of the `vhdlToPinList` function.

```
pinList =
     list(nil
          'ports list(
               list(nil 'name "clock" 'direction "input")
               list(nil 'name "a0"    'direction "output"
                    'prop list(nil 'delay 55.0))
          )
          'prop list(nil 'partName "count" 'myProp 5)
     )
```

Related Topics

VHDL Toolbox Functions

vhmsCompilationFailure

```
vhmsCompilationFailure(
     t_libName
     t_cellName
     t_viewName
     t_fileName
     )
     => l_list / nil
```

Description

Informs about the compilation failures of the file specified with the *fileName* parameter. A compilation error message is displayed and you can open the file for editing by clicking 'Yes'.

Arguments

t_libName	The name of the library currently in use.
t_cellName	The name of the cell currently in use.
t_viewName	The name of the view currently in use.
t_fileName	The name of the file to be compiled.

Value Returned

l_list	A list of compilation failures is returned.
nil	The operation was unsuccessful.

Related Topics

VHDL Toolbox Functions

vhmsDefaultEdit

```
vhmsDefaultEdit(
    t_libName
    t_cellName
    t_viewName
    t_fileObj
    t_mode
    [ t_errFile ]
    )
    => t / nil
```

Description

Opens an existing file or creates a template and opens it in an editor. If the mode is set to `r`, which indicates read-only then it tries to open an existing file.

First an entity needs to be created, only then can the architecture, package, or body views be created. When creating a template for an entity, if the symbol exists, then this function calls the `vhmsSymbolToPinListGen` function and then use this pin list to create the template.

Arguments

t_libName	The name of the library currently in use.
t_cellName	The name of the cell currently in use.
t_viewName	The name of the view currently in use.
b_fileObj	The file object obtained from the `ddGetObj()` procedure.
t_mode	The mode, which can be read or write.
t_errFile	Name of the error file.

Value Returned

`t`	The operation is successful.
`nil`	The operation was unsuccessful.

Related Topics

VHDL Toolbox Functions

vhmsGetCellParameters

```
vhmsGetCellParameters(
     t_libName
     t_cellName
     )
     => list / nil
```

Description

Returns the CDF parameters of the `vhdl` generics. The sub-list returned contains the name of the generic, the default value, and the type.

Arguments

t_libName	The name of the library currently in use.
t_cellName	The name of the cell currently in use.

Value Returned

list	The CDF parameters of the `vhdl` generics.
`nil`	Theoperation was unsuccessful because `ddGetObj` was unable to return a valid `lib/cell/view` name, or if there were no CDF parameters defined, or if the CDF could not be read.

Related Topics

VHDL Toolbox Functions

vhmsPinListToVHDLAMS

```
vhmsPinListToVHDLAMS(
     t_libName
     t_cellName
     t_viewName
     l_pinList
     )
     => t / nil
```

Description

Creates a `vhdlams` entity, package, body, configuration, or architecture depending on the view name. Specify the view name as `entity`, `package`, `body`, or `configuration` for creating an entity, package, body, or configuration respectively. If nothing is specified, an architecture view is created. The view is created in `./libName/cellName/viewName`.

Arguments

t_libName	The name of the library currently in use.
t_cellName	The name of the cell currently in use.
t_viewName	The name of the view currently in use.
l_pinList	The pin list to be converted.

Value Returned

`t`	The operation is successful.
`nil`	The operation was unsuccessful.

Related Topics

VHDL Toolbox Functions

vhmsSaveFile

```
vhmsSaveFile(
     t_libName
     t_cellName
     t_viewName
     t_fileName
     )
     => l_list / nil
```

Description

Compiles the `vhms` file. If the symbol does not exist, it displays a prompt to create the symbol. If the compilation fails, it calls the `vhmsCompilationFailure` function.

Arguments

t_libName	The name of the library currently in use.
t_cellName	The name of the cell currently in use.
t_viewName	The name of the view currently in use.
t_fileName	The `vhms` file name.

Value Returned

l_list	The created `vhms` compilation.
`nil`	the operation was unsuccessful.

Related Topics

VHDL Toolbox Functions

vhmsSymbolToPinListGen

```
vhmsSymbolToPinListGen(
     t_libName
     t_cellName
     [ t_viewName ]
     )
     => l_pinList / nil
```

Description

Generates a pin list from the symbol.

Arguments

t_libName	The name of the library currently in use.
t_cellName	The name of the cell currently in use.
t_viewName	The view name. The default value is `symbol`.

Value Returned

l_pinList	The generated pin list.
`nil`	The operation was unsuccessful. In this case, `vhmsSymbolToPinListGen` fails only when `schSymbolToPinList` cannot be called.

Related Topics

VHDL Toolbox Functions

vhmsToPinList

```
vhmsToPinList(
      t_libName
      t_cellName
      t_viewName
      )
      => l_list / nil
```

Description

Translates a VHDLAMS cellview into an intermediate pin list format. The pin list represents all the ports to the VHDLAMS description and their directions, such as `input`, `output`, and `inout`.

Arguments

t_libName	The name of the library currently in use.
t_cellName	The name of the cell currently in use.
t_viewName	The name of the view currently in use.

Value Returned

l_list	The pin list.
`nil`	The operation was unsuccessful.

Related Topics

VHDL Toolbox Functions

vhmsUpdateCellCDFParams

```
vhmsUpdateCellCDFParams(
     d_cellId
     l_pinList
     )
     => cdf_obj
```

Description

Updates the CDF parameters by reading all the parameters from the pin list. The CDF is updated if a given parameter does not exist in the CDF.

Arguments

d_cellId	The cellview identifier.
l_pinList	The pin list for which the CDF parameters need to be updated.

Value Returned

cdf_obj	The CDF object.

Related Topics

VHDL Toolbox Functions

vosHiDisplayNetlist

```
vosHiDisplayNetlist(
     )
     => t / nil
```

Description

Displays the generated VHDL netlist. This function can be used only when a single netlist is generated for the complete design.

Arguments

None

Value Returned

`t`	The VHDL netlist is displayed.
`nil`	The VHDL netlist was not displayed or the operation was unsuccessful.

Examples

The following example displays the generated VHDL netlist.

```
vosHiDisplayNetlist()
=> t
```

Related Topics

VHDL Toolbox Functions

vosLaunchIrunSimulation

```
vosLaunchIrunSimulation(
     )
     => t / nil
```

Description

Launches the xrun simulator from the command line. If the *`vhdlSimNetlistandSimulate`* variable is set to `t`, the function netlists the design before launching the xrun simulator. Otherwise, it launches the simulator without netlisting the design.

Arguments

None

Value Returned

`t`	The xrun simulator was launched.
`nil`	The xrun simulator was not launched or the operation was unsuccessful.

Examples

The following example launches the xrun simulator from the command line.

```
si <runDirName> -batch -command vosLaunchIrunSimulation
=> t
```

Related Topics

VHDL Toolbox Functions