

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

**Product Version IC23.1
June 2023**

© 2023 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

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

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Cadence is committed to using respectful language in our code and communications. We are also active in the removal and replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1

Verilog In and VHDL In Import Functions 5

<u>Licensing Requirements</u>	6
<u>impHdlDisplay</u>	7
<u>vhdlHiImport</u>	8
<u>vhdlImport</u>	9
<u>vhdlPinListToVHDL</u>	11
<u>vhdlRegisterSimulator</u>	13
<u>vhdlToPinList</u>	15

2

Connectivity-to-Schematic Functions 17

<u>Licensing Requirements</u>	17
<u>conn2Sch</u>	18
<u>conn2SchImpHdlDisplay</u>	21
<u>conn2SchStartUp</u>	22

Verilog In and VHDL In Import Functions

This manual is intended for designers who want to use Verilog and VHDL files in designs maintained in the Virtuoso Studio design environment.

This topic describes the SKILL functions used to import HDL files into the Virtuoso® design environment and to convert netlists to schematic diagrams.

- Verilog In Function - Virtuoso Verilog In lets you convert structural Verilog netlists into one of the following forms:

- ☐ Virtuoso schematics
- ☐ Netlists in Cadence OpenAccess format
- ☐ Verilog text views in a Cadence-format library

In each case, the design is converted into a data format that can be used by Cadence® tools. The following SKILL function lets you access the Verilog In tool:

- ☐ [impHdlDisplay](#)

- VHDL In Functions - Virtuoso VHDL In lets you convert a VHDL structural or behavioral description into one of the following types of views:

- ☐ Schematic view
- ☐ Netlist view
- ☐ VHDL text views

In each case, VHDL In imports the design from the VHDL format into Virtuoso database format, a data format that can be used by Cadence tools. You can import the following into a Virtuoso library:

- ☐ VHDL designs
- ☐ VHDL ASIC libraries
- ☐ VHDL designs, minus modules that already exist in the Virtuoso® Design Environment library

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Verilog In and VHDL In Import Functions

- ❑ Pieces of a hierarchical design
- ❑ A combination of the above architecture

The following SKILL functions let you perform various tasks using the VHDL In tool:

- ❑ [vhdlHiImport](#)
- ❑ [vhdlImport](#)
- ❑ [vhdlPinListToVHDL](#)
- ❑ [vhdlRegisterSimulator](#)
- ❑ [vhdlToPinList](#)

Licensing Requirements

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

Related Topics

[*Getting Started with VHDL In*](#)

[*Using Verilog In*](#)

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Verilog In and VHDL In Import Functions

impHdlDisplay

```
impHdlDisplay(  
    impHdlOptionsForm  
)  
=> nil
```

Description

Displays the *Verilog In* form. Alternatively, in the CIW, choose *File – Import – Verilog* to open the *Verilog In* form.

Arguments

<i>impHdlOptionsFormMain</i>	The <i>Verilog In</i> form name.
------------------------------	----------------------------------

Values Returned

<i>nil</i>	The operation was unsuccessful.
------------	---------------------------------

Examples

```
impHdlDisplay(impHdlOptionsFormMain)  
=> nil
```

Related Topics

[Verilog In and VHDL In Import Functions](#)

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Verilog In and VHDL In Import Functions

vhdlHiImport

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

Description

Builds and displays the VHDL Import form.

Arguments

None

Value Returned

t	The form is displayed.
nil	The operation was unsuccessful.

Examples

```
vhdlHiImport ()  
=> t
```

Related Topics

[vhdlImport](#)

[vhdlPinListToVHDL](#)

[vhdlRegisterSimulator](#)

[vhdlToPinList](#)

[Verilog In and VHDL In Import Functions](#)

vhdlImport

```
vhdlImport(  
    t_libName  
    l_srcFiles  
    t_logName  
    l_params  
    [ g_runInBackground ]  
    [ g_displayResults ]  
)  
=> nil / t
```

Description

Opens VHDL In to let you import a list of VHDL source files into the specified library with the given parameters. The parameters are the names of the VHDL In parameters, passed in as a disembodied property list. Optionally, it can run VHDL In as a background process (vhdlin) and/or display the results interactively.

Arguments

<i>t_libName</i>	Name of the target library where files are imported.
<i>l_srcFiles</i>	List of the VHDL text files to be imported by vhdlin.
<i>t_logName</i>	Name of the log file for vhdlin to generate.
<i>l_params</i>	Disembodied Property List (DPL) defining the vhdlin parameters, where the members of the DPL match the names of the parameters used by vhdlin.
<i>g_runInBackground</i>	Boolean flag, specifying whether vhdlin should be run in the foreground, blocking the current session, or as a background process. The default is <code>nil</code> . (runs in foreground.)
<i>g_displayResults</i>	Boolean flag, specifying whether the results of the vhdlin run should be displayed interactively using the VHDL Toolbox log/error viewing window. The default is <code>nil</code> . (does not display results interactively.)

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Verilog In and VHDL In Import Functions

Value Returned

<code>t</code>	The VHDL source files were imported in the specified library with the given parameters.
<code>nil</code>	The operation was unsuccessful.

Examples

Related Topics

[vhdlHiImport](#)

[vhdlPinListToVHDL](#)

[vhdlRegisterSimulator](#)

[vhdlToPinList](#)

[Verilog In and VHDL In Import Functions](#)

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Verilog In and VHDL In Import Functions

Value Returned

t	The VHDL view is generated.
nil	The command was unsuccessful.

Examples

Related Topics

[vhdlHiImport](#)

[vhdlImport](#)

[vhdlRegisterSimulator](#)

[vhdlToPinList](#)

[Verilog In and VHDL In Import Functions](#)

vhdlRegisterSimulator

```
vhdlRegisterSimulator(  
    [ parserCallBack ]  
    [ analyzerCallBack ]  
    [ analyzerFileExt ]  
    [ elaboratorCallBack ]  
    [ simulatorCallBack ]  
    [ dataDirCallBack ]  
    [ dataFileCallBack ]  
    [ workLibCallBack ]  
)  
=> t / nil
```

Description

To register your callbacks, add the procedures for the callbacks in some file, say `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")
```

To use non-Cadence VHDL tools, you need to define your own SKILL procedures and register this information with the toolbox using the SKILL routine, `vhdlRegisterSimulator()`.

If you do not provide your own callback routines to invoke any of the non-Cadence tools, namely, the `parser/analyzer/elaborator/simulator`, then by default, XM-VHDL tools such as the `parser/analyzer xmvhdl`, `elaborator xmelab`, and `simulator xmsim` are run.

Arguments

<i>parserCallBack</i>	Takes the VHDL source file and the name of the library in which this file is contained and runs the parser on it.
<i>analyzerCallBack</i>	Invokes the analyzer that analyzes the specified <i>sourceFileName</i> which exists in the specified directory <i>filePath</i> .
<i>analyzerFileExt</i>	A string representing the name of the analyzed file.
<i>elaboratorCallBack</i>	Invokes the elaborator to elaborate the VHDL design unit.
<i>simulatorCallBack</i>	Invokes the simulator that simulates the specified simulation model.

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Verilog In and VHDL In Import Functions

<i>dataDirCallBack</i>	Given the library, cell, and view name, this procedure returns the physical directory where the VHDL text file is to be stored.
<i>dataFileCallBack</i>	Given the library, cell, and view names, this procedure returns the physical file name under which the VHDL text file is to be stored.
<i>workLibCallBack</i>	Returns the library that contains the compiled design unit information.

Value Returned

<i>t</i>	The command is successful.
<i>nil</i>	The command is unsuccessful.

Examples

Related Topics

[vhdlHiImport](#)

[vhdlImport](#)

[vhdlPinListToVHDL](#)

[vhdlToPinList](#)

[Verilog In and VHDL In Import Functions](#)

vhdlToPinList

```
vhdlToPinList(  
    t_libName  
    t_cellName  
    t_viewName  
)  
=> list / nil
```

Description

Translates a VHDL cellview into an intermediate pin list format.

Arguments

<i>t_libName</i>	Name of the library containing the VHDL cellview.
<i>t_cellName</i>	Cell name of the VHDL cellview to be translated.
<i>t_viewName</i>	View name of the VHDL cellview to be created.

Value Returned

<i>list</i>	The pin list.
<i>nil</i>	The command is unsuccessful.

The pin list is returned in the following format:

<i>l_pinList</i>	A DPL list describing the cellview ports, cellview properties, and port properties in the following format:
<i><l_pinList></i>	(nil ports <i><portList></i> [props <i><propList></i>])
<i><portList></i>	(<i><port></i> [<i><portList></i>])
<i><port></i>	(nil name "termName" direction "termDir" [prop <i><propList></i>] [pins <i><pinList></i>])
<i><propList></i>	(<i><prop></i> [<i><propList></i>])
<i><prop></i>	(nil s_propName t_propValue s_propName t_propValue)
<i><pinList></i>	(<i><pin></i> [<i><pinList></i>])

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Verilog In and VHDL In Import Functions

```
<pin>                (nil name "pinName" accessDir "pinAccessDir"  
                      [prop <propList>] )
```

Examples

```
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

[vhdlHiImport](#)

[vhdlImport](#)

[vhdlPinListToVHDL](#)

[vhdlRegisterSimulator](#)

[Verilog In and VHDL In Import Functions](#)

Connectivity-to-Schematic Functions

This topic is intended for designers who want to generate schematic views from netlist views.

The Virtuoso Connectivity-to-Schematic tool is used to generate digital and analog schematic views from netlist views.

Using the Connectivity-to-Schematic SKILL functions, you can perform the following tasks:

- Generate a schematic view from an imported netlist view.
- Invoke the graphical user interface of Connectivity-to-Schematic.
- Accept the user interface name of Connectivity-to-Schematic as an argument and display the graphical user interface for the tool.

Licensing Requirements

For information on licensing in the Virtuoso design environment, see the [*Virtuoso Software Licensing and Configuration User Guide*](#).

Related Topics

[conn2Sch](#)

[conn2SchImpHdlDisplay](#)

[conn2SchStartUp](#)

[*Introducing Connectivity-to-Schematic*](#)

conn2Sch

```
conn2Sch(  
    t_srcLibName  
    t_srcCellName  
    t_srcViewName  
    [ ?destLibName t_destLibName ]  
    [ ?destCellName t_destCellName ]  
    [ ?destViewName t_destViewName ]  
    [ ?block t_block ]  
    [ ?paramFile t_paramFile ]  
    [ ?cdslib t_cdslib ]  
)  
=> t / nil
```

Description

Generates a schematic view from an imported netlist view.

The values of the following environment variables are taken from the `.cdsenv` file, by default. However, you can modify the values of these variables to customize the appearance of a schematic.

- `dest_symbol_view_name`
- `ref_lib_list`
- `log_file_name`
- `import_if_exists`
- `sheet_symbol`
- `line_line_spacing`
- `line_component_spacing`
- `density_level`
- `label_height`
- `page_col_limit`
- `page_row_limit`
- `pin_placement`
- `full_place_and_route`
- `optimize_wire_label_locn`

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Connectivity-to-Schematic Functions

- *minimize_crossovers*
- *generate_square_schematic*
- *extract_schematic*
- *verbose*
- *asg_options*
- *power_net*
- *ground_net*
- *power_symbol*
- *ground_symbol*

If you do not specify the *dest_sch_lib*, *dest_cell_name*, or *dest_view_name*, or if these parameters are empty or set to *nil*, then the values for these parameters are taken from the *.cdsenv* file. If the values for these parameters in the *.cdsenv* file are invalid, then the schematic is saved in the specified *src_sch_lib* with the specified *src_cell_name*.

If you specify a parameter file, *conn2sch* ignores the *log_file_name* parameter, if specified in the file. The tool saves the errors and log messages in the file specified using the *log_file_name* environment variable. If you set the *log_file_name* environment variable to "", its default value, *conn2sch.log*, is used.

Arguments

- | | |
|-------------------------------------|---|
| <i>t_srcLibName</i> | Specifies the source library name. |
| <i>t_srcCellName</i> | Specifies the cell name to be imported. |
| <i>t_srcViewName</i> | Specifies the view name to be imported. |
| <i>?destLibName t_destLibName</i> | Specifies the destination library name. |
| <i>?destCellName t_destCellName</i> | Specifies the destination cell name. |
| <i>?destViewName t_destViewName</i> | Specifies the destination view name. |
| | The default value is <i>schematic</i> . |

HDL Import and Netlist-to-Schematic Conversion SKILL Reference

Connectivity-to-Schematic Functions

`?block t_block` Specifies if conn2Sch needs to wait until import is complete.
The default value is `nil`.

`?paramFile t_paramFile`
Specifies the parameter file.
The default value is `nil`.

`?cdslib t_cdslib` Specifies the cdslib file.
The default value is `nil`.

Values Returned

`t` The command was successful.
`nil` The command was unsuccessful or an error was reported.

Examples

```
conn2Sch("testLib" "testCell" "netlist")  
=> t
```

Related Topics

[conn2SchImpHdlDisplay](#)

[conn2SchStartUp](#)

[Connectivity-to-Schematic Functions](#)

conn2SchImpHdlDisplay

```
conn2SchImpHdlDisplay(  
    conn2schOptionsForm  
)  
=> t / nil
```

Description

Accepts the user interface name of Connectivity-to-Schematic tool as an argument and displays the GUI for the tool.

Arguments

conn2schOptionsForm

Refers to the user interface of Connectivity-to-Schematic. The user interface accepts user inputs and other optional settings.

Values Returned

<i>t</i>	The command was successful and the GUI of the tool is invoked.
<i>nil</i>	The command was unsuccessful or an error was reported.

Note: The functions, `conn2SchImpHdlDisplay()` and `conn2SchStartUp()` perform the same task. The `conn2SchImpHdlDisplay()` function will be removed in the next release.

Examples

```
conn2SchImpHdlDisplay(conn2schOptionsFormMain)  
=> t
```

Related Topics

[conn2Sch](#)

[conn2SchStartUp](#)

[Connectivity-to-Schematic Functions](#)

conn2SchStartUp

```
conn2SchStartUp(  
    )  
=> t / nil
```

Description

Invokes the graphical user interface of the Connectivity-to-Schematic tool. This is a CIW menu callback function.

Arguments

None

Values Returned

t	The command was successful and the GUI of the tool is invoked.
nil	The command was unsuccessful or an error was reported.

Examples

```
conn2SchStartUp()  
=> t
```

Related Topics

[conn2Sch](#)

[conn2SchImpHdlDisplay](#)

[Connectivity-to-Schematic Functions](#)