Netgen 1.5

Table of Contents

Using Netgen version 1.5:

Command-line invocation:

netgen [**-noconsole**] [*command_line*]
netgen -batch *command_line*

With no arguments, **netgen** will bring up a Tk console window with an interpreter prompt, waiting for command input from the user (see command set, below).

With argument **-noconsole**, there will be no console window, and the interpreter prompt will be in the terminal.

For Netgen version 1.4 and eariler, the *command_line* argument is unavailable. With netgen version 1.5 and later, any valid netgen (or Tcl/Tk) command may be passed through *command_line* and operates as a type of batch-mode processing; for example,

netgen lvs circuit1.spc circuit2.spc

For batch-mode processing of multiple commands, put the commands into a file and source that file through the command line; for example,

netgen source batch_script.tcl

Note that a batch script will end by returning to the Tcl interpreter unless it ends with a "**quit**" command, or if the command-line argument **-batch** is passed to netgen. The **-batch** command implies "noconsole" mode, and exits after executing the *command_line* argument. If no *command_line* is present, then netgen prints version information and exits.

Versions of Netgen compiled without Tcl will not have a console window and will simply generate a prompt at the terminal, and will accept a limited set of single-character commands. Because the Tcl-based version is vastly superior, usage information on this website will primarily focus on that version.

Using Tcl-based Netgen:

The Tcl command-line commands completely replace the original command-line interface, which consisted on one-key entries in a hierarchically stacked format, difficult to use. I have removed the interface and replaced it with Tcl command-line commands plus a few scripts. These are outlined below. The original X11 interface was a complete and unnecessary disaster and has been removed. With the Tcl interface, it could presumably be rewritten in Tk, although personally, I can't see why anybody would want to.

The most common use of "netgen" is to make use of its network comparison feature to compare a layout generated from **magic** with "extract" and "ext2spice" commands against a layout generated from a schematic (e.g., **xcircuit**). Starting with version 1.2, there is a script-level command which performs the sequence of commands above. This command is:

- `lvs` *circuit1 circuit2* [*setup_file* [*logfile* [*options*]]]
- `lvs {` *file1 cell1*`} {` *file2 cell2*`}` [*setup_file* [*logfile* [*options*]]]

Arguments *circuit1* and *circuit2* are both filename and cellname, and assume that each file contains a top-level cell that is not defined as a subcircuit (thus the filename becomes the name of the top-level cell). The alternative syntax is to give, for one or both of the circuits, a separate *file* and *cell* name. This covers cases where the file does not contain a top-level cell, and also cases in which one wants to run LVS starting at a level of hierarchy lower than the top. *logfile* is the file to dump all output, and defaults to "`comp.out`" if not specified. Note that certain useful information, such as the initial report on the two files, and the final "yea or nay" result, is duplicated both to the log file and to the console (or terminal) window. The output file should be consulted for all detailed information about netlist errors. The optional file *setup_file* is a Tcl-format file containing any commands that should be run after both files are read in but before the comparison is made. Typically, this will contain information on how to map device model names or subcircuit names between the two files, and how to permute pins on subcircuits.

Note that in the second form above, due to the vagaries of Linux shell syntax, when this command is passed to the "**netgen -batch** *command*" option, the braces should be replaced with quotes, as in:

> `netgen -batch lvs "`*file1 cell1*`" "`*file2 cell2*`"` [*setup_file* [*logfile* [*options*]]]

setup_file can be the word **nosetup** to avoid reading any setup file at all.
logfile can be the word **nolog** to avoid writing any log output at all.

"*options*" may be zero or more options from the list below:

> Option **-blackbox** treats empty subcircuits as "black boxes" rather than cells to be optimized away. The option is the same as running the "model blackbox on" command from the script file (see the **model** command description).
>
> Option **-json** makes netgen generate an output file "comp.json" in addition to the normal text output file. The file is in JSON format and contains all the same information as the text output file, and is appropriate for reading into, say, a python script for interactive display.
>
> Option **-list** returns the output as a hierarchical Tcl list. This is similar to the JSON format, but is a data structure that remains within the Tcl interpreter.

The most common use of *setup_file* is to map subcircuit names between the two circuits. Each line of the file should contain the command

> **equate classes** *name1 name2*

declaring that device class or subcircuit name *name1* in *circuit1* is equivalent to device class or subcircuit name *name2* in *circuit2*. If the *setup_file* is not present, then the default setup file name "**setup.tcl**" will be read, if it exists. If not, then every device class in *circuit1* must have the same name as that used in *circuit2*, or else the circuits will never match.

It is important to note that the **equate classes** command changes values in *name2* to match those in *name1*. Consequently, to match *N*-to-1 cells by name, where one netlist has *N* different names for the same cell, but the opposing netlist has only one cell, the netlist with the *N* names should always be the second netlist passed to the **equate classes** command. *N*-to-*M* comparisons are also possible, as long as the same cell name always appears as the first argument; all others will be modified to match.

In SPICE files, each semiconductor device has a **model** name, which netgen uses as the device class. Devices with different models are compared independently. In addition, if no setup file is specified, then default permutations are allowed (transistor sources and drains permute, and resistor endpoints permute).

The command sequence used by the **lvs** command to compare two netlists is as follows:

- `readnet` *circuit1*
- `readnet` *circuit2*
- `source` *setup_file*
- `compare hierarchical` *circuit1 circuit2*
- `permute`
- `run converge`

Arguments *circuit1* and *circuit2* are filenames. They do not need to have the file extension if a standard extention (.sim, .spc, .spice) is used, but do need to contain any directory path components (relative to the current working directory).

For circuits which do *not* match, it will be necessary to run the command "`verify`" to look at the list of illegal elements and nodes.

Note that the "`verify`" command produces copious output. Generally, it is preferable to use the "`log`" command to dump this output to a file. The command "`verify only`" produces just one line stating whether or not the circuits match. The output can be terminated at any time with a Control-C interrupt into either the terminal or console windows.

Using the non-Tcl-based (legacy) Netgen:

Information on the legacy version of netgen can be found in the source `doc` directory, in the file `netgen.doc`. This text file has been formatted for a web browser and can be found here.

Standalone programs

For a long time, I did LVS by compiling netgen and using the standalone program "netcomp" for the LVS part, ignoring the rest of the "netgen" package, which largely deals with handling the loading and conversion of different netlist formats.

Version 1.3 dispenses with the standalone programs, instead making the command-line interface more accessible. See the description of Tcl commands below for details.

Tcl Command-Line Interface (for Netgen 1.5 and above)

Common options: Netgen 1.5 has a more consistent command set with common options for indicating cell names. Previous versions of netgen would not differentiate between files when searching for cells, so cells with the same could be confused between files when the cell was queried from the command line. When two cells were compared, one would have an underscore prepended to all the cell names so that they could be differentiated from cells in the other file. However, that was an arbitrary solution, and as it did not address the issue of how to tell apart cells of the same name in different files from the command line. Netgen 1.5 (from revisision 9) takes a different approach of having a consistent interface in which cells must always be uniquely identified. The **readnet** command returns a number (not the operating system's file number) that can be used to identify the file and all of its cells in subsequent commands. However, there are several ways that a cell can be identified from the command line, as long as the method ensures that the cell name is unique. Any of the following ways will work:

cellname
: The cellname alone identifies the cell if there are no other cells of the same name belonging to other files.

{*cellname file_number*}
: The *file_number* is the value returned from the **readnet** command that loaded the file containing *cellname*. When a cell is identified by a pair of values like this, they must be presented to the Tcl interpreter as a list, which means that they must be enclosed in braces (double-quotes will also work).

{*file_number cellname*}
: In a two-item list, *cellname* and *file_number* may be in either order.

{*filename cellname*}

> The *filename* is the name of the file containing the cell *cellname*. When applied as an argument to the **lvs** command, this form is required if the file *filename* has not yet been read; the *filename* alone (see below) is acceptable if *cellname* is the top-level cell (that is, it is the same as *filename*).

{*cellname filename*}

> In a two-item list, *cellname* and *filename* may be in either order, unless used as an argument to the **lvs** command when *filename* has not yet been read in (see above).

filename

> If a file contains a top-level cell that is not in a subcircuit, then that top-level cell has no internal name, and netgen gives the cell the same name as the file (including the extension). So the *filename* is equivalent to the top-level cell in this case.

file_number

> Like the syntax above, a single number can refer to the top-level cell of the file that has been assigned the number *file_number*.

-circuit1

> Refers to the first of the two circuits passed to the "**lvs**" procedure, or more specifically, to the "**compare assign**" command. This implies that the "**compare assign**" command must be run prior to using this method, and likewise, that the "**compare assign**" command itself cannot use this method.

-circuit2

> Refers to the second of the two circuits passed to the "**lvs**" procedure, or more specifically, to the "**compare assign**" command. This implies that the "**compare assign**" command must be run prior to using this method, and likewise, that the "**compare assign**" command itself cannot use this method.

In the list below, the term *valid_cellname* is used to refer to any of the syntax methods above for describing a unique cell name.

The two-item list form was introduced with netgen version 1.4.13 but was not extended to include all forms, or extended for use in all commands requiring a cell name, until version 1.5.9.

Netlist manipulation commands:

canonical *valid_cellname*

> Return a 2-item list containing the cell name and the file number of the cell indicated by *valid_cellname*. See the description above for the syntax of *valid_cellname*.

readnet [*format*] *filename* [*filenum*]

> read a netlist file (default format=auto). Auto-detection first looks for files with standard extensions. If no file is found with a standard extention, then the file is checked for the initial character, which is always "*****" in SPICE files and "**|**" in .sim files.
>
> Returns the file number for the file, which can be used in other commands to reference the file and its cell hierarchy.
>
> If *filenum* is specified, then the contents of the cell are added to the netlist of file number *filenum*. This option may be used if subcells are in separate files without the use of ".include" statements. However, for reading models, standard cells, or subcircuits as "black boxes", use **readlib** instead (see below). The option may also be used to assign file numbers to netlists manually.

readlib *format* [*filenum*]

> Initialize a library module definition, where *format* is one of **actel**, **spice**, or **xilinx**. The library definitions need to be initialized prior to reading any netlist file that is to be rewritten as "**writenet actel** *filename*", "**writenet xilinx** *filename*", or as a SPICE netlist with the definitions from the library included in the file.
>
> Returns the file number for the file, which can be used in other commands to reference

the file and its cell hierarchy.

In general, a library is not useful as a standalone file, and normally *filenum* will be passed to the command to attach the library to a specific netlist. If the library contents are common to more than one netlist, then the **readlib** command should be issued once for each netlist.

writenet *format filename*

write a netlist file

flatten *valid_cellname*

flatten a hierarchical cell.

Note: Always flatten a cell in a hierarchical format (such as SPICE) before writing a flat format, even if the original netlist is already flat. Otherwise, the flattened namespace will not be cached, and the write process will incur a horrific performance hit.

flatten class [*cellname*] *valid_cellname*

In this form of the **flatten** command, the whole cell database is searched for instances of the cell *valid_cellname*, which are then flattened by merging them into the parent cell. From netgen version 1.5.53, the optional *cellname* allows instances of the cell *valid_cellname* to be flattened only within the specified parent cell. Note that "*cellname*" is not in the general form of "*valid_cellname*", but is the cell name only. The cell is assumed to be in the same circuit as *valid_cellname*.

model *valid_cellname* [*model*]

declare a cell to be specific model type. Normally, model types are derived from low-level device calls in an input file (such as SPICE). This command will override the assigned model type, assuming that the cell has the correct number of pins for the model type. The supported low-level device model types are: **nmos** (3 or 4 port), **pmos** (3 or 4 port), **pnp**, **npn**, **resistor**, **capacitor**, **diode**, **inductor**, or **xline**. A low-level device can be treated as a subcircuit by declaring *model* to be **subcircuit**. A circuit declared to be model **blackbox** will only recognize the pin names and pin order but will otherwise ignore the subcell contents.

The special value **copy** for *model* will cause the model type of the subcircuit to be taken from the other circuit being compared (this requires that the circuits have been queued for comparison already). This avoids the problem where a SPICE subcircuit is undefined and therefore classified as a black-box entry, and cannot be compared to the equivalent subcircuit in the other netlist.

Without option *model*, return the device model of the cell *valid_cellname*.

model blackbox on|off

With specific option **model blackbox**, any subcircuit read in that has no contents will be treated as a "black box" (essentially treated like a device) if this option is **on**. If **off**, the subcircuit will treated as an actual subcircuit with no contents.

nodes [[**-list**] *element*] *valid_cellname*

print nodes of an element or cell
with option **-list**, return a list of nodes to the Tcl interpreter.

elements [[**-list**] *node*] *valid_cellname*

print elements of a node or cell
with option **-list**, return a list of elements to the Tcl interpreter. Element names are returned in the format "*element_name/pin_name*".

debug on|off|*command* [*options...*]

turn debugging on or off or debug a command. When debugging is on, the output of LVS comparisons reverts to the original, list-oriented format instead of the newer side-by-side format.

protochip

embed protochip structure

instances *valid_cellname*

list instances of the cell

contents *valid_cellname*
  list contents of the cell
describe *valid_cellname*
  describe the cell
cells [**list**] [**-all**] [**-top**] [*valid_filename*]
  print known cells
  option **-all**: print all cells, including primitives. If *valid_filename* is given, then restrict the listing of cells to those belonging to the specified file.

  option **-top**: print the top-level cells. There is one top-level cell associated with each loaded file. If the loaded file is a library, the top-level cell may be empty.

  Note: *valid_filename* takes the form of syntax for *valid_cellname*; however, a cell name that is not a top-level cell will act as if the filename of the file containing that cell has been passed to the command.
ports *valid_cellname*
  print ports of the cell
leaves [*valid_cellname*]
  print leaves of the cell
quit
  exit netgen and Tcl
reinitialize
  reintialize netgen data structures
log [**file** *name*|**start**|**end**| **reset**|**suspend**|**resume**|**echo**]
  enable or disable output log to file named *file*.
help
  print basic usage information

Netlist comparison commands:

compare *valid_cellname1 valid_cellname2*
  declare two cells for netlist comparison. Subcircuits that are not declared equivalent or that do not have the same name in both cells are flattened. Otherwise, subcircuit instances will be compared as black-box devices, without comparison of their contents (if any).
compare hierarchical [*valid_cellname1 valid_cellname2*]
  do a full hierarchical comparison. The first call must declare the two top-level cells *valid_cellname1* and *valid_cellname2*. The cell hierarchy will be searched; all cells with matching names or cells that are declared equivalent will be saved in a stack. All cells whose match in the other top-level cell's hierarchy cannot be determined will be flattened. Upon the completion of the search, the top of the stack (the bottommost cells in the hierarchy) will be prepared for comparison. Subsequent calls to **compare hierarchical** should be called without any arguments. Each time **compare hierarchical** is called, the next pair of cells is taken off of the top of the stack and prepared for comparison. Once there are no cell pairs left to compare, **compare hierarchical** returns the string "`empty queue`".
compare assign [*valid_cellname1 valid_cellname2*]
  Assign netlists to be compared, but take no further action. This allows all future commands requiring *valid_cellname* to use the generic names "**-circuit1**" and "**-circuit2**" to refer to the two netlists, respectively.
iterate
  do one netlist comparison iteration
summary [**elements**|**nodes**]
  summarize internal data structures
print [**-list**] [**elements**|**nodes**|**queue**] [**legal**|**illegal**]
  print full report on the internal data structures. This is a list of the partitions generated by the comparison algorithm.
  with option **elements**, print only the element partitions.

with option **nodes**, print only the node partitions.
with option **queue**, print the comparison queue.
with option **legal**, print only legal (matching) partitions.
with option **illegal**, print only illegal (non-matching) partitions.
with option **-list**, return a nested list with the information, where each item in the list is one partition, and each item is a list of two parts, one for each circuit, and each part is a list of either element pins, or nodes. The **-list** option may only be used with either **print elements** or **print nodes**.

format [*col_width*]
format *col1_width col2_width*

With no arguments, return the column widths (in characters) of the formatted side-by-side output. With one argument *col_width*, set the column width (in characters) for each of the two columns to the value given. With two arguments *col1_width* and *col2_width*, set the column width in number of characters for the left side to *col1_width*, and the column width in number of columns for the right side to *col2_width*.

run [**converge**|**resolve**]

with option **converge**: run netlist comparison to completion (convergence)
with option **resolve**: run to completion and resolve automorphisms by arbitrary symmetry breaking.

verify [**elements**|**nodes**|**only**| **equivalent**|**unique**]

verify results

automorphisms

print automorphisms, which are sections of a circuit that are symmetrical and may match in various permutations.

equate [**-list**] [**-force**] **pins** *name1 name2*

Equate pins in the two cells being evaluated. This requires that two cells have been declared with the **compare hierarchical** command, and that LVS has run to convergence on any pair of cells in the stack (see **compare hierarchical**), and the netlists verified as matching. The **equate pins** command will sort the ports of one cell to match those of the other.
Use the [**-force**] option to force matching of pins on two cells that have been evaluated but did not match. The default policy of netgen is to flatten mismatching subcells, because this will resolve errors that arise from a difference between the hierarchy of the two netlists. However, if two cells are truly mismatched, then flattening the contents of the two cells will usually make the rest of output from LVS more difficult to understand and diagnose. The **equate -force pins** option allows the cells to remain in the circuit unmatched and unflattened, so that the comparison can continue as if the cells were matched. This option must be used carefully because the top level circuits may end up correctly matched. See the **lvs** command option **-noflatten**.

equate [**-list**] [**-unique**] [**elements**|**nodes**|**classes**] *name1 name2*

with option **elements**: equate two elements
with option **nodes**: equate two nodes
with option **classes**: equate two device classes.
In all cases the values (e.g., hash values) of cell *name2* are altered to match those of *name1*. *N*-to-1 or *N*-to-*M* equivalences are valid using multiple **equate** commands, as long as only one cell name appears in the *name1* argument position. For the **elements**, **nodes**, and **pins** uses, the name *name1* must belong to the first of the two circuits being compared and *name2* must belong to the other one. For **classes** there are no limitations on which netlist cells must be first, and cells from the same netlist can be equated. Also note that **equate classes** does not imply that that two cells *are* equal, only that they are expected to be equal but must be checked during LVS. For **elements**, **nodes**, and **pins**, the **equate** command forces a match.
If the **-unique** option is used with **equate classes**, then the two names *name1* and *name2* will be uniquely paired and will no longer match any other cells of the same name.

ignore class *valid_cellname*

Remove all instances of the device class "*valid_cellname*" from the database, and prevent any subsequent addition of any instance of the class when reading a netlist file. Use with caution! Can be useful in some cases, such as to ignore diode devices that cannot be recognized during layout extraction, or to ignore all parasitic devices in a netlist file. Available on Netgen versions 1.4.45 and higher.

ignore shorted *valid_cellname*

Remove all instances of the device class "*valid_cellname*" which have all pins shorted. This is a common configuration for "dummy" devices that are used in a layout to improve matching but are not otherwise electrically active. Because they are not active circuit elements, they may not appear in a schematic and may need to be ignored for purposes of LVS.

global *valid_cellname netname* [...]

This command makes the net named *netname* in the hierarchy of cell *valid_cellname* into a global node. The cell *valid_cellname* must exist in the database (it must have already been read into netgen). This command is typically used within the setup file to assert which nodes in a file need to be declared global. For instance, if the SPICE netlist "**file.spice**" contains a subcircuit using the default SPICE ground node name "**0**", then the setup file would need the line "**global file.spice 0**". Note that a SPICE netlist file can contain a *forward* reference to a global node name using the "**.global** *name*" card, in which case the **global** statement is not needed.

convert *valid_cellname*

This command searches cell *valid_cellname* for global nodes, and converts each one into a local node while adding that node to the pin list of the cell. The database is searched for all instances of *valid_cellname*, and the global node is passed from the parent cell to the instance by way of this pin.

permute default

This command is implicitly run at the beginning of every LVS run *unless* the setup file contains any "**permute**" command, in which case *all* permutations must be declared in the setup file. If the intent of the setup file is to modify the default values, then **permute default** should be put in the setup file before all other **permute** statements.

permute [**transistors**|**resistors**|**capacitors**]
permute [**pins**] *valid_cellname pin1 pin2*
permute forget *valid_cellname pin1 pin2*

with option **pins**: permute pins *pin1* and *pin2* on device *valid_cellname*. If more than two pins may be permuted, this command may be issued multiple times between pairs of pins.
With option **forget**: remove any permutation between pins *pin1* and *pin2* on device *valid_cellname*. This option is provided for situations where it is more convenient to first apply the default permutations and then specify the exceptions individually (netgen 1.4.67 and later only).
With option **transistors**: enable transistor source/drain permutations for any device recognized as a MOSFET.
With option **resistors**: enable permutations of resistor endpoints.
With option **capacitors**: enable permutations of capacitor top and bottom plates. Because capacitor top and bottom plates have different parasitics, capacitor plate permutation is not normally enabled.
With no options: enable transistor source/drain and resistor endpoint permutations.

property default

This command is implicitly run at the beginning of every LVS run *unless* the setup file contains any "**property**" command, in which case *all* properties must be declared in the setup file. If the intent of the setup file is to modify the default values, then **property default** should be put in the setup file before all other **property** statements.

property *valid_cellname*|*device_name*
property *valid_cellname*|*device_name* **add { ***key type tolerance* **}** ...
property *valid_cellname*|*device_name* **remove** [*key* ...]
property *valid_cellname*|*device_name* **tolerance {** *key tolerance* **}** ...

property *valid_cellname*|*device_name* **associate {** *key pin_name* **}** ...
property parallel none|all
property parallel connected|open
property series none|all
property *valid_cellname*|*device_name* **series|parallel enable|disable**
property *valid_cellname*|*device_name* **series|parallel {** *key combine_type* **}** ...
property topology strict|relaxed

with no options other than a valid cellname, return a list of triplets, each triplet being a list of three items representing a property of the cell. Items are, in order, the name (key) of the property, the type of property (which is one of **string**, **integer**, **double**, **value**, or **expression**), and the tolerance for checking (which is type-dependent). Note that properties of cells, as opposed to instances of cells, do not have values. The main purpose of the property command is to establish which device properties will be compared, and what will be the criterion for comparison. When no tolerance value is given, then the default tolerances of 0, for integer values, and 0.01 (1 percent) for floating-point values, will be assigned.
With option **add**, the full property description may be given as a triplet, or *tolerance* may be omitted, in which case the default tolerance will be used; *type* may be omitted, in which case the type will be either **string** or **double**, depending on the tolerance. Generally, the full description is preferred to avoid ambiguity.
With option **remove**, if no other option is given, the all properties will be removed from the device or cell.
Option **tolerance** takes one or more duplets as arguments. It operates on an existing property *key* for which the type has already been defined. The tolerance for the property is changed to the value *tolerance*, which is given as a fraction (see above).
Option **associate** takes one or more duplets as arguments. It is used for the special case where a property is specific to a device terminal, and the terminal is permutable. In that case, if the device terminals are swapped to make a netlist match, then the corresponding properties need to be swapped as well. An example is the **ad** and **as** properties fore FET area of drain and source, respectively. The *key* in the duplet is the name of the property, and the *pin_name* in the duplet is the pin to associate with the property.
With option **series** or **parallel**, the device is declared to be able to be combined in parallel or in series, depending on the additional arguments. By default, all devices are allowed to be combined in parallel, while only resistors and inductors are allowed to be combined in series. This command can be used to override the defaults and set specific behavior.
The argument *combine_type* may be one of: **none**, **add**, **par**, or **critical**. "**critical**" implies that the property *key* must be the same on every device in order for the devices to be combined. Critical properties are never combined for the type of combining for which they are defined. For example, to combine transistors in parallel, key "**L**" (length) is "critical": All transistors must have the same length before they can be combined in parallel by adding the widths together.
Properties specifying **add** will combine by addition, while properties specifying **par** will combine in parallel (i.e., the inverse of the sum of the inverses of the components).
Another example: default semiconductor resistor behavior is for length to be the critical parameter when combining in parallel, for which widths are added together; while width is the critical parameter when combining in series, for which lengths are added together. If the resistor value is provided in addition to the length and width, then the values are added for devices in series, and the values are combined in parallel for devices in parallel. If devices have only value given, then they may be combined in series and parallel with no constraints. The default behavior for resistors therefore implies the combination of the following:

property *valid_cellname* **series {W critical}**
property *valid_cellname* **series {L add}**
property *valid_cellname* **series {value add}**

property *valid_cellname* **parallel {L critical}**
property *valid_cellname* **parallel {W add}**
property *valid_cellname* **parallel {value par}**

Use of *combine_type* implies **enable** unless the only *combine_type* is "**none**".
Once series or parallel combining is enabled for a device, differences between the series/parallel networks of that device in the circuits being compared will be treated as property errors, not topological (wiring) errors.
Option **property parallel none** removes the default parallel merging behavior from every device. Note that parallel merging is enabled for all devices and subcircuits by default, so this is the only way to change the underlying behavior. Subsequent calls to the **property** command can be used to reinstate the parallel merging behavior on a per-device basis.
Option **property parallel connected** disables the default behavior of allowing devices with unconnected pins to be combined in parallel. This can be useful for debugging, as nets that are shorted or broken can change device pins from connected to unconnected or vice versa, leading to a difference in device count, and making the LVS output more difficult to debug. The reason for allowing devices with unconnected pins to be combined in parallel is that many such devices in the same cell (e.g., a standard cell layout) can produce a huge number of symmetries, causing an absurdly long symmetry-breaking stage at the end of LVS. The behavior can be reenabled with **property parallel open**. Option **property series none** removes the default series merging behavior from every device. Subsequent calls to the **property** command can be used to reinstate the series merging behavior on a per-device basis.
Note that for devices that connect in series, the terminals must be the first two pins of the device; any additional pins (e.g., substrate connection) must be connected to the same node on all instances of that device. Devices connect in parallel only if every corresponding pin of each device is connected to the same node.
Also note that netgen does not do full series/parallel topology reduction. The series/parallel network handling is specifically designed for LVS and attempts to match networks between two circuits with as little reduction as possible. However, only series and parallel reductions are allowed, as those transformations preserve network topology. Delta-Wye transformations are prohibited, as they do not preserve network topology.
Option **topology** sets a global option for the behavior of comparing device networks. If set to **strict**, any devices that form a network (such as resistors or capacitors) must match exactly in topology. If set to **relaxed**, two networks may have different topology as long as they are numerically the same after series and parallel combinations. The default behavior is **relaxed**.

property *valid_cellname*|*device_name* **merge {** *key merge_type* **}** ...

(Deprecated usage) With option **merge**, the behavior of devices combining in parallel or in series may be declared. The *key* is a valid property name for the device, while *merge_type* may be one of **add**, **add_critical**, **par**, **par_critical**, **ser**, or **ser_critical**. **add** and **add_critical** indicate that when two devices combine in parallel, the values of the devices for the indicated parameter will be summed. **par** and **par_critical** indicate that the values will be parallel-combined (1/((1/*A*)+(1/*B*))). **ser** and **ser_critical** indicate that when two devices combine in series, the values of the devices for the indicated parameter will be summed.
The "**_critical**" indicates that the comparison will actively attempt to merge devices by combining this parameter to make a better match. Without "**_critical**", the parameters are combined whenever the critical parameters are combined, but will not be used to actively resolve differences between the two circuits.
Note that this usage has been deprecated, as it does not properly distinguish between the topology (series or parallel) and the method of combination (addition or parallel combination). Use (**property** ... **parallel** ...) and (**property** ... **series** ...) instead (see above).

exhaustive [**on**|**off**]

report or set exhaustive or simple subdivision (default **on**).

restart
: start over (reset data structures)

matching [**element**|**node**] *name1*
: return the corresponding node or element name in the compared cell

Scripted Tcl procedures:

lvs *valid_cellname1 valid_cellname2* [*setup_file*] [*logfile*] [*options*]

Run the netlist comparison commands to load and compare the two netlists, leaving the bulk of the output in the file *logfile* (defaults to "comp.out" if unspecified), with a succinct summary printed to the terminal. If the optional *setup_file* is present, source this file prior to comparing the circuits. This file is Tcl-syntax compatible, and normally contains netgen commands that affect the way LVS is run. Typical commands used in the setup file are "**equate classes**..." and "**permute**...". If *setup_file* is not present, netgen will check for the default setup filename "**setup.tcl**", and source it if it exists. If no setup file exists, then default permutations (transistor source/drain and resistor endpoints) are used, and no assumptions are made about class equivalences. The LVS comparison is hierarchical (see command "**compare hierarchical**", above). If the files to be compared have a top-level structure (that is, components that are not inside a subcircuit definition), then the top-level cell takes the (full) name of the filename (see the description of "*valid_cellname*" syntax at the top of the command list). If the files contain all subcircuit definitions, or one wishes to compare cells below the top level of hierarchy, or the files have not been read prior to calling the **lvs** procedure, then the two-item list form **{***filename cellname***}** for *valid_cellname* must be used.

Valid *options* are one or more of:

-list
: Generate the output as a nested list result to the Tcl interpreter. This is only useful within the Tcl/Tk interpreter.

-json
: Provide output in JSON format as well as the usual text format.

-blackbox
: Treat all empty subcircuits as black-box cells.

-noflatten=*cellname_list*|*filename*
: Pass a list of names of subcells that indicate that netgen should not flatten the contents of the cells into the hierarchy above if they do not match. This list may be passed directly to the option as the value (in quotes or braces if more than one cell is specified), or the list may be in a file with one cell per line, and the filename passed as the argument to **-noflatten=**. Netgen's default policy is to flatten mismatched cells to resolve problems arising from differences between the hierarchy in the two netlists. But if the cells are really mismatched, then the flattening propagates to the circuit top level and makes the LVS output long and difficult to evaluate. This option should be used with care, as the netlists will report as matching if everything outside of the indicated cells in *cellname_list* matches.

Tcl Command-Line Interface (for Netgen 1.4 and below)

Note that the command-line interface for version 1.4 does not include the general form for specifying cells shown above for version 1.5. Be wary of cell names that can be confused between two different files.

Netlist manipulation commands:

readnet [*format*] *file*

read a netlist file (default format=auto). Auto-detection first looks for files with standard extensions. If no file is found with a standard extention, then the file is checked for the initial character, which is always "*****" in SPICE files and "**|**" in .sim files.

readlib *format*

Initialize a library module definition, where *format* is either **actel** or **xilinx**. The library definitions need to be initialized prior to reading any netlist file that is to be rewritten as "**writenet actel** *filename*" or "**writenet xilinx** *filename*".

writenet *format file*

write a netlist file

flatten *cell*

flatten a hierarchical cell. *Note*: Always flatten a cell in a hierarchical format (such as SPICE) before writing a flat format, even if the original netlist is already flat. Otherwise, the flattened namespace will not be cached, and the write process will incur a horrific performance hit.

flatten class *cell*

In this form of the **flatten** command, the whole cell database is searched for instances of the cell *cell*, which are then flattened by merging them into the parent cell.

flatten prohibit *cell*

In this form of the **flatten** command, the cell *cell* is prohibited from being flattened at any time during the comparison. It is equivalent in use to the "lvs" script option "-noflatten". Note that if subcells prohibited from flattening are mismatched, then the top level cell may still match; the output will then follow the top cell match with a list of subcells that failed matching. The keyword **deny** is accepted as an alternative to **prohibit**.

model *cell model*

declare a cell to be specific model type. Normally, model types are derived from low-level device calls in an input file (such as SPICE). This command will override the assigned model type, assuming that the cell has the correct number of pins for the model type. The supported low-level device model types are: **nmos** (3 or 4 port), **pmos** (3 or 4 port), **pnp**, **npn**, **resistor**, **capacitor**, **diode**, **inductor**, or **xline**. A low-level device can be treated as a subcircuit by declaring *model* to be **subcircuit**.

nodes [[**-list**] *element*] *cell*

print nodes of an element or cell
with option **-list**, return a list of nodes to the Tcl interpreter.

elements [[**-list**] *node*] *cell*

print elements of a node or cell
with option **-list**, return a list of elements to the Tcl interpreter. Element names are returned in the format "*element_name*/*pin_name*".

debug on|**off**|*command* [*options...*]

turn debugging on or off or debug a command. When debugging is on, the output of LVS comparisons reverts to the original, list-oriented format instead of the newer side-by-side format.

protochip

embed protochip structure

instances *cell*

list instances of the cell

contents *cell*

list contents of the cell

describe *cell*

describe the cell

cells [**list**|**all**]

print known cells
option **all**: print all cells, including primitives

ports *cell*

print ports of the cell

leaves [*cell*]

print leaves of the cell

quit

exit netgen and Tcl

reinitialize

reintialize netgen data structures

log [**file** *name*|**start**|**end**| **reset**|**suspend**|**resume**|**echo**]

enable or disable output log to file named *file*.

help

print basic usage information

Netlist comparison commands:

compare *cell1 cell2*

declare two cells for netlist comparison. Subcircuits that are not declared equivalent or that do not have the same name in both cells are flattened. Otherwise, subcircuit instances will be compared as black-box devices, without comparison of their contents (if any).

compare hierarchical [*cell1 cell2*]

do a full hierarchical comparison. The first call must declare the two top-level cells *cell1* and *cell2*. The cell hierarchy will be searched; all cells with matching names or cells that are declared equivalent will be saved in a stack. All cells whose match in the other top-level cell's hierarchy cannot be determined will be flattened. Upon the completion of the search, the top of the stack (the bottommost cells in the hierarchy) will be prepared for comparison. Subsequent calls to **compare hierarchical** should be called without any arguments. Each time **compare hierarchical** is called, the next pair of cells is taken off of the top of the stack and prepared for comparison. Once there are no cell pairs left to compare, **compare hierarchical** returns the string "`empty queue`".

iterate

do one netlist comparison iteration

summary [**elements**|**nodes**]

summarize internal data structures

print [**-list**] [**elements**|**nodes**|**queue**] [**legal**|**illegal**]

print full report on the internal data structures. This is a list of the partitions generated by the comparison algorithm.
with option **elements**, print only the element partitions.
with option **nodes**, print only the node partitions.
with option **queue**, print the comparison queue.
with option **legal**, print only legal (matching) partitions.
with option **illegal**, print only illegal (non-matching) partitions.
with option **-list**, return a nested list with the information, where each item in the list is one partition, and each item is a list of two parts, one for each circuit, and each part is a list of either element pins, or nodes. The **-list** option may only be used with either **print elements** or **print nodes**.

run [**converge**|**resolve**]

with option **converge**: run netlist comparison to completion (convergence)
with option **resolve**: run to completion and resolve automorphisms

verify [**elements**|**nodes**|**only**| **equivalent**|**unique**]

verify results

automorphisms

print automorphisms

equate [**elements**|**nodes**|**classes**] *name1 name2*

equate pins

with option **elements**: equate two elements
with option **nodes**: equate two nodes
with option **classes**: equate two device classes.
The name *name1* must belong to the first of the two circuits being compared and *name2* must belong to the other one.
with option **pins**: equate pins in the two cells being evaluated. This requires that two cells have been declared with the **compare hierarchical** command, and that LVS has

run to convergence on any pair of cells in the stack (see **compare hierarchical**), and the netlists verified as matching. The **equate pins** command will sort the ports of one cell to match those of the other.

ignore class *name*
: Remove all instances of the device class "*name*" from the database, and prevent any subsequent addition of any instance of the class when reading a netlist file. Use with caution! Can be useful in some cases, such as to ignore diode devices that cannot be recognized during layout extraction, or to ignore all parasitic devices in a netlist file. Available on Netgen versions 1.4.45 and higher.

global *cellname netname*
: This command makes the net named *netname* in the hierarchy of cell *cellname* into a global node. The cell *cellname* must exist in the database (it must have already been read into netgen). This command is typically used within the setup file to assert which nodes in a file need to be declared global. For instance, if the SPICE netlist "**file.spice**" contains a subcircuit using the default SPICE ground node name "**0**", then the setup file would need the line "**global file.spice 0**". Note that a SPICE netlist file can contain a *forward* reference to a global node name using the "**.global** *name*" card, in which case the **global** statement is not needed.

convert *cellname*
: This command searches cell *cellname* for global nodes, and converts each one into a local node while adding that node to the pin list of the cell. The database is searched for all instances of *cellname*, and the global node is passed from the parent cell to the instance by way of this pin.

permute [**transistors**|**resistors**|**capacitors**]
permute [**pins**] *model pin1 pin2*
permute forget *model pin1 pin2*
: with option **pins**: permute pins *pin1* and *pin2* on device *model*. If more than two pins may be permuted, this command may be issued multiple times between pairs of pins.
With option **forget**: remove any permutation between pins *pin1* and *pin2* on device *model*. This option is provided for situations where it is more convenient to first apply the default permutations and then specify the exceptions individually (netgen 1.4.67 and later only).
With option **transistors**: enable transistor source/drain permutations for any device recognized as a MOSFET.
With option **resistors**: enable permutations of resistor endpoints.
With option **capacitors**: enable permutations of capacitor top and bottom plates. Because capacitor top and bottom plates have different parasitics, capacitor plate permutation is not normally enabled.
With no options: enable transistor source/drain and resistor endpoint permutations.

exhaustive on|**off**
: set or clear exhaustive subdivision (default **on**).

restart
: start over (reset data structures)

matching [**element**|**node**] *name1*
: return the corresponding node or element name in the compared cell

Scripted Tcl procedures:

lvs *circuit1 circuit2* [*setup_file*] [*logfile*]
lvs **{** *file1 cell1***}** **{** *file2 cell2***}** [*setup_file*] [*logfile*]
: Run the netlist comparison commands to load and compare the two netlists, leaving the bulk of the output in the file *logfile* (defaults to "comp.out" if unspecified), with a succinct summary printed to the terminal. If the optional *setup_file* is present, source this file prior to comparing the circuits. This file is Tcl-syntax compatible, and normally contains netgen commands that affect the way LVS is run. Typical commands used in the setup file are "**equate classes**..." and "**permute**...". If *setup_file* is not present, netgen will check for the default setup filename "**setup.tcl**", and source it if it exists. If no setup file exists, then default permutations (transistor source/drain and resistor

endpoints) are used, and no assumptions are made about class equivalences. The LVS comparison is hierarchical (see command "**compare hierarchical**", above). If the files to be compared have a top-level structure (that is, components that are not inside a subcircuit definition), then the first syntax is used, and the top-level cell takes the (full) name of the filename. If the files contain all subcircuit definitions, or one wishes to compare cells below the top level of hierarchy, then the second form may be used, where one or both circuits may be split into a list of two items, the first being the filename, and the second being the name of the subcircuit to compare. The second form was introduced with netgen version 1.4.13.

Integrating Netgen With Tcl/Tk-Based Magic

Currently, there are no specific advantages to using netgen directly under magic. It is still necessary to run the "extract" and "ext2spice" commands to generate a netlist, then run netgen's "lvs" command to compare the layout-derived netlist to a schematic-derived netlist.

Magic versions 7.5 and 8.0, as of December 2010, have an option "ext2spice hierarchical on", that will produce hierarchical netlists for LVS using the netgen "lvs" command. Of course, it is still possible to compare flattened netlists, but hierarchical comparison is vital for any large project.

Magic versions 8.1 and 8.2, as of June 2017, have an option "ext2spice blackbox on", which can be used in conjunction with the "**-blackbox**" option to the "**lvs**" script command to handle any abstract view in a layout for which contents are unknown but assumed to be LVS clean.

See the "To-Do" section in the release notes for more on planned integration capabilities.

email: tim@opencircuitdesign.com

Last updated: March 6, 2023 at 10:03am