

# **Virtuoso SystemVerilog Netlister User Guide**

**Product Version IC23.1  
August 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.

**Patents:** The Cadence Products covered in this manual are protected by U.S. Patents

5,790,436; 5,812,431; 5,859,785; 5,949,992; 6,493,849; 6,278,964; 6,300,765; 6,304,097; 6,414,498; 6,560,755; 6,618,837; 6,693,439; 6,826,736; 6,851,097; 6,711,725; 6,832,358; 6,874,133; 6,918,102; 6,954,908; 6,957,400; 7,003,745; 7,003,749.

**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

<b>Introduction to Virtuoso SystemVerilog Netlister</b>	7
Licensing Requirements	7
Benefits of SystemVerilog Netlister	8
Benefits of SystemVerilog Netlister over NC Verilog Netlister	11
Prerequisites for Using SystemVerilog Netlister	12
Launching SystemVerilog Netlister	13
SystemVerilog Netlister Batch Mode	15
SystemVerilog Config Views	17
Migrating SystemVerilog Integration Designs to SystemVerilog Designs	17

## 2

<b>Netlist Generation</b>	19
Netlist Generation Flow in SystemVerilog Netlister	20
Specifying a Design for Netlist Generation	21
Configuring Netlist Generation Options	22
Configuring Design Variables	24
Generating and Regenerating a Netlist	25
Viewing a Netlist	26
State Management	28
ATPG Compatible Verilog Netlists	30
Generating ATPG Compatible Verilog Netlists	31
Importing a SystemVerilog Package File	32
Specifying Additional xrun Arguments in SystemVerilog Netlister	33
DataType Propagation	35
Netlist Customization Using the .simrc File	40

## A

<b>SystemVerilog Netlister Forms</b>	45
Additional Arguments Form	46

## Virtuoso SystemVerilog Netlister User Guide

---

<u>Editing Design Variables Form</u>	46
<u>Load State Form</u>	46
<u>Save State Form</u>	47
<u>Select Design/Directory Form</u>	47
<u>SystemVerilog Netlister Graphical User Interface</u>	48
<u>SystemVerilog Netlister Options Form</u>	50
<u>Netlister Tab</u>	50
<u>Miscellaneous Tab</u>	54
<u>Verilog Tab</u>	57

## B

<u>Environment Variables</u>	59
<u>addArgTableList</u>	60
<u>addPrefixToCells</u>	61
<u>cellPrefixName</u>	62
<u>createXrunArgs</u>	63
<u>createXrunBinding</u>	64
<u>defaultNettype</u>	65
<u>enableDataPropagate</u>	66
<u>enableVerilogATPG</u>	67
<u>expandIterateInst</u>	68
<u>enableTimeScale</u>	69
<u>hdlVarFile</u>	70
<u>isPortInANSIFormat</u>	71
<u>mergedNetlist</u>	72
<u>nettypesToIgnore</u>	73
<u>refLib</u>	74
<u>reUseHdlCompilationSetup</u>	75
<u>simPrecision</u>	76
<u>simPrecisionUnit</u>	77
<u>simTime</u>	78
<u>simTimeUnit</u>	79
<u>termMismatch</u>	80
<u>termDirectionMismatch</u>	81
<u>useTranForCdsAliasThru</u>	82

## Virtuoso SystemVerilog Netlister User Guide

---

<u>vlogCompatVersion</u>	83
<u>vlogSupply0Sigs</u>	84
<u>vlogSupply1Sigs</u>	85

### C

#### SKILL Functions and Flags 87

<u>SKILL Functions</u>	87
<u>asiDigitalSimAutoloadProc</u>	88
<u>simDeRegPostNetlistTrigger</u>	89
<u>simDeRegPreNetlistTrigger</u>	90
<u>simPostNetlistTriggerList</u>	91
<u>simPreNetlistTriggerList</u>	92
<u>simRegPostNetlistTrigger</u>	93
<u>simRegPreNetlistTrigger</u>	94
<u>SKILL Flags</u>	95

### D

#### Batch Mode Options in SystemVerilog Netlister 97

<u>runsv</u>	98
<u>cdsCreateConfig</u>	100
<u>si2runsv</u>	102

# Virtuoso SystemVerilog Netlister User Guide

---

---

# Introduction to Virtuoso SystemVerilog Netlister

---

Digital system verification uses the SystemVerilog language extensively, and this has introduced the Digital-Mixed-Signal (DMS) use model. The DMS use model allows discrete models to represent analog circuits. SystemVerilog allows you to use user-defined type and resolution functions, which make the net obsolete as a scalar object.

These use-model changes require a netlister that supports modern constructs, imports data from a design database, and produces a simulator-compatible netlist. A netlister that has these capabilities can traverse the design hierarchy to build the complete structure of a netlist. In such cases, the hierarchy can be a schematic view and a text view, or only a text view.

Virtuoso® SystemVerilog Netlister is a utility that helps you generate netlists of digital SystemVerilog designs. This utility imports configuration views of digital designs for netlist generation, directly parses and accesses SystemVerilog and Verilog text models and creates LRM-compliant SystemVerilog configurations to generate compatible netlists.

This topic describes how to use the SystemVerilog Netlister to configure the environment for generating netlists of SystemVerilog designs. This topic is aimed at developers and 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 Schematic Editor.
- The Virtuoso Studio design environment technology file.

## Licensing Requirements

Virtuoso SystemVerilog Netlister requires the following licenses:

- Cadence Design Framework II license (License Number 111)

## Virtuoso SystemVerilog Netlister User Guide

### Introduction to Virtuoso SystemVerilog Netlister

---

- Virtuoso Schematic Editor Verilog(R) Interface license (License Number 21400)
- Virtuoso AMS Designer Environment license (License Number 70000) or Virtuoso Schematic Editor XL license (License Number 95115)

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

### ***Related Topics***

[Benefits of SystemVerilog Netlister](#)

[Benefits of SystemVerilog Netlister over NC Verilog Netlister](#)

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Batch Mode](#)

[Migrating SystemVerilog Integration Designs to SystemVerilog Designs](#)

## **Benefits of SystemVerilog Netlister**

SystemVerilog Netlister provides a quick and efficient approach to netlist SystemVerilog designs and provides the following features and capabilities:

---

Feature	Description
Dual Interface	Provides a graphical user interface and a command-line interface. Supports the batch or command-line mode using the <code>runsv</code> command.
LRM-compliant Netlist	Uses only SystemVerilog LRM constructs. This makes the netlist simulator-independent and available to a SystemVerilog-compliant tool. Vendor extensions are not supported.
HED Config	Fully supports HED configurations that match UNL. Netlist generation is based on HED configurations.



# Virtuoso SystemVerilog Netlister User Guide

## Introduction to Virtuoso SystemVerilog Netlister

Feature	Description
Direct Access to HDL File	<ul style="list-style-type: none"><li>■ Restricts the use of the direct access OA file.</li><li>■ Updating the OA for text or symbol for text-in-text instance is redundant.</li><li>■ Supports read-only libraries.</li><li>■ Supports handling ports of packed or unpacked arrays in the design.</li><li>■ Ensures accurate datatype propagation from leaf-level SystemVerilog and Verilog cellviews to top-level schematic views. The datatype propagation is based on the native datatype definitions from text files. By default, datatype propagation is disabled.</li><li>■ Ensures that instances are saved with explicit port connections. Supports smart connections with module ports of Verilog and SystemVerilog views.</li><li>■ Requires the following:<ul style="list-style-type: none"><li>□ Instance parameters are netlisted correctly and parameters are propagated in a UNL-supported method.</li><li>□ Instance parameters that differ from the default parameters use the explicit declaration format.</li></ul></li></ul>
Schematic Text Sandwich Configuration	<p>Supports the following formats:</p> <ul style="list-style-type: none"><li>■ Text-in-schematic</li><li>■ Schematic-in-text</li><li>■ Text-in-text</li></ul>
Text on Top	<p>Supports digital text-on-top views (Verilog, SystemVerilog). Allows descending into the leaf-level schematic or text views.</p>
Creation of SV 2001 Config for Bindings	<p>Supports:</p> <ul style="list-style-type: none"><li>■ Same cell from different libraries</li><li>■ Multiple views of the same cell</li></ul>

## Virtuoso SystemVerilog Netlister User Guide

### Introduction to Virtuoso SystemVerilog Netlister

Feature	Description
Config in Config	Supports config-in-config views where config view has the top-level schematic of different cells.
Symbol Avoidance	Avoids the requirement for a symbol unless the cell is instantiated in a schematic view.
Data type Handling	Supports SystemVerilog data types. Requires that any net, port, or bus from a schematic or symbol uses the <code>interconnect</code> net type, by default. Support for the <code>wire</code> net type is also available. Allows declaring internal signals with these nettypes.
Instance Parameters Handling	Requires the following: <ul style="list-style-type: none"><li>■ Instance parameters are netlisted correctly and parameters are propagated in a UNL-supported method.</li><li>■ Instance parameters that differ from the default parameters use the explicit declaration format.</li></ul>
Port Connection Handling	Ensures that instances are saved with explicit port connections. Supports smart connections with module ports of Verilog and SystemVerilog views.
Optional xrunArgs File Creation	Creates an <code>xrunArgs</code> file, which includes the full set of generated files.
Keywords Handling	Prefixes each 1800-2012 keyword with an escape character.
Inherited Connection Handling	Mandates the use of port-drilling only for schematic views.
Self-contained Netlist Creation	Creates a self-contained set of files without links to the text views in <code>dfl*</code> .
ANSI Port Declaration	Supports module port declaration in ANSI format.
Iterated Instance Support	Supports instance arrays instead of flattened instances in the design.
Design Variable Support	Supports the use of design variables. All design variables are saved in the <code>cds_globals.sv</code> file.
Shorting Support	Supports shorting devices with two terminals.
Bind to Open Support	Supports binding instances to open ports.

## Virtuoso SystemVerilog Netlister User Guide

### Introduction to Virtuoso SystemVerilog Netlister

---

Feature	Description
Save/Load State	Supports saving states with specified settings and subsequently loading these saved states for reuse.
cdsenv Support	Provides various <code>.cdsenv</code> options for configuring netlist generation. Saves the values of these options into states.
CDF Parameter Support	Fully supports CDF parameters, including <code>pPar</code> .  <b>Note:</b> For all stopping views, the defined CDF parameters are printed on the instance line, instead of the <code>defparam</code> statement.

---

### ***Related Topics***

[Introduction to Virtuoso SystemVerilog Netlister](#)

[Benefits of SystemVerilog Netlister over NC Verilog Netlister](#)

[Prerequisites for Using SystemVerilog Netlister](#)

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Batch Mode](#)

[Migrating SystemVerilog Integration Designs to SystemVerilog Designs](#)

## Benefits of SystemVerilog Netlister over NC Verilog Netlister

Some of the advantages that SystemVerilog Netlister offers, which the NC Verilog Netlister does not, are as follows:

- Symbol is redundant for text-in-text instances
- OA update for text is redundant
- Schematic text sandwich structure
- Support for read-only libraries
- Full support for HED config
- Flexible flowchart control

- Advanced parameter and bus handling
- Simulator-independent netlist and binding

### ***Related Topics***

[Introduction to Virtuoso SystemVerilog Netlister](#)

[Benefits of SystemVerilog Netlister](#)

[Prerequisites for Using SystemVerilog Netlister](#)

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Batch Mode](#)

[Migrating SystemVerilog Integration Designs to SystemVerilog Designs](#)

## **Prerequisites for Using SystemVerilog Netlister**

You must have access to the following to use Virtuoso SystemVerilog Netlister:

- Cadence Virtuoso 64-bit version (IC6.1.8 ISR8 or higher)

SystemVerilog Netlister is launched from Virtuoso.

- Cadence Xcelium™ `xrun` utility 64-bit version (18.09 or higher)

The `xrun` utility helps you specify all input files and options in a single command.

It can take SystemVerilog designs as input. The utility uses the Cadence Native Code tools to compile and netlist designs.

Additionally, ensure that you update your `.cshrc` file as follows:

- Specify the path to the Virtuoso installation.
- Specify the path to the Xcelium installation.



### ***Tip***

To ensure that you use the 64-bit version of Virtuoso and Xcelium, add the environment variable `setenv CDS_AUTO_64BIT ALL` to your `.cshrc` file.

### ***Related Topics***

[Introduction to Virtuoso SystemVerilog Netlister](#)

[Benefits of SystemVerilog Netlister](#)

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Batch Mode](#)

[Migrating SystemVerilog Integration Designs to SystemVerilog Designs](#)

## **Launching SystemVerilog Netlister**

SystemVerilog Netlister provides a simple and efficient interface to let you configure settings and options for netlist generation.

To launch the SystemVerilog Netlister application and specify the design:

1. Launch Virtuoso.
2. In the CIW, choose *Tools – SystemVerilog – Netlister*.

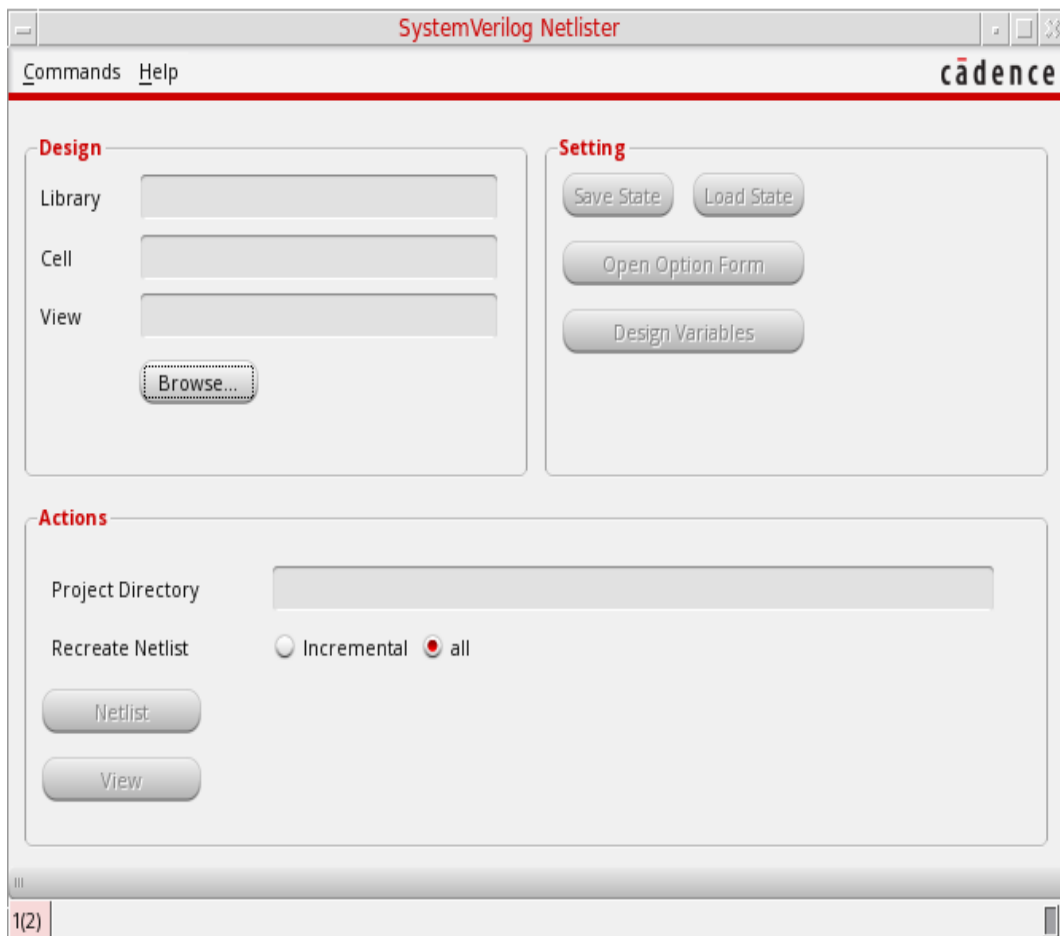
The SystemVerilog Netlister window appears.

## Virtuoso SystemVerilog Netlister User Guide

### Introduction to Virtuoso SystemVerilog Netlister

---

3. Click *Browse* to specify the design.



Alternatively, to first select the design and then launch SystemVerilog Netlister:

1. Launch Virtuoso.
2. In the CIW, choose *Tools – Library Manager*.  
The Library Manager window appears.
3. In the Library Manager window, select a library from the *Library* list.  
The cells present in the specified library appear in the *Cell* list.
4. Select a cell from the *Cell* list.  
The views present in the specified cell appear in the *View* list.
5. Select a `dn1_state*` view from the *View* list.

The SystemVerilog Netlister window appears and shows the design.

### ***Related Topics***

[Introduction to Virtuoso SystemVerilog Netlister](#)

[Benefits of SystemVerilog Netlister over NC Verilog Netlister](#)

[Prerequisites for Using SystemVerilog Netlister](#)

[SystemVerilog Netlister Batch Mode](#)

[Migrating SystemVerilog Integration Designs to SystemVerilog Designs](#)

## **SystemVerilog Netlister Batch Mode**

Before you launch SystemVerilog Netlister ensure that you meet all the [prerequisites](#).

SystemVerilog Netlister can be launched in batch mode by running the `runsv` command. This command supports the following options:

```
-lib <libName>
-cell <cellName>
-view <config_viewname>
-rundir <projectdir>
-netlist
-cdsenv <path of .cdsenvfile>
-state <saved SystemVerilog Netlister state (stateLib/stateCell/stateView)>
```

For example, if you have a design `topLib/topCell/config_sv` and you specify the project directory as `runsv_run`, use the following command to generate the netlist in batch mode:

```
runsv -lib topLib -cell topCell -view config_sv -netlist -rundir runsv_run
```

To increase performance, load the initialization details in a `.runsvinit` file instead of the `.cdsinit` file.



### ***Tip***

You can optionally use the `-cdsenv` option to read the file. However, if you want to enable data type propagation or create an additional argument file, set the `enableDataPropagate`, `defaultNettype`, `mergedNetlist`, and `createXrunArgs` environment variables in the `.cdsenv` or the `.cdsinit` file.

***Related Topics***

[Introduction to Virtuoso SystemVerilog Netlister](#)

[Prerequisites for Using SystemVerilog Netlister](#)

[createXrunArgs](#)

[defaultNettype](#)

[enableDataPropagate](#)

[mergedNetlist](#)

[runsv](#)



## SystemVerilog Config Views

When using SystemVerilog Netlister, ensure that your specified design has a config view that contains only digital content. This digital content can be a SystemVerilog or Verilog text view, a schematic view, or a symbol view. If your design does not have a config view, use the `cdsCreateConfig` utility available with SystemVerilog Netlister. This utility helps you create a config view.

You can use the following options with the `cdsCreateConfig` utility:

```
cdsCreateConfig
  -lib          topLibName
  -cell         topCellName
  -view         topViewName
  [-config      newConfigName]
  [-liblist     'lib1 lib2']
  [-viewlist    'view1 view2']
  [-stoplist    'stopview1 stopview2']
```

For example, if you have a design `topLib/topCell/schematic`, use the following command in the terminal window to create a config view:

```
cdsCreateConfig -lib topLib -cell topCell -view schematic
```

To view the various options available with the `cdsCreateConfig` utility, type the following command in the terminal window.

```
cdsCreateConfig -help
```

### ***Related Topics***

[`cdsCreateConfig`](#)

[Introduction to Virtuoso SystemVerilog Netlister](#)

[SystemVerilog Netlister Batch Mode](#)

## Migrating SystemVerilog Integration Designs to SystemVerilog Designs

Convert commands used in the Virtuoso Verilog Environment for SystemVerilog Integration Environment (SI) to a format supported by SystemVerilog Netlister by using the `si2runsv` utility.

You run [SystemVerilog Integration netlister](#) by using the following command:

## Virtuoso SystemVerilog Netlister User Guide

### Introduction to Virtuoso SystemVerilog Netlister

---

```
si [run directory] [-batch [-command commandName]]
```

For example:

```
si dir1 -batch -command netlist
```

Alternatively, SystemVerilog Netlister uses the following command:

```
runsv -lib lib -cell cell -view configView -netlist -state stateName
```

To convert a SystemVerilog Integration environment command to a SystemVerilog Netlister command:

1. Run the following command to generate a script file:

```
si2runsv
```

This command generates a `runsv_xxxfile` script file.

2. Run the following command to run SystemVerilog Netlister using this script file:

```
si2runsv dir1 -batch -command netlist
```

To view the various options available with the `si2runsv` utility, type the following command in the terminal window.

```
si2runsv -help
```

### ***Related Topics***

[si2runsv](#)

[Introduction to Virtuoso SystemVerilog Netlister](#)

[SystemVerilog Netlister Batch Mode](#)

---

## Netlist Generation

---

Use the SystemVerilog Netlister to specify a design, configure the netlist generation options, configure design variables, netlist the design, view the netlist, and manage the states. You can generate a netlist, which contains connectivity information of a design, after you have specified the design. Configure the netlist generation options before you generate the netlist. When you generate the netlist, SystemVerilog Netlister creates a netlist file of your design based on the settings that you specify and lets you view the netlist file.

### ***Related Topics***

[Specifying a Design for Netlist Generation](#)

[Configuring Netlist Generation Options](#)

[ATPG Compatible Verilog Netlists](#)

[Importing a SystemVerilog Package File](#)

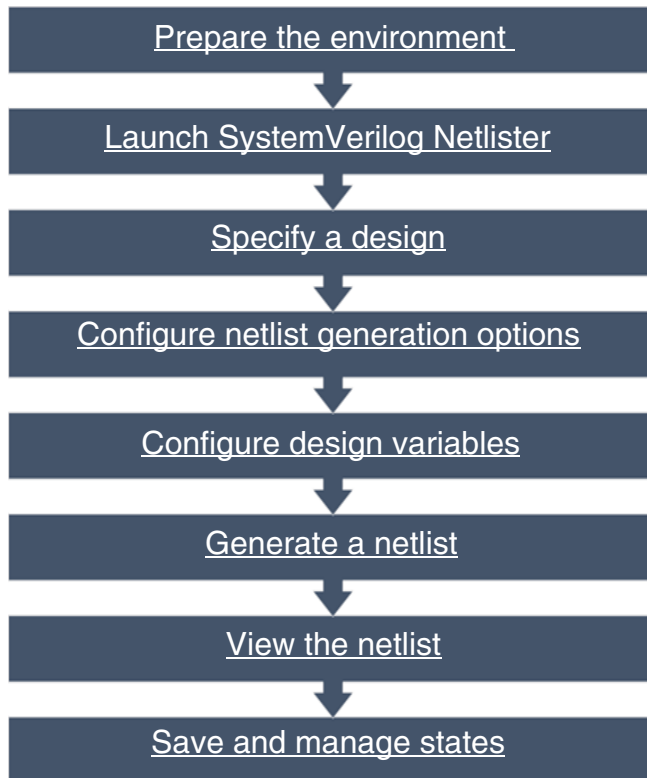
[Specifying Additional xrun Arguments in SystemVerilog Netlister](#)

[DataType Propagation](#)

[Netlist Customization Using the .simrc File](#)

## Netlist Generation Flow in SystemVerilog Netlister

The following diagram depicts the overall netlist generation flow.



To know about any issues that you might encounter while using SystemVerilog Netlister, refer to the log in the Virtuoso CIW. The status of the last operation is also visible on the SystemVerilog Netlister window.

### ***Related Topics***

[Specifying a Design for Netlist Generation](#)

[ATPG Compatible Verilog Netlists](#)

[Importing a SystemVerilog Package File](#)

[Specifying Additional xrun Arguments in SystemVerilog Netlister](#)

[DataType Propagation](#)

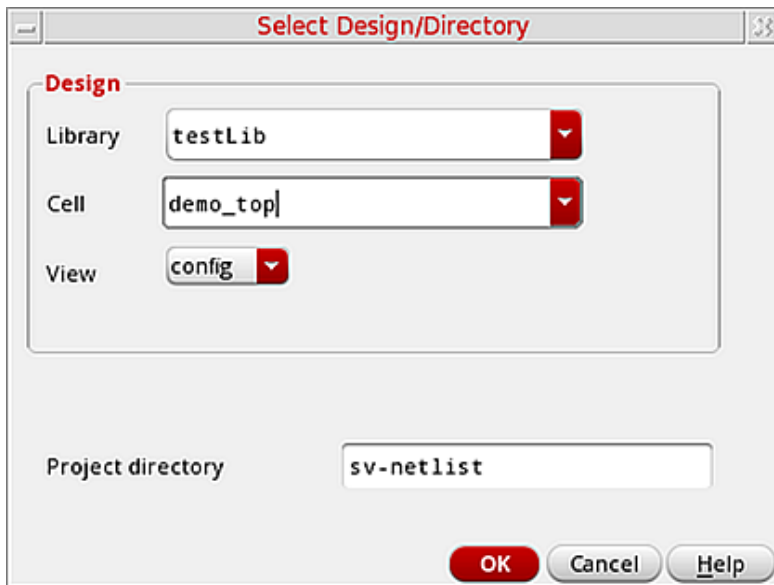
## Netlist Customization Using the .simrc File

# Specifying a Design for Netlist Generation

To start netlist generation by specifying a design:

1. Open the SystemVerilog Netlister window.
2. Click *Browse* in the *Design* group box.

The Select Design/Directory Form appears.



3. Select a library from the *Library* list.  
The cells in the specified library appear in the *Cell* list.
4. Select a cell from the *Cell* list.  
The views in the specified cell appear in the *View* list.
5. Select a config view from the *View* list.
6. Specify a new project directory name in the *Project directory* field, if required.  
The default project directory name is `sv-netlist`.
7. Click *OK* to select the design.

The CIW displays an appropriate message to indicate that the design specification is successful.

### ***Related Topics***

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Graphical User Interface](#)

[Select Design/Directory Form](#)

[State Management](#)

## **Configuring Netlist Generation Options**

Before you generate the netlist of a design, ensure that you configure the netlist generation options, as required. SystemVerilog Netlister generates the netlist based on how you configure these options. By default, the `netlist` file is stored in the `<projectDir>/<lib>_<cell>_<view>/netlist/` directory. For example:

`projectDir/lib_cell_view/netlist/netlist.sv`

To configure netlist generation options, based on which the SystemVerilog Netlister generates a netlist:

1. Launch SystemVerilog Netlister.
2. In the *Settings* group box, click *Open Option Form* to set additional options.

## Virtuoso SystemVerilog Netlister User Guide

### Netlist Generation

The SystemVerilog Netlister Options Form appears with the *Netlister* tab selected.

The screenshot shows the 'SystemVerilog Netlister Options' dialog box. The 'Netlister' tab is active. The options are as follows:

- Port Declaration Format: ☒ Non-ANSI, ☐ ANSI
- Default Nettype: ☒ interconnect, ☐ wire
- Enable Datatype Propagation: ☐
- Enable Array Type Propagation Only: ☐
- Enable auto package importing: ☐
- Merge text source to single netlist: ☐
- netTypes to ignore on schematic nets: [Text Field]
- supply0: [Text Field]
- supply1: [Text Field]
- Pre-Module Include File: [Text Field] [Browse Button]
- In-Module Include File: [Text Field] [Browse Button]

Buttons at the bottom: **OK**, Cancel, Defaults, Apply, Help.

3. In the *Netlister* and *Miscellaneous* tabs of the SystemVerilog Netlister Options form, specify the required values or options and click *OK*.

For Verilog cellviews, specify the values or options on the *Verilog* tab of the SystemVerilog Netlister Options form.

The SystemVerilog Netlister options are successfully configured.

### ***Related Topics***

SystemVerilog Netlister Options Form

Launching SystemVerilog Netlister

SystemVerilog Netlister Graphical User Interface

## Generating and Regenerating a Netlist

### Viewing a Netlist

## Configuring Design Variables

You can add, delete, or change design variables in your netlisting settings in SystemVerilog Netlister after choosing your design.

To set up a design variable:

1. Click *Design Variables* in the SystemVerilog Netlister window.

The Editing Design Variables Form opens.

Design Variables	
Name	Value

2. Specify a name for the design variable in the *Name* field.
3. Specify a value for the design variable in the *Value (Expr)* field.
4. Click *Add* to add a variable.

The design variable appears in the *Design Variables* list on the right and is saved in the `cds_globals.sv` file.

5. Click *Delete* to delete a design variable by selecting the variable name from the *Design Variables* list box.
6. Click *Change* to change the name or value of a design variable by selecting the variable name from the *Design Variables* list box.

Edit the name or the value of the design variable, as required.



7. Click *OK*.

The design variable is updated as specified.

### ***Related Topics***

[Launching SystemVerilog Netlister](#)

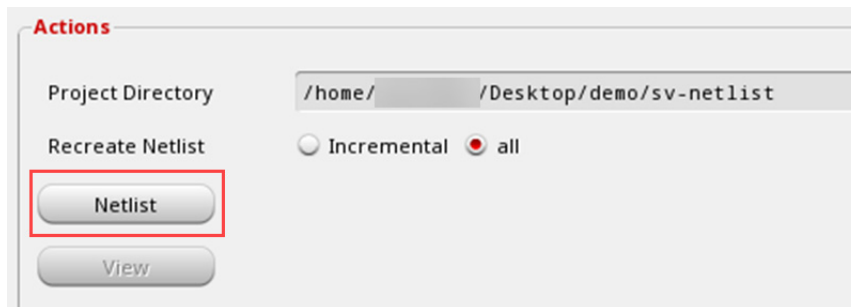
[SystemVerilog Netlister Graphical User Interface](#)

[Editing Design Variables Form](#)

## **Generating and Regenerating a Netlist**

To generate a netlist of a selected design:

- After setting up options and variables for the selected design, click *Netlist* in the Actions group box of the SystemVerilog Netlister window.



SystemVerilog Netlister does the following:

- ❑ Generates the netlist and stores it in the `<projDir>/<lib>_<cell>_<view>/netlist` directory.
- ❑ Enables the *View* button to allow viewing the netlist file.

**Note:** By default, SystemVerilog Netlister creates the `netlist.sv` file in the `./sv-netlist/<lib>_<cell>_<view>/netlist` directory.

To regenerate the netlist:

- In the *Actions* group box, select one of the following:

## Virtuoso SystemVerilog Netlist User Guide

### Netlist Generation

- ❑ *Incremental*: Netlists only the changes that you make to the design or the settings. In case of large designs, this mode of netlist generation highly improves the performance.
- ❑ *All*: Recreates the complete netlist. In case of large designs, this mode of netlist generation might decrease the performance.

### **Related Topics**

[SystemVerilog Netlist Graphical User Interface](#)

[Launching SystemVerilog Netlist](#)

[Netlist Generation Flow in SystemVerilog Netlist](#)

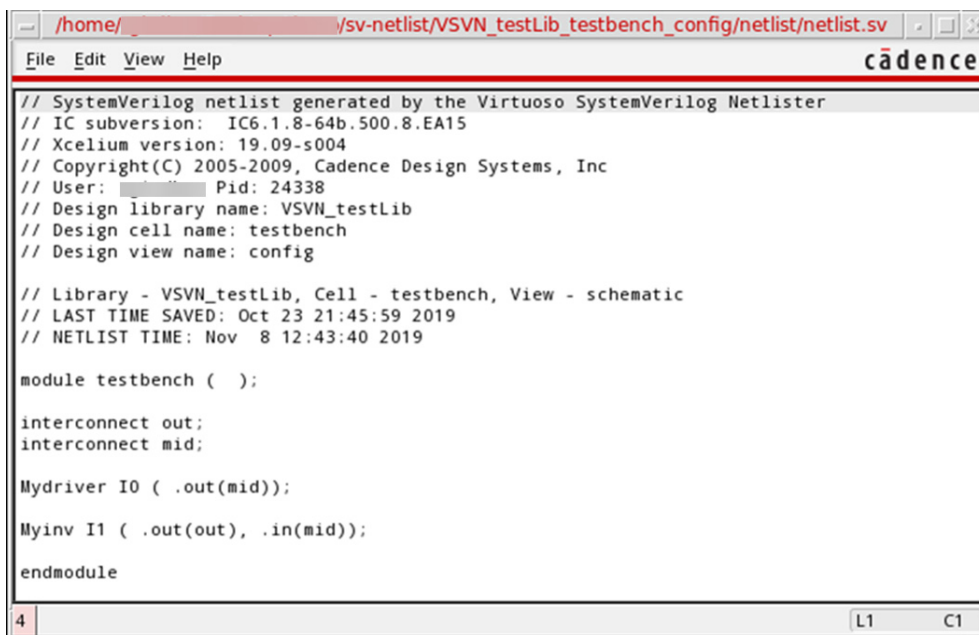
## Viewing a Netlist

A netlist contains the connectivity information of a design. To view the netlist of your design:

- ➔ In the *Actions* group box of the SystemVerilog Netlist window, click *View*.

The `netlist.sv` file opens.

The following figure illustrates how SystemVerilog Netlist displays the netlist file:



***Related Topics***

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Graphical User Interface](#)

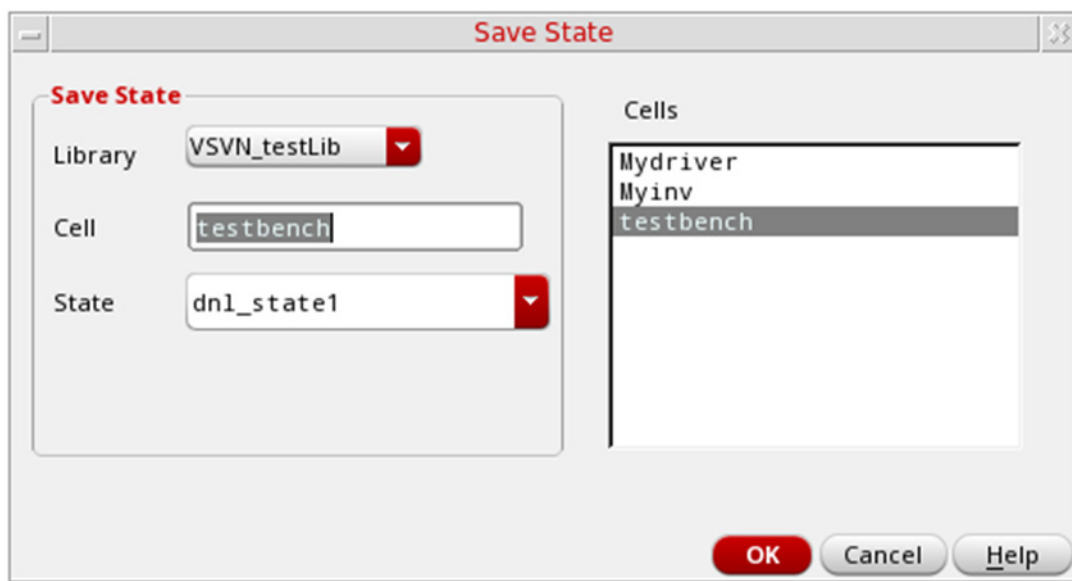
[Netlist Generation Flow in SystemVerilog Netlister](#)

## State Management

You can save the current state of your settings or load saved states of settings in SystemVerilog Netlister. It is useful when you want to avoid repeated setups of netlisting settings.

To save a state with the specified settings:

1. In the *Settings* group box of the SystemVerilog Netlister window, click *Save State*. The Save State form appears.

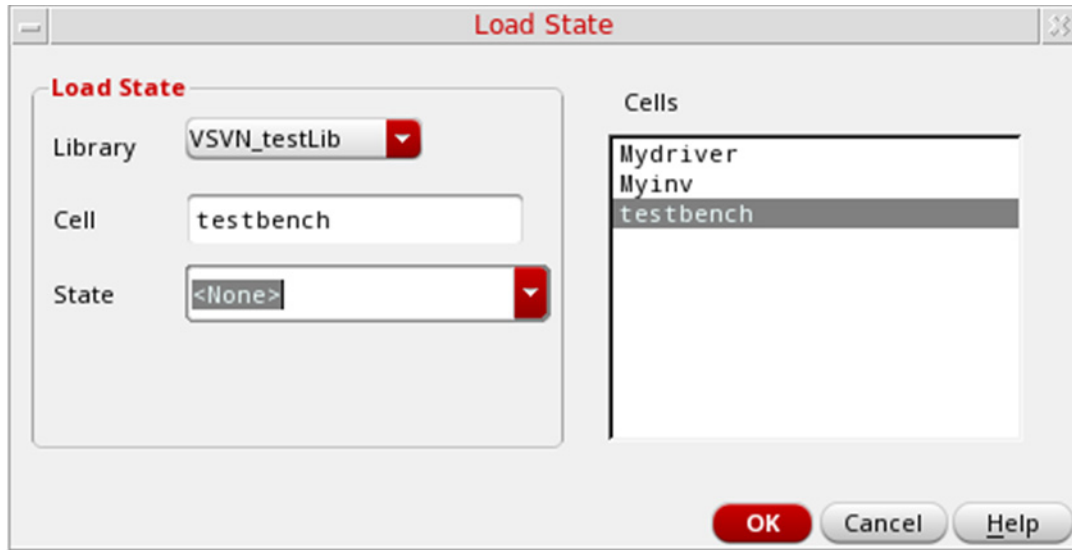


2. In the *State* field, specify a new name for the current state.
3. Click *OK*.

The current settings are saved as a state in SystemVerilog Netlister.

To load a saved state when you launch SystemVerilog Netlister:

1. In the *Settings* group box of the SystemVerilog Netlister window, click *Load State*. The Load State form appears.



2. In the *State* list, select the name of the state that you want to load.
3. Click *OK*.

The saved state is loaded in SystemVerilog Netlister.

### ***Related Topics***

[Save State Form](#)

[Load State Form](#)

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Graphical User Interface](#)

## ATPG Compatible Verilog Netlists

The primary responsibility of engineers is to deliver scan patterns with complete confidence in their accuracy. Automatic Test Pattern Generation (ATPG) tools are used to generate scan patterns. Scan pattern generation as a process requires three inputs: the netlist of the design, ATPG tool configuration files, and the definition of the library cells used.

However, due to limited support for behavioral constructs, usually ATPG tools cannot use Verilog definitions of library cells except Verilog-1995 structural netlists. As a result, the library cells must be defined in a tool-specific language or in a simpler, structural form of Verilog.

Virtuoso SystemVerilog Netlister lets you generate netlists of Verilog text files that are compatible with the ATPG technology. These netlists let you find an input or test sequence in Verilog text files. The ATPG functionality within Virtuoso SystemVerilog Netlister has the following main features:

- All text views must be Verilog constructs. Views created in other languages are not supported.
- All nets are defined as `wire` in the netlist.
- All parameters of cells or instances are ignored and are not printed in netlist.
- All instance arrays are expanded by default.
- Aliases can be printed using `tran` statements instead of using `cds_alias`, `cds_thru`, or `assign` statements.
- Generates a single netlist file. This netlist file contains the Verilog modules created from the schematic and the Verilog modules from the text views. Cells with same names from different libraries and different views are generated in a single netlist file with unique names.

### ***Related Topics***

[Generating ATPG Compatible Verilog Netlists](#)

[Verilog Tab](#)

[enableVerilogATPG](#)

[useTranForCdsAliasThru](#)

[vlogCompatVersion](#)

## Generating ATPG Compatible Verilog Netlists

To generate an ATPG compatible netlist using the SystemVerilog Netlister graphical interface:

1. Launch SystemVerilog Netlister.
2. Click *Browse* in the *Design* group box to specify a design.
3. Click the *Open Option Form* button in the *Settings* group box to specify the netlisting options.

The SystemVerilog Netlister Options form appears.

4. Select the *Verilog* tab.
5. Select *Create ATPG compatible netlist*.
6. Click *OK*.
7. Click the *Netlist* button in the *Actions* group box.

The ATPG compatible netlist is generated.

To generate an ATPG compatible netlist using the SystemVerilog Netlister batch mode:

1. Include the following in the `.cdsinit` file.

```
envSetVal("digitalSim.netlisterOpts" "enableVerilogATPG" 'boolean t)
```

2. Run the following command in the terminal window.

```
runsv -lib topLib -cell topCell -view config_sv -netlist -rundir runsv_run
```

The ATPG compatible netlist is generated.

### ***Related Topics***

[ATPG Compatible Verilog Netlists](#)

[Verilog Tab](#)

[enableVerilogATPG](#)

[useTranForCdsAliasThru](#)

[vlogCompatVersion](#)

## Importing a SystemVerilog Package File

Before you import SystemVerilog package files into your design for netlisting, ensure that your design has a package file, for example, `global_package.sv`, and a systemVerilog view that has imported this package file.

To import a package:

**1. In the Library Manager window:**

- a.** Create a new `systemVerilogPackage` view by choosing *File – New – Cell View*.

The New File form opens.

- b.** Specify the library and cell name in their respective fields.

- c.** In the *View* list, select `systemVerilogPackage`.

- d.** Click *OK*.

The New File form closes, and the *View* list in the Library Manager shows the new view.

- e.** Double-click the `systemVerilogPackage` view.

The view opens with an empty package module in Virtuoso Text Editor.

**2. In the Virtuoso Text Editor window:**

- a.** Choose *File – Open*.

The Open File form opens.

- b.** Select the `global_package.sv` package file and open it in a *new tab*.

- c.** Copy the contents of the package file and paste them into the package module of the `systemVerilogPackage` view.

- d.** Click the *Check and Save* button on the toolbar.

Correct any errors that are reported.

- e.** Open the `systemVerilog` view in a *new tab*.

This is the view that has imported the package file.

- f.** Click the *Check and Save* icon on the toolbar.

Ensure that no errors are reported.



3. In the SystemVerilog Netlister window:

a. In the *Setting* group box, click *Open Option Form*.

This opens the SystemVerilog Netlister Options form and shows the Netlister tab.

b. On the Netlister page, select *Enable auto package importing*.

c. Click *OK* to close the SystemVerilog Netlister Options form.

d. In the Action group box, click *Netlist*.

Ensure that no errors are reported.

The package file is imported into your design.

### ***Related Topics***

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Graphical User Interface](#)

[SystemVerilog Netlister Options Form](#)

## **Specifying Additional xrun Arguments in SystemVerilog Netlister**

To specify additional `xrun` arguments in the netlisting settings in SystemVerilog Netlister:

1. In the SystemVerilog Netlister window, click *Browse* to specify a design.

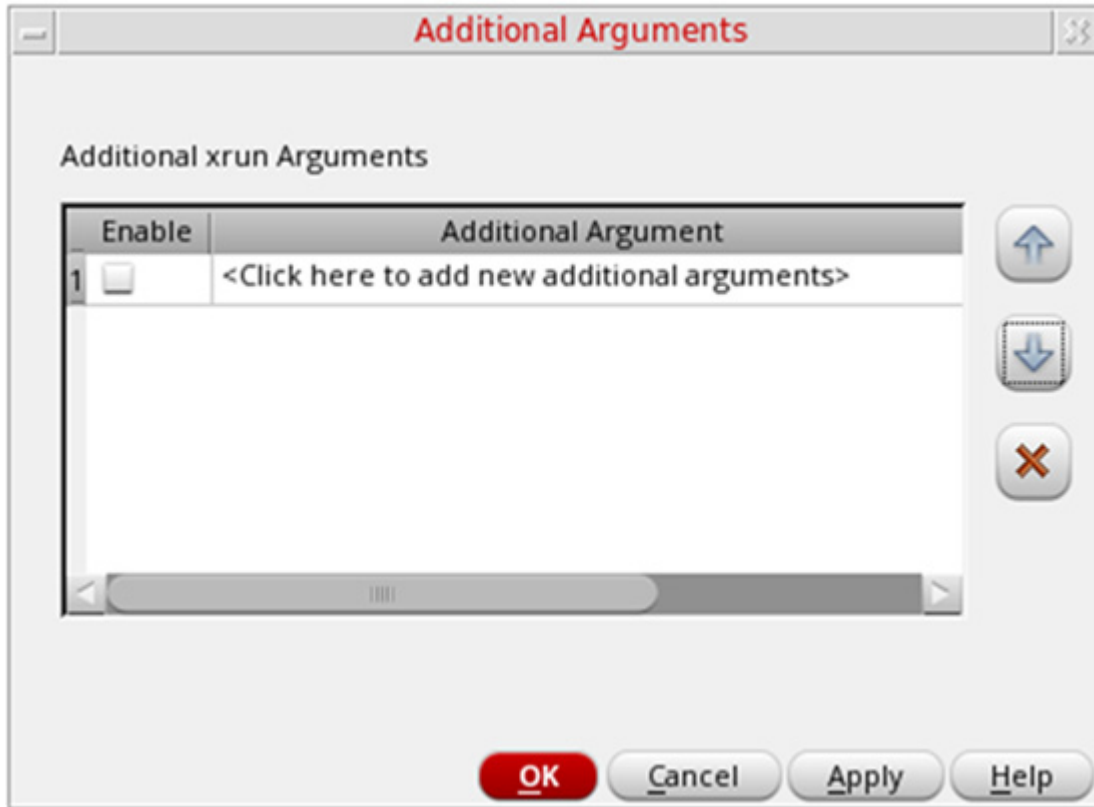
The fields in the window are populated with the design details.

2. Click *Open Options Form* to access the netlisting settings.

3. In the Open Options Form window, click the *Miscellaneous* tab.

4. In the Miscellaneous page, click *Additional Arguments*.

The Additional Arguments form opens.



5. Specify valid `xrun` arguments in the Additional Arguments form.

These arguments are appended to the `xrunArgs` file that is generated. If *Create binding files for xrun only* is enabled, SystemVerilog Netlister appends these additional arguments to both the `xrunArgs` and `xrunArgs_vy` files.

6. Click OK.

The Additional Arguments form closes.

### ***Related Topics***

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Graphical User Interface](#)

[Additional Arguments Form](#)

## DataType Propagation

The SystemVerilog Netlister allows you to modify the `dataType`, `portKind`, and `isUnpacked` properties, which are specific to an instance.

The datatype can be propagated from the following:

- From text views
- From instance ports on the symbol (place master).
- From instance ports in the schematic

When `ignoreDataType` is set to `t`, the `dataType`, `portKind`, and `isUnpacked` properties are ignored. Instead, the `dataType` information that is propagated from the bottom-level cell to the top-level cell is considered.

You can add the `ignoreDataType` property on a specific instance terminal in the schematic. If this property is selected, the SystemVerilog Netlister will not print the *Master Value* and the *Local Value*.

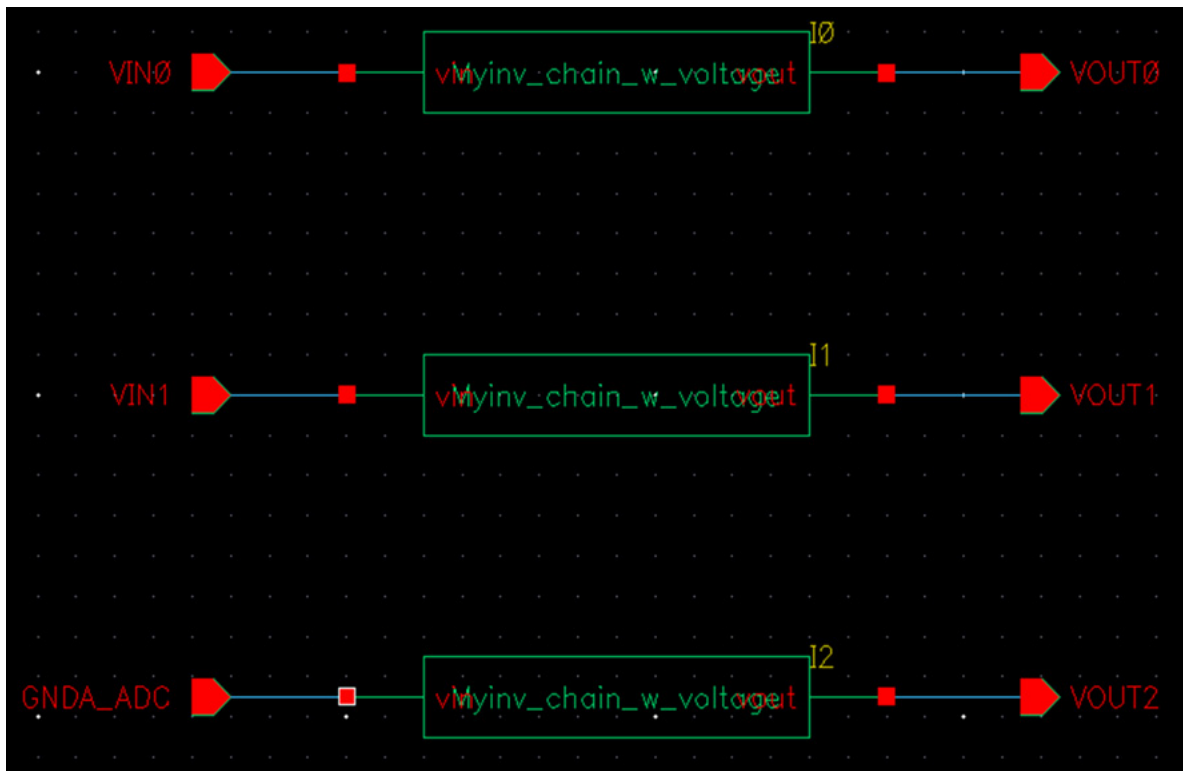
Additionally, you can modify the local values of the `dataType`, `portKind`, and `isUnpacked` port properties that are associated with a specific instance of a cell.

The following table clearly describes the impact of enabling and disabling the `ignoreDataType` property on the port of an instance in different scenarios:

Condition	Additional Condition	Result
<code>ignoreDataType = t</code>		The master values and local values of <code>dataType</code> , <code>portKind</code> , and <code>isUnpacked</code> are ignored. Instead, <code>dataType</code> information propagated from the bottom cell to the top cell is used.
<code>ignoreDataType = nil</code>	The local value of <code>dataType</code> is set to <i>custom_value</i> .	The local value of the specific instance is used instead of the <i>value</i> set on the symbol cell.
<code>ignoreDataType = nil</code>	The local values are not set for <code>dataType</code> , <code>portKind</code> , and <code>isUnpacked</code> .	The master value of the symbol cell property is used.

### Example

Consider the input port I2 in the following schematic and the related condition scenarios that follow.



### ■ When ignoreDataType is set to t on the port of an instance

When you set `ignoreDataType` to `t` on I2, the master and local values of `dataType` and `portKind` are ignored. In such a case, `dataType` information that is propagated from the bottom-level cell to the top-level cell is used.

User Property	Master Value	Local Value	Display
dataType	voltage		off
isRefPort	false		off
portKind			off
ignoreDataType		<input checked="" type="checkbox"/>	off

The following example shows how the netlist displays the port information in this scenario:

```
module tb_w_voltage (
output wire logic VOUT0,
output wire logic VOUT1,
output wire logic VOUT2,
input wire logic GNDA_ADC,
input voltage VIN0,
input voltage VIN1
);
```

Here, the `wire logic` value is derived from the `dataType` property of the bottom-level cell.

■ **When ignoreDataType is set to nil and the local value of dataType is set to *custom\_value***

When you set ignoreDataType to nil on I2 and the local value dataType to myvoltage, the local value myvoltage of the specific instance is used, instead of the master value voltage that is set on the symbol cell.

User Property	Master Value	Local Value	Display
dataType	voltage	myvoltage	off
isRefPort	false		off
portKind			off
ignoreDataType		nil	off

The following example shows how the netlist displays the port information in this scenario:

```
module tb_w_voltage (
output wire logic VOUT0,
output wire logic VOUT1,
output wire logic VOUT2,
input myvoltage GNDA_ADC,
input voltage VIN0,
input voltage VIN1
);
```

Here, the local value overrides the master value.

### ■ When ignoreDataType is set to nil and the local value of dataType is not set

When you set ignoreDataType to nil on I2 and the local values of dataType and portKind are not set, the master value voltage of the symbol cell property is used.

The following example shows how the netlist displays the port information in this scenario:

```
module tb_w_voltage (
output wire logic VOUT0,
output wire logic VOUT1,
output wire logic VOUT2,
input voltage GNDA_ADC,
input voltage VIN0,
input voltage VIN1
);
```

Here, the property of the symbol cell (master value) is used.

### ***Related Topics***

[Launching SystemVerilog Netlister](#)

[SystemVerilog Netlister Graphical User Interface](#)

## Netlist Customization Using the .simrc File

SystemVerilog Netlister supports customizing netlist generation by setting the following variables in the .simrc file.

**Note:** SystemVerilog Netlister might not always support all the variables in the .simrc file that are supported by the SI Netlister.

Variable	Description
hnlGetSimulator	Returns SystemVerilog Netlister for SystemVerilog Netlister.
hnlUserShortCVList	Allows specifying a shorting list that contains devices that need to be shorted.  Example: <code>hnlUserShortCVList=list(list(&lt;libName&gt;&lt;cellName&gt;))</code>
simVerilogEnableEscapeNameMapping	Includes escaped names in the netlist. It also allows you to escape names that are reserved keywords in SystemVerilog.  Default value is t.  Example: If you have a module "assign" in your design, and you set <code>simVerilogEnableEscapeNameMapping</code> to t, it is mapped to "\assign" in the netlist.



# Virtuoso SystemVerilog Netlist User Guide

## Netlist Generation

---

Variable	Description
----------	-------------

---

`simSVPortPropertyList`

When enabled, allows specifying the `dataType` and `portKind` properties on a symbol cell. For example:

```
simSVPortPropertyList = '( ("analogLib" "res" "symbol"  
"PLUS real var" "MINUS wrealdriver nil") )
```

Here,

- `"analogLib"` indicates the library name
- `"res"` indicates the cell name
- `"symbol"` indicates the view name
- `"PLUS real var"` indicates that `dataType` is set to `real`, and `portKind` is set to `var` on port `PLUS`;
- `"MINUS wrealdriver nil"` indicates that only the `dataType` property is set to `wrealdriver` on port `MINUS`, leaving the `portKind` property blank.

`vlogExpandIteratedInst`

When set to `nil`, allows instances in the array in the following format:

```
"I_iter[0:2]"
```

When set to `t`, allows splitting iterated instances in the following format:

```
I_iter_1 ...  
I_iter_2 ...  
I_iter_3 ...
```

## Virtuoso SystemVerilog Netlister User Guide

### Netlist Generation

Variable	Description
<code>hnlVerilogDumpIncludeFilesInNetlist</code>	<p>When set to <code>t</code>, copies the content of an included HDL file directly to the netlist, instead of inserting an <code>`include</code> statement.</p> <p>The contents of the text cellviews in the design hierarchy are copied to the netlist. Any file specified in the <i>Pre-Module Include File</i> or <i>In-Module Include File</i> option of the SystemVerilog Netlister Options form is also copied to the netlist.</p> <p>This variable works only in single netlist file mode. It does not support recursive inclusion of text files. Consider that <code>fileA.sv</code> includes <code>fileB.sv</code>. If you copy the contents of <code>fileA.sv</code> in the netlist using this variable, the contents of <code>fileB.sv</code> will not be copied in the netlist.</p> <p>Default: <code>nil</code></p>
<code>simVerilogGenerateSingleNetlistFile</code>	<p>A flag to generate a single Verilog netlist containing multiple modules instead of one netlist per module. By default, this flag is set to <code>nil</code>. If set to <code>t</code>, the netlister generates a single Verilog netlist file in the current simulation run directory with the name <code>netlist</code>.</p>
<code>hnlUserStopCVList</code>	<p>List of user specified cellviews, which are treated as stop views while netlisting a design. You can specify this list in the <code>.simrc</code> file. Although instances of such a cellview appear in a netlist, the cellview module is not printed in the netlist.</p> <p>In the example below, all the cellviews in the <code>libN</code> library will be treated as stop views. However, in the <code>lib1</code> library, only the <code>cell11</code>, <code>cell12</code>, and <code>cell13</code> cellviews will be treated as stop views.</p> <pre>hnlUserStopCVList = list (     ;;; all cells from this library     "libN"     ;;; cell11, cell12 and cell13 from lib1     list("lib1" "cell11" "cell12" "cell13"s) )</pre> <p><b>Note:</b> The list should have only one entry for each library, listing all the cellviews that need to be treated as stop views.</p>

## Virtuoso SystemVerilog Netlister User Guide

### Netlist Generation

---

Variable	Description
<code>vlogifDeclareGlobalNetLocal</code>	A flag to declare global signals locally. When you disable this flag, the netlister uses the default signals (Global Power Nets and Global Grounds Nets). If the <code>vlogifDeclareGlobalNetLocal</code> flag is set to <code>nil</code> , global signals are declared in the <code>cds_globals</code> module.

**Note:** The SystemVerilog Netlister does not support the `hnlUserStubCVList` variable.

### ***Related Topics***

[The .simrc File](#)

# **Virtuoso SystemVerilog Netlister User Guide**

## **Netlist Generation**

---

---

## SystemVerilog Netlister Forms

---

This chapter describes the various forms in the SystemVerilog Netlister environment. These include:

- [Additional Arguments Form](#)
- [Editing Design Variables Form](#)
- [Load State Form](#)
- [Save State Form](#)
- [Select Design/Directory Form](#)
- [SystemVerilog Netlister Graphical User Interface](#)
- [SystemVerilog Netlister Options Form](#)

## Additional Arguments Form

The following table describes the columns available in the Additional Arguments form:

Field	Description
<i>Enable</i>	Enables the selected <code>xrun</code> argument.
<i>Additional Argument</i>	Specifies the name of the <code>xrun</code> argument. Click the field to add a new argument name or edit an existing argument name.

## Editing Design Variables Form

The following table describes the fields and buttons available in the Editing Design Variables form:

Field	Description
<i>Name</i>	Specifies a name for the design variable.
<i>Value(Expr)</i>	Specifies a value for the design variable.
<i>Add</i>	Lets you add a variable.
<i>Delete</i>	Lets you delete a variable.
<i>Change</i>	Specifies the name of the <code>xrun</code> argument. Click the field to add a new argument name or edit an existing argument name.
<i>Design Variables</i>	Lists the design variables that exist in the design.

## Load State Form

Use the Load State form to load previously saved settings or a state to the current setup. The form contains the following fields and lists:

Field	Description
<i>Library</i>	Specifies the name of the library that contains the design.
<i>Cell</i>	Specifies the name of the cell in the selected library for which the state needs to be saved.

## Virtuoso SystemVerilog Netlister User Guide

### SystemVerilog Netlister Forms

---

---

Field	Description
<i>State</i>	Specifies the name of the state that must be saved.
<i>Cells</i>	Lists the names of all cells in the selected library.

---

## Save State Form

Use the Save State form to save the settings from the current setup as a state. The form contains the following fields and lists:

---

Field	Description
<i>Library</i>	Specifies the name of the library that contains the design.
<i>Cell</i>	Specifies the name of the cell in the selected library for which the state needs to be saved.
<i>State</i>	Specifies the name of the state that must be saved.
<i>Cells</i>	Lists the names of all cells in the selected library.

---

## Select Design/Directory Form

The Select Design/Directory form contains the following fields:

---

Field	Description
<i>Library</i>	Specifies the name of the library that contains the design.
<i>Cell</i>	Specifies the name of the cell in the selected library that contains the design.
<i>View</i>	Specifies the view in the selected library and cell.

## Virtuoso SystemVerilog Netlister User Guide

### SystemVerilog Netlister Forms

---

Field	Description
<i>Project directory</i>	<p>Specifies the name of the project directory. If the project directory name already exists, a dialog box is displayed to confirm if the previous project directory must be overwritten. If you choose not to overwrite the project directory, specify a new name in the text field.</p> <p>The default project directory name is <code>sv-netlist</code>.</p> <p><b>Note:</b> The project directory contains various subdirectories. When you select a library, cell, and view in the <i>Select Design/Directory</i> window, a new subdirectory is created in the project directory. The subdirectory derives its name from the library, cell, and view names that you select in the design. For example, the typical directory structure of a design is as follows:</p> <pre>projectDir/lib_cell_view/netlist</pre>

---

## SystemVerilog Netlister Graphical User Interface

The graphical user interface of SystemVerilog Netlister can be categorized into the following main sections:

Section	Description
Menus	Displays the Commands and Help menus.
Design	<p>Specifies the library, cell, and view of the top-level design.</p> <p><b>Note:</b> You can select only configuration views.</p>
Setting	Specifies the options and design variables required for netlist generation.
Actions	Specifies the netlisting actions. You can generate or regenerate a netlist to view the netlisting results.
Status bar	Displays the current status of the selected item.

---

### ***Related Topics***

[Specifying a Design for Netlist Generation](#)

[Configuring Netlist Generation Options](#)

[Configuring Design Variables](#)



# **Virtuoso SystemVerilog Netlister User Guide**

## **SystemVerilog Netlister Forms**

---

### Netlist Generation Flow in SystemVerilog Netlister

## SystemVerilog Netlister Options Form

The SystemVerilog Netlister Options form lets you specify the options for running SystemVerilog Netlister. The form contains the following tabs.

Tab	Description
<u>Netlister Tab</u>	Lets you specify port declaration format, default nettype, enablement of datatype and array type propagation, auto-package import, merging text files to a single netlist, and values for global ground and power nets and pre-module and in-module include files.
<u>Miscellaneous Tab</u>	Lets you specify the <code>hdl.var</code> file, pre-compiled libraries, creation of argument files, creation of binding files for xrun, printing the global time scale, and additional arguments.
<u>Verilog Tab</u>	Lets you specify options for generating Verilog netlists.

### Netlister Tab

The following table describes the fields available on the *Netlister* tab of the SystemVerilog Netlister Options form.

Field	Description
<i>Port Declaration Format</i>	<p>Specifies one of the following port declaration formats:</p> <ul style="list-style-type: none"> <li>■ Non-ANSI (default)</li> <li>■ ANSI</li> </ul> <p>When <code>hdlPrintNonAnsiSV</code> is set to <code>nil</code> in the <code>.simrc</code> file and datatype propagation is enabled, port declaration can be done only in ANSI format.</p>
<i>Default Nettype</i>	<p>Specifies one of the following nettypes to netlist the nets on the schematic:</p> <ul style="list-style-type: none"> <li>■ <i>interconnect</i> (default): If no nettype is propagated, the net and the bus are declared explicitly as <i>interconnect</i>.</li> <li>■ <i>wire</i>: A net without another explicitly set type is not declared. Here, only the bus is declared. Single-bit wires are not saved in the declaration because Xcelium resolves signals that are not explicitly declared to be of type <i>wire</i>.</li> </ul>

## Virtuoso SystemVerilog Netlister User Guide

### SystemVerilog Netlister Forms

Field	Description
<i>Enable Datatype Propagation</i>	<p>Enables datatype propagation. It is not selected by default.</p> <p>Selecting this check box allows the propagation of both packed or unpacked and datatype or portKind properties. When not selected and <code>netType</code> properties are not set on the schematic nets, the default nettype that is currently set is printed in the netlist.</p> <p>Datatype propagation and <u>default net type</u> selection result in the following scenarios:</p> <ul style="list-style-type: none"> <li>■ When the <code>netType</code> property is explicitly placed on a net and <i>Enable Datatype Propagation</i> is selected, the propagated datatype is ignored and the explicit nettype defined on the net is printed in the generated netlist. A warning is displayed in the CIW if there is a conflict between the nettype set on schematic net and the propagated nettype.</li> <li>■ When datatype propagation is enabled and <code>netType</code> property is not set on schematic nets, the propagated <code>datatype</code> or <code>portKind</code> properties for these nets are printed in netlist.</li> <li>■ When datatype propagation is disabled, and <code>netType</code> property is set on schematic nets, the value of <code>netType</code> property is printed for these nets in the netlist. Otherwise, the default <code>netType</code> is printed.</li> <li>■ When datatype propagation is enabled, and a net has a propagated <code>datatype</code> as well as a <code>netType</code> property placed on it, the <code>netType</code> property value is printed for the net. To print the propagated datatype while ignoring the <code>netType</code> property set on nets, specify the <code>netType</code> value to be ignored in the <i>netTypes to ignore on schematic nets</i> field.</li> </ul>
<i>Enable Array Type Propagation Only</i>	<p>Enables array type propagation. It is not selected by default.</p> <p>Selecting this check box disables the <i>Enable Datatype Propagation</i> check box and allows recognition of packed and unpacked arrays. As a result, the SystemVerilog Netlister prints a bus connected to a port of type <code>real</code> or a nettype, as unpacked, and propagates the unpacked property up to the top level. However, the datatype of the port is not propagated.</p>
<i>Expand Iterated Instances</i>	<p>Prints iterated instances in expanded form in the netlist.</p>

## Virtuoso SystemVerilog Netlister User Guide

### SystemVerilog Netlister Forms

Field	Description
<i>Enable auto package importing</i>	<p>Enables the auto-package handling feature, which allows SystemVerilog Netlister to automatically search and find the SystemVerilog package files on which the <code>systemverilog</code> modules in the design depend. It is not selected by default.</p> <p>Alternatively, you can manually pre-compile a package into a library that you define.</p>
<i>netTypes to ignore on schematic nets</i>	Specifies the net types that must be ignored during netlising.
<i>Merge text source to single netlist</i>	<p>Merges the text files to create a single and self-contained <code>netlist.sv</code> file. This file includes the netlist from the schematic and the original source files from the text view.</p> <p>If you select the <i>Merge text source to single netlist</i> check box, the contents of the <code>cds_alias.sv</code> or the <code>cds_globals.sv</code> file are printed in the <code>netlist.sv</code> file.</p> <p>However, deselecting the <i>Merge text source to single netlist</i> check box results in the following:</p> <ul style="list-style-type: none"> <li>■ The absolute path of <code>cds_alias.sv</code> is printed in <code>textInput</code>s but removed from the <code>netlist.sv</code> file.</li> <li>■ The absolute path of <code>cds_global.sv</code> is printed in <code>xrunArgs</code> but removed from the <code>netlist.sv</code> file.</li> <li>■ When <code>simVerilogGenerateSingleNetlistFile</code> is set to <code>t</code> in the <code>.simrc</code> file, it generates a single merged netlist, including <code>cds_alias.sv</code> and <code>cds_globals.sv</code>, if these files exist. When set to <code>nil</code>, it generates a split netlist.</li> <li>■ When <i>Pre-Module Include File</i> or <i>In-Module include File</i> and <i>Merge text source to single netlist</i> are selected, the contents of <i>Pre-Module Include File</i> or <i>In-Module Include File</i> are saved in <code>netlist.sv</code>.</li> </ul>
<i>supply0</i>	<p>Specifies a value to set the global ground net. This value is saved in the <code>cds_globals.sv</code> file.</p> <p>This option supports only global signals in the design. Cross-checking with the schematic is not supported.</p>

## Virtuoso SystemVerilog Netlister User Guide

### SystemVerilog Netlister Forms

Field	Description
<i>supply1</i>	<p>Specifies a value to set the global power net. This value is saved in the <code>cds_globals.sv</code> file.</p> <p>This option supports only global signals in the design. Cross-checking with the schematic is not supported.</p>
<i>Pre-Module Include File</i>	<p>Specifies the file that the netlister must use as the include file before the module declaration in the netlist file generated for each hierarchical cellview.</p> <p>To print the contents of the include file in the generated netlist file instead of using the file inclusion directive <code>'include</code>, set <code>hnlVerilogDumpIncludeFilesInNetlist</code> to <code>t</code>.</p> <p><b>Note:</b> When <i>Merge text source to single netlist</i> is selected, the contents of the include file are also printed in a single netlist file even if <code>hnlVerilogDumpIncludeFilesInNetlist</code> is not set.</p> <p>Flag: <code>vlogifPreModuleIncludeFile</code></p>
<i>In-Module Include File</i>	<p>Specifies the file that the netlister must use as an include file immediately after the module declaration in the netlist file generated for each hierarchical cellview.</p> <p>To print the contents of the include file in the generated netlist file instead of using the file inclusion directive <code>'include</code>, set <code>hnlVerilogDumpIncludeFilesInNetlist</code> to <code>t</code>.</p> <p><b>Note:</b> When <i>Merge text source to single netlist</i> is selected, the contents of the include file are also printed in a single netlist file even if <code>hnlVerilogDumpIncludeFilesInNetlist</code> is not set.</p> <p>Flag: <code>vlogifInModuleIncludeFile</code></p>

## Miscellaneous Tab

The following table describes the fields available on the *Miscellaneous* tab of the SystemVerilog Netlister Options form.

Field	Description
<i>hdl.var file</i>	Specifies the name of an <code>hdl.var</code> file to include the simulation options.  For more details on the <i>hdl.var</i> file, see the <a href="#">Xcelium XRUN User Guide</a> .
<i>Pre-compiled libraries (-reflib)</i>	Specifies the name of a pre-compiled library. If you have a package definition, you can pre-compile the package into the same library or different libraries.  <b>Note:</b> To manage text views of a design, pre-compile all the packages into a library. You can use <code>-reflib</code> to include all the libraries that contain the pre-compiled packages required in the design.
<i>Create arguments file</i>	Creates the <code>xrunArgs</code> file and other standard binding files. This file includes <code>netlist.sv</code> , <code>config.sv</code> , <code>textInputs</code> , binding files, and other files, such as <code>cds_globals.sv</code> and <code>cds_alias.sv</code> . These files are generated during netlisting and are needed for simulations.

## Virtuoso SystemVerilog Netlister User Guide

### SystemVerilog Netlister Forms

Field	Description
<i>Create binding files for xrun only</i>	<p>Select this check box only if you use <code>xrun</code>. Selecting this check box automatically selects the <i>Create arguments file</i> check box and creates two sets of binding files in the netlist directory: <code>xrunArgs</code>, <code>xrunArgs_vy</code> (compatible with Xcelium), and the <code>hdl.var</code>, <code>cds_xrun.lib</code> and <code>lib.map</code> files.</p> <p>Both <code>xrunArgs</code> and <code>xrunArgs_vy</code> include the following:</p> <ul style="list-style-type: none"> <li>■ <code>xrun argument vcfg_inst_precedence</code></li> <li>■ <code>xrun argument compcnfg</code></li> <li>■ <code>config.sv</code> file</li> <li>■ <code>textInputs</code> file</li> </ul> <p>Additionally:</p> <ul style="list-style-type: none"> <li>■ <code>xrunArgs</code> includes the <code>lib.map</code> file.</li> <li>■ <code>xrunArgs_vy</code> includes the <code>hdl.var</code> and <code>cdslib</code> or <code>cds_xrun.lib</code> file.</li> </ul> <p>If you specify an <code>hdl.var</code> file in the <i>Miscellaneous</i> tab of the SystemVerilog Netlister Options form, this file is included in the <code>hdl.var</code> file that SystemVerilog Netlister generates.</p> <p><b>Note:</b> You do not need to select this check box if you use a third-party simulation tool.</p>
<i>Print global timescale</i>	<p>Prints the global timescale. By default, SystemVerilog Netlister does not print the timescale globally.</p> <p>Enabling <i>Print global timescale</i> prints the following:</p> <ul style="list-style-type: none"> <li>■ <code>'timescale</code> on the module header of each schematic cell in netlist</li> <li>■ <code>-timescale</code> in the <code>xrunArgs</code> file if <i>Create arguments file</i> is enabled</li> </ul>
<i>Global sim time</i>	<p>Specifies the simulation time in global timescale. The possible values are 1, 10, and 100. The default is 1.</p> <p>This field is enabled only when <i>Print global timescale</i> is selected.</p>

## Virtuoso SystemVerilog Netlister User Guide

### SystemVerilog Netlister Forms

---

Field	Description
<i>Unit for global sim time</i>	<p>Specifies the unit to be used for simulation time in global timescale. The possible values are <i>s</i>, <i>ms</i>, <i>us</i>, <i>ns</i>, <i>ps</i>, <i>fs</i>. The default is <i>ns</i>.</p> <p>This field is enabled only when <i>Print global timescale</i> is selected.</p>
<i>Global sim precision</i>	<p>Specifies the simulation precision in global timescale. The possible values are 1, 10, and 100. The default is 1.</p> <p>This field is enabled only when <i>Print global timescale</i> is selected.</p>
<i>Unit for global sim precision</i>	<p>Specifies the unit to be used for simulation precision in global timescale. The possible values are <i>s</i>, <i>ms</i>, <i>us</i>, <i>ns</i>, <i>ps</i>, <i>fs</i>. The default is <i>ns</i>.</p> <p>This field is enabled only when <i>Print global timescale</i> is selected.</p>
<i>Additional Arguments</i>	<p>Specifies additional <code>xrun</code> arguments.</p>

---

### ***Related Topics***

[Importing a SystemVerilog Package File](#)

[Specifying Additional xrun Arguments in SystemVerilog Netlister](#)

[simVerilogGenerateSingleNetlistFile](#)

[hnlVerilogDumpIncludeFilesInNetlist](#)



## Verilog Tab

The following table describes the fields available on the *Verilog* tab of the SystemVerilog Netlister Options form.

Field	Description
<i>Create ATPG compatible netlist</i>	Enables the creation of ATPG compatible netlists in Virtuoso SystemVerilog Netlister.
<i>Use tran statement for aliasing</i>	Prints <code>tran</code> statements for aliases or patches instead of using the <code>cds_alias</code> and <code>cds_thru</code> and <code>cds_thrualias</code> constructs. All buses and concatenated bundles are expanded and printed in <code>tran</code> statements.
<i>Text source compliance with xrun</i>	<p>Checks the Verilog IEEE Standard compliance for text source in the design using Verilog command-line compilation options.</p> <p>Possible values are as follows:</p> <ul style="list-style-type: none"> <li>■ <i>None</i>: Compiles text source files using <code>xrun</code> default standard.</li> <li>■ <i>Verilog-2001</i>: Compiles text source files using the <code>xrun</code> option, <code>-v2001</code>.</li> <li>■ <i>Verilog-1995</i>: Compiles text source files using the <code>xrun</code> option, <code>-v1995</code>.</li> </ul> <p>The selected option is added to the <code>xrun</code> command line for <code>mapi</code> compilation during netlisting. It is also printed in the <code>xrunArgs</code> files if the <i>Create arguments file</i> option is enabled.</p>
<i>Add prefix to cell names</i>	<p>Adds a predefined prefix to cell names in the generated netlist.</p> <p>Possible values are as follows:</p> <ul style="list-style-type: none"> <li>■ <i>None</i>: Default. Does not add prefix to cell names.</li> <li>■ <i>All</i>: Adds the prefix specified in the <i>Cell prefix name</i> field to all cell names, including the top cell and non-textual stopping cells. These non-textual stopping cells are cells without module definitions.</li> <li>■ <i>Keep name of top cell and symbol cells</i>: Adds the prefix specified in the <i>Cell prefix name</i> field to all cell names, excluding names of the top cell and non-textual stopping cells. These non-textual stopping cells are cells without module definitions.</li> </ul>

## Virtuoso SystemVerilog Netlister User Guide

### SystemVerilog Netlister Forms

---

Field	Description
<i>Cell prefix name</i>	Specifies a prefix string for cell names. The prefix must start with either an alphabetic character or an underscore and the remaining string may contain only alphanumeric characters or underscores.

#### ***Related Topics***

[ATPG Compatible Verilog Netlists](#)

[Generating ATPG Compatible Verilog Netlists](#)

[enableVerilogATPG](#)

[useTranForCdsAliasThru](#)

[vlogCompatVersion](#)

[addPrefixToCells](#)

[cellPrefixName](#)

## Environment Variables

This chapter describes the environment variables that control the characteristics of the SystemVerilog Netlister environment.

### Compiler

<u>createXrunArgs</u>	<u>createXrunBinding</u>	<u>hdlVarFile</u>
<u>refLib</u>	<u>reUseHdlCompilationSetup</u>	

### Elaborator

<u>enableTimeScale</u>	<u>simPrecision</u>	<u>simPrecisionUnit</u>
<u>simTime</u>	<u>simTimeUnit</u>	

### NC-Verilog

<u>addArgTableList</u>	<u>vlogCompatVersion</u>
------------------------	--------------------------

### Netlister

<u>addPrefixToCells</u>	<u>cellPrefixName</u>	<u>defaultNettype</u>
<u>enableDataPropagate</u>	<u>enableVerilogATPG</u>	<u>expandIterateInst</u>
<u>isPortInANSIFormat</u>	<u>mergedNetlist</u>	<u>nettypesToIgnore</u>
<u>termMismatch</u>	<u>termDirectionMismatch</u>	<u>useTranForCdsAliasThru</u>
<u>vlogSupply0Sigs</u>	<u>vlogSupply1Sigs</u>	

## addArgTableList

```
digitalSim.ncverilogOpts addArgTableList string ""
```

### Description

Controls the default setting for the *Create arguments file* check box in the SystemVerilog Netlister Options form.

The default is " ".

### GUI Equivalent

Command	<i>Open Options Form</i>
Form Field	<i>Create arguments file</i>

### Examples

```
envGetVal("digitalSim.ncverilogOpts" "addArgTableList")  
envSetVal("digitalSim.ncverilogOpts" "addArgTableList" 'string "argList1")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## addPrefixToCells

```
digitalSim.netlisterOpts addPrefixToCells string { "None" | "All" | "Keep name of  
top cell and symbol cells" }
```

### Description

Adds a predefined prefix to cell names in the generated netlist.

Possible values are as follows:

- **None:** Default. Does not add a prefix to cell names, even if it is defined using the *Cell prefix name* field or the `cellPrefixName` environment variable.
- **All:** Adds the prefix specified using the *Cell prefix name* field or the `cellPrefixName` environment variable to all cell names, including the top cell and non-textual stopping cells. These non-textual stopping cells are cells without module definitions.
- **Keep name of top cell and symbol cells:** Adds the prefix specified in the *Cell prefix name* field or the `cellPrefixName` environment variable to all cell names, excluding names of the top cell and non-textual stopping cells. These non-textual stopping cells are cells without module definitions.

The default is "None".

### GUI Equivalent

Command	<i>Open Options Form</i>
Form Option	<i>Add prefix to cell names</i>

### Examples

```
envGetVal("digitalSim.netlisterOpts" "addPrefixToCells")  
envSetVal("digitalSim.netlisterOpts" "addPrefixToCells" 'string "All")  
envSetVal("digitalSim.netlisterOpts" "addPrefixToCells" 'string "Keep name of top  
cell and symbol cells")
```

### Related Topics

[Configuring Netlist Generation Options](#)

## cellPrefixName

```
digitalSim.netlisterOpts cellPrefixName string { "prefixstring" }
```

### Description

Specifies a prefix for cell names. The prefix must start with either an alphabetic character or an underscore and the remaining string may contain only alphanumeric characters or underscores.

This setting applies to the selection made using the *Add prefix to cell names* check box in the Verilog tab of the SystemVerilog Netlister Options form or the [addPrefixToCells](#) environment variable.

The default is " ".

### GUI Equivalent

Command      *Open Options Form*

Form Option    *Cell prefix name*

### Examples

```
envGetVal("digitalSim.netlisterOpts" "cellPrefixName")  
envSetVal("digitalSim.netlisterOpts" "cellPrefixName" 'string "cds_")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## createXrunArgs

```
digitalSim.compilerOpts createXrunArgs boolean { t | nil }
```

### Description

Controls the default setting for the *Create arguments file* check box in the SystemVerilog Netlister Options form.

The default is `nil`.

### GUI Equivalent

Command	<i>Open Options Form</i>
Form Field	<i>Create arguments file</i>

### Examples

```
envGetVal("digitalSim.compilerOpts" "createXrunArgs")  
envSetVal("digitalSim.compilerOpts" "createXrunArgs" 'boolean t)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## createXrunBinding

```
digitalSim.compilerOpts createXrunBinding boolean { t | nil }
```

### Description

Creates the `hdl.var` and `cds_xrun.lib` files in the netlist directory when you use `xrun`.

The default is `nil`.

### GUI Equivalent

Command      *Open Options Form*

Form Field    *Create binding files for xrun only*

### Examples

```
envGetVal("digitalSim.compilerOpts" "createXrunBinding")  
envSetVal("digitalSim.compilerOpts" "createXrunBinding" 'boolean t)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)



### defaultNettype

```
digitalSim.netlisterOpts defaultNettype string { "interconnect" | "wire" }
```

#### Description

Sets the default net type to netlist nets on the schematic.

- `interconnect` prints all interconnects in the netlist.
- `wire` prints all wires and ports in the netlist.

The default is `interconnect`.

#### GUI Equivalent

Command      *Open Options Form – Netlister*

Form Field    *Default Nettype*

#### Examples

```
envGetVal("digitalSim.netlisterOpts" "defaultNettype")
```

```
envSetVal("digitalSim.netlisterOpts" "defaultNettype" 'string "wire")
```

#### ***Related Topics***

[Configuring Netlist Generation Options](#)

## enableDataPropagate

```
digitalSim.netlisterOpts enableDataPropagate boolean { t | nil }
```

### Description

Controls the datatype propagation in SystemVerilog Netlister. The default is `nil`.

### GUI Equivalent

Command      *Open Options Form – Netlister*

Form Field    *Enable Datatype Propagation*

### Examples

```
envGetVal("digitalSim.netlisterOpts" "enableDataPropagate")
envSetVal("digitalSim.netlisterOpts" "enableDataPropagate" 'boolean t)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## enableVerilogATPG

```
digitalSim.netlisterOpts enableVerilogATPG boolean { t | nil }
```

### Description

Enables the creation of ATPG compatible netlists in Virtuoso SystemVerilog Netlister.

The default is `nil`.

### GUI Equivalent

Command      *Open Options Form – Verilog*

Form Field    *Create ATPG compatible netlist*

### Examples

```
envGetVal("digitalSim.netlisterOpts" "enableVerilogATPG")
envSetVal("digitalSim.netlisterOpts" "enableVerilogATPG" 'boolean t)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## expandIterateInst

```
digitalSim.netlisterOpts expandIterateInst boolean { t | nil }
```

### Description

Prints all iterated instances in expanded form in the netlist.

The default is `nil`.

### GUI Equivalent

Command      *Open Options Form – Netlister*

Form Field    *Expand Iterated Instances*

### Examples

```
envGetVal("digitalSim.netlisterOpts" "expandIterateInst")  
envSetVal("digitalSim.netlisterOpts" "expandIterateInst" 'boolean t)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## enableTimeScale

```
digitalSim.elabOpts enableTimeScale boolean { t | nil }
```

### Description

Enables printing the global time scale.

The default is `nil`.

### GUI Equivalent

Command      *Open Options Form – Miscellaneous*

Form Field    *Print Global Time Scale*

### Examples

```
envGetVal("digitalSim.elabOpts" "enableTimeScale")  
envSetVal("digitalSim.elabOpts" "enableTimeScale" 'boolean t)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

### hdlVarFile

```
digitalSim.compilerOpts hdlVarFile string { "pathName" }
```

### Description

Specifies the location of the `hdl.var` file.

The default is `" "`.

### GUI Equivalent

Command      *Open Options Form – Miscellaneous*

Form Field    *hdl.var file*

### Examples

```
envGetVal("digitalSim.compilerOpts" "hdlVarFile")
envSetVal("digitalSim.compilerOpts" "hdlVarFile" 'string "/myDesigns/user1/
hdl.var")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

[Xcelium XRUN User Guide](#)

### isPortInANSIFormat

```
digitalSim.netlisterOpts isPortInANSIFormat string { "Non-ANSI" | "ANSI" }
```

#### Description

Controls if the port declaration is in *Non-ANSI* or *ANSI* format.

The default is `Non-ANSI`.

#### GUI Equivalent

Command      *Open Options Form – Netlister*

Form Field    *Port Declaration Format*

#### Examples

```
envGetVal("digitalSim.netlisterOpts" "isPortInANSIFormat")
```

```
envSetVal("digitalSim.netlisterOpts" "isPortInANSIFormat" 'string "ANSI")
```

#### ***Related Topics***

[Configuring Netlist Generation Options](#)

## mergedNetlist

```
digitalSim.netlisterOpts mergedNetlist boolean { t | nil }
```

### Description

Merges the text files to create a single and self-contained netlist. This netlist includes the netlist from the schematic and the original source files from the text view.

The default is `nil`.

### GUI Equivalent

Command	<i>Open Options Form – Netlister</i>
Form Field	<i>Merge text source to single netlist</i>

### Examples

```
envGetVal("digitalSim.netlisterOpts" "mergedNetlist")  
envSetVal("digitalSim.netlisterOpts" "mergedNetlist" 'boolean t)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)



## **nettypesToIgnore**

```
digitalSim.netlisterOpts nettypesToIgnore string ""
```

### **Description**

Controls the net types to be ignored.

The default is " ".

### **GUI Equivalent**

Command      *Open Options Form – Netlister*

Form Field    *netTypes to ignore on schematic nets*

### **Examples**

```
envGetVal("digitalSim.netlisterOpts" "nettypesToIgnore")  
envSetVal("digitalSim.netlisterOpts" "nettypesToIgnore" 'string "wired")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## refLib

```
digitalSim.compilerOpts refLib string ""
```

### Description

Allows the selection of a pre-compiled library from a specific location. If you have a package definition, you can pre-compile the package into a single library or multiple libraries.

The default is " ".

### GUI Equivalent

Command	<i>Open Options Form – Netlister</i>
Form Field	<i>Pre-compiled libraries (-reflib)</i>

### Examples

```
envGetVal("digitalSim.compilerOpts" "refLib")  
envSetVal("digitalSim.compilerOpts" "refLib" 'string "/myDesigns/user1/")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## **reUseHdlCompilationSetup**

```
digitalSim.compilerOpts reUseHdlCompilationSetup boolean { t | nil }
```

### **Description**

Controls whether the HDL package setup is reused by Virtuoso SystemVerilog Netlister.

The default is `nil`.

### **GUI Equivalent**

Command      *Open Options Form – Miscellaneous*

Form Field    *Reuse HDL package setup*

### **Examples**

```
envGetVal("digitalSim.compilerOpts" "reUseHdlCompilationSetup")  
envSetVal("digitalSim.compilerOpts" "reUseHdlCompilationSetup" 'boolean t)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## **simPrecision**

```
digitalSim.elabOpts simPrecision int pre-defined_numeric_value
```

### **Description**

Specifies the simulation precision in global timescale. Possible values are 1, 10, and 100.

The default is 1.

### **GUI Equivalent**

Command      *Open Options Form – Miscellaneous*

Form Field    *Global sim precision*

### **Examples**

```
envGetVal("digitalSim.elabOpts" "simPrecision")  
envSetVal("digitalSim.elabOpts" "simPrecision" 'int 100)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

### simPrecisionUnit

```
digitalSim.elabOpts simPrecisionUnit string { "s" | "ms" | "us" | "ns" | "ps" |  
"fs" }
```

### Description

Specifies the unit to be used for simulation precision in global timescale.

The default is "ns".

### GUI Equivalent

Command	<i>Open Options Form – Miscellaneous</i>
Form Field	<i>Unit for global sim precision</i>

### Examples

```
envGetVal("digitalSim.elabOpts" "simPrecisionUnit")  
envSetVal("digitalSim.elabOpts" "simPrecisionUnit" 'string "s")  
envSetVal("digitalSim.elabOpts" "simPrecisionUnit" 'string "ms")  
envSetVal("digitalSim.elabOpts" "simPrecisionUnit" 'string "us")  
envSetVal("digitalSim.elabOpts" "simPrecisionUnit" 'string "ps")  
envSetVal("digitalSim.elabOpts" "simPrecisionUnit" 'string "fs")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## simTime

`digitalSim.elabOpts simTime int pre-defined_numeric_value`

### Description

Specifies the simulation time in global timescale. Possible values are 1, 10, and 100.

The default is 1.

### GUI Equivalent

Command      *Open Options Form – Miscellaneous*

Form Field    *Global sim time*

### Examples

```
envGetVal("digitalSim.elabOpts" "simTime")  
envSetVal("digitalSim.elabOpts" "simTime" 'int 100)
```

### ***Related Topicss***

[Configuring Netlist Generation Options](#)

### simTimeUnit

```
digitalSim.elabOpts simTimeUnit string { "s" | "ms" | "us" | "ns" | "ps" | "fs" }
```

### Description

Specifies the unit to be used for simulation time in global timescale.

The default is "ns".

### GUI Equivalent

Command      *Open Options Form – Miscellaneous*

Form Field    *Unit for global sim time*

### Examples

```
envGetVal("digitalSim.elabOpts" "simTimeUnit")
envSetVal("digitalSim.elabOpts" "simTimeUnit" 'string "s")
envSetVal("digitalSim.elabOpts" "simTimeUnit" 'string "ms")
envSetVal("digitalSim.elabOpts" "simTimeUnit" 'string "us")
envSetVal("digitalSim.elabOpts" "simTimeUnit" 'string "ps")
envSetVal("digitalSim.elabOpts" "simTimeUnit" 'string "fs")
```

### ***Related Topicss***

[Configuring Netlist Generation Options](#)

## **termMismatch**

```
digitalSim.netlisterOpts termMismatch string { "ignore" | "warning" | "error" }
```

### **Description**

Controls the tool behavior when terminal mismatch is found between the switch master and the place master. The valid values are ignore, warning, and error,

- **ignore**: Ignores any terminal mismatch found between the switch master and place master in the text view and continues netlisting.
- **warning**: Displays a warning message when terminal mismatch is found between the switch master and place master in the text view and continues netlisting.
- **error**: Displays an error if terminal mismatch is found between the switch master and place master in the text view and stops netlisting.

The default is **error**.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("digitalSim.netlisterOpts" "termMismatch")
envSetVal("digitalSim.netlisterOpts" "termMismatch" 'string "ignore")
envSetVal("digitalSim.netlisterOpts" "termMismatch" 'string "warning")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)



## **termDirectionMismatch**

```
digitalSim.netlisterOpts termDirectionMismatch string { "ignore" | "warning" |  
    "error" }
```

### **Description**

Controls the tool behavior when mismatch in terminals directions is found between the switch master and the place master. The valid values are ignore, warning, and error,

- **ignore**: Ignores any mismatch in terminals directions found between the switch master and place master in the text view and continues netlisting.
- **warning**: Displays a warning message when mismatch in terminals directions is found between the switch master and place master in the text view and continues netlisting.
- **error**: Displays an error if mismatch in terminals directions is found between the switch master and place master in the text view and stops netlisting.

The default is **error**.

### **GUI Equivalent**

None

### **Examples**

```
envGetVal("digitalSim.netlisterOpts" "termDirectionMismatch")  
envSetVal("digitalSim.netlisterOpts" "termDirectionMismatch" 'string "ignore")  
envSetVal("digitalSim.netlisterOpts" "termDirectionMismatch" 'string "warning")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## useTranForCdsAliasThru

```
digitalSim.netlisterOpts useTranForCdsAliasThru boolean { t | nil }
```

### Description

Prints `tran` statements for aliases or patches instead of using the `cds_alias` and `cds_thru` constructs. This option is only available when the *Create ATPG compatible netlist* option is enabled on the *Verilog* tab of the Virtuoso SystemVerilog Netlister Options form.

The default is `nil`.

### GUI Equivalent

Command	<i>Open Options Form – Verilog</i>
Form Field	<i>Use tran statement for aliasing</i>

### Examples

```
envGetVal("digitalSim.netlisterOpts" "useTranForCdsAliasThru")
envSetVal("digitalSim.netlisterOpts" "useTranForCdsAliasThru" 'boolean t)
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## **vlogCompatVersion**

```
digitalSim.ncverilogOpts vlogCompatVersion string { "None" | "Verilog-2001" |  
    "Verilog-1995" }
```

### **Description**

Checks the text source compliance with Verilog IEEE Standard in the design using Verilog command-line compilation options.

Possible values are:

- *None*: Compiles text source files using `xrun` default standard.
- *Verilog-2001*: Compiles text source files using the `xrun` option, `-v2001`.
- *Verilog-1995*: Compiles text source files using the `xrun` option, `-v1995`.

The default is `None`.

### **GUI Equivalent**

Command	<i>Open Options Form – Verilog</i>
Form Field	<i>Text source compliance with xrun</i>

### **Examples**

```
envGetVal("digitalSim.ncverilogOpts" "vlogCompatVersion")  
envSetVal("digitalSim.ncverilogOpts" "vlogCompatVersion" 'string "Verilog-2001")  
envSetVal("digitalSim.ncverilogOpts" "vlogCompatVersion" 'string "Verilog-1995")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## **vlogSupply0Sigs**

```
digitalSim.netlisterOpts vlogSupply0Sigs string "name_of_global_ground_net"
```

### **Description**

Specifies the global ground net.

The default is " ".

### **GUI Equivalent**

Command	<i>Open Options Form – Netlister</i>
Form Field	<i>supply0</i>

### **Examples**

```
envGetVal("digitalSim.netlisterOpts" "vlogSupply0Sigs")  
envSetVal("digitalSim.netlisterOpts" "vlogSupply0Sigs" 'string "I18")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

## **vlogSupply1Sigs**

```
digitalSim.netlisterOpts vlogSupply1Sigs string "name_of_global_power_net"
```

### **Description**

Specifies the global power net.

The default is " ".

### **GUI Equivalent**

Command	<i>Open Options Form – Netlister</i>
Form Field	<i>supply1</i>

### **Examples**

```
envGetVal("digitalSim.netlisterOpts" "vlogSupply1Sigs")  
envSetVal("digitalSim.netlisterOpts" "vlogSupply1Sigs" 'string "V2")
```

### ***Related Topics***

[Configuring Netlist Generation Options](#)

# **Virtuoso SystemVerilog Netlister User Guide**

## **Environment Variables**

---

---

## SKILL Functions and Flags

---

This section provides syntax, descriptions, and examples for the Cadence SKILL functions and SKILL flags associated with the Virtuoso SystemVerilog Netlister (SystemVerilog Netlister) flow.

Only the functions documented in this chapter are supported for public use. Any other functions, and undocumented aspects of the functions described below, are private and subject to change or removal at any time.

### SKILL Functions

- [asiDigitalSimAutoloadProc](#)
- [simDeRegPostNetlistTrigger](#)
- [simDeRegPreNetlistTrigger](#)
- [simPostNetlistTriggerList](#)
- [simPreNetlistTriggerList](#)
- [simRegPostNetlistTrigger](#)
- [simRegPreNetlistTrigger](#)

## **asiDigitalSimAutoloadProc**

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

### **Description**

Automatically loads the context for class and tool initialization when you launch SystemVerilog Netlister.

### **Arguments**

None

### **Value Returned**

t	The function call is successful.
nil	The function is unsuccessful.

### **Example**

The following example shows using the function:

```
asiDigitalSimAutoloadProc( )  
=> t
```



## **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**

<i>S_triggerFunc</i>	Name of SKILL function symbol.
----------------------	--------------------------------

### **Value Returned**

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

### **Example**

```
simDeRegPostNetlistTrigger( 'S_triggerFunc )
```

## **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**

<i>S_triggerFunc</i>	Name of SKILL function symbol.
----------------------	--------------------------------

### **Value Returned**

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

### **Example**

```
simDeRegPreNetlistTrigger( 'S_triggerFunc )
```

## **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**

<i>l_triggerFunc</i>	List of functions registered using <code>simRegPostNetlistTrigger</code> .
<code>nil</code>	There are no functions registered using <code>simPostNetlistTriggerList</code> .

### **Example**

```
simPostNetlistTriggerList()
```

## **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**

<i>l_triggerFunc</i>	List of functions registered using <code>simRegPostNetlistTrigger</code> .
<code>nil</code>	There are no functions registered using <code>simPostNetlistTriggerList</code> .

### **Example**

```
simPreNetlistTriggerList()
```

## **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**

<i>S_triggerFunc</i>	Name of SKILL function symbol.
----------------------	--------------------------------

### **Value Returned**

t	The function was registered.
nil	The function could not be registered.

### **Example**

```
procedure(postNetlistTrigger()  
    printf("\n\npostNetlistTrigger is called\n\n")  
)  
simRegPostNetlistTrigger('postNetlistTrigger)
```

## **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**

<i>S_triggerFunc</i>	Name of SKILL function symbol.
----------------------	--------------------------------

### **Value Returned**

t	The function was registered.
nil	The function could not be registered.

### **Example**

**Setting useMfactorToIterateInstances using simRegPreNetlistTrigger**

```
procedure (setUseMfactorFlag()  
    fnlMfactorPropertyName="m"  
    useMfactorToIterateInstances = "OnSubckt"  
)  
simRegPreNetlistTrigger('setUseMfactorFlag)
```

## SKILL Flags

---

`hnlUseSchematicForSystemVerilogView`

---

Supports printing inherited connections from the schematic when the instances are bound to a SystemVerilog view. To know more, see [hnlUseSchematicForSystemVerilogView](#).

---

# **Virtuoso SystemVerilog Netlist User Guide**

## **SKILL Functions and Flags**

---



---

## Batch Mode Options in SystemVerilog Netlister

---

SystemVerilog Netlister can be launched in batch mode using the following commands:

runsv

cdsCreateConfig

si2runsv

## runsv

Use the `runsv` command with the following options:

```
runsv
    [-help | -version | -W]
    | runsv
    -lib <libName>
    -cell <cellName>
    -view <viewName>
    action_options
    [setup_options]
    [netlisting_options]
    [simulation_options]
```

The following table describes the various options that can be used with the `runsv` command.

Option	Description
<code>-help</code>	Displays the tool help.
<code>-version</code>	Prints the program version.
<code>-W</code>	Prints the program sub-version.
<code>-usage</code>	Displays the syntactical usage to indicate how the tool works under various conditions.
<code>-nocdsinit</code>	Skips reading the <code>.cdsinit</code> file and uses the SKILL search path.
<code>-lib &lt;libName&gt;</code>	Specifies the library name of the configuration.
<code>-cell &lt;cellName&gt;</code>	Specifies the cell name of the configuration.
<code>-view &lt;viewName&gt;</code>	Specifies the view name of the configuration.

### Action Options

<code>-netlist [ skip   incremental   all ]</code>	Runs the netlister in the specified mode. Default: <code>incremental</code> .
<code>-log &lt;logFileName&gt;</code>	Writes the output log messages to <code>&lt;logFileName&gt;</code> Default: <code>./runsv.log</code> .
<code>-rundir &lt;runDir&gt;</code>	Specifies the run directory path to use. Netlisting takes place in <code>&lt;runDir&gt;/netlist</code> .

`-cdsenv <filePath>` Specifies the `.cdsenv` file to use.

## Example

To generate the netlist using the configuration `mylib.top:config` with the settings specified in the `.cdsenv` file, use the following batch mode command:

```
runsv -lib mylib -cell top -view config -netlist -rundir top_run1 -cdsenv .cdsenv
```



### Tip

By default, environment variables specified in file paths are expanded in the `runSimulation` file which is created by the `runsv` command. If you want the `runSimulation` file to be portable, enclose the file paths with the `'` character so that the environment variables in the file paths are not expanded in the `runSimulation` file. The relative paths specified using the `~` or `.` characters are resolved with respect to the directory from which you run the `runsv` command.

## Related Topics

[SystemVerilog Netlister Batch Mode](#)

## cdsCreateConfig

Use the `cdsCreateConfig` utility with the following options:

```
cdsCreateConfig
  -lib topLibName
  -cell topCellName
  -view topViewName
  [-config newConfigName]
  [-liblist 'l_lib']
  [-viewlist 'l_view']
  [-stoplist 'l_stopview']
```

The following table describes the various options that can be used with the `cdsCreateConfig` utility.

Option	Description
-help	Displays the <code>cdsCreateConfig help</code> .
-V   -version	Prints the utility version.
-W	Prints the utility sub-version.
-lib < <i>libName</i> >	Specifies the top library name of the configuration.
-cell < <i>cellName</i> >	Specifies the top cell name of the configuration.
-view < <i>viewName</i> >	Specifies the top view name of the configuration.

### Optional Options

-config <" <i>newConfigName</i> ">	Specifies the name of the config view that needs to be created.
-liblist <' <i>lib1 lib2 ...</i> '>	Specifies the names of the libraries that are bound to the new config view.
-viewlist <' <i>view1 view2 ...</i> '>	Specifies the names of the views that are bound to the new config view.
-stoplist <' <i>stopview1 stopview2 ...</i> '>	Specifies the names of the stop views that are bound to the new config view.

***Related Topics***

[SystemVerilog Netlister Batch Mode](#)

[SystemVerilog Config Views](#)

## si2runsv

The `si2runsv` utility can be used with the following options:

```
si2runsv  
    [run directory]  
    [-batch [-command commandName]]  
    [-cdslib path]
```

The following table describes the various options that can be used with the `si2runsv` command.

Option	Description
<code>-help</code>   <code>-h</code>	Displays the <code>si2runsv</code> help.
<code>-V</code>   <code>-version</code>	Prints the utility version.
<code>-W</code>	Prints the program sub-version.
<code>&lt;run directory&gt;</code>	Specifies the name of the run directory. Ensure that the run directory includes the <code>si.env</code> file with the necessary configurations.
<code>-batch</code>	Specifies the batch simulation mode.
<code>-command &lt;commandName&gt;</code>	<p>Specifies the command to generate a netlist or simulate a design. This option can only be specified with the <code>-batch</code> option.</p> <p>You typically use the following option with <code>-command</code> for SystemVerilog designs:</p> <pre>-command netlist</pre> <p>This instructs the simulation system to netlist the design.</p>
<code>-cdslib path</code>	Specifies the path where the <code>.cdslib</code> file is located.

### ***Related Topics***

[SystemVerilog Netlister Batch Mode](#)

[Migrating SystemVerilog Integration Designs to SystemVerilog Designs](#)