gEDA gnetlist Users Guide

Ales Hvezda

September 21st, 2003

Contents

1 Introduction

This document describes how to use gnetlist. This document and gnetlist in general are pretty ALPHA, so keep that in mind as you use it to generate netlists. As all engineers know, it is very important that you do not blindly trust tools, assuming that they will always create correct output. gnetlist is certainly no exception to this rule. It is very important that you verify *every* netlists you create. As with most programs (including all the programs in gEDA), gnetlist comes with NO WARRANTY. Blah, I hate having to say that, but I'm hoping that this warning will keep the user from assuming that gnetlist generates perfect netlists. Though if you find a bug, please let `ahvezda@geda.seul.org` know.

This document is very rough, so please e-mail all corrections to `ahvezda@geda.seul.org` or file a bug report on the gEDA homepage at `http://www.geda.seul.org`. Thanks!

2 Overview

`gnetlist` is the gEDA netlister. It takes as input schematic files and produces a netlist. A netlist is a textual representation of a schematic. This textual representation has all of the connections between devices completely resolved. This means that all the connections associated with a net are grouped together. The netlister also handles hierarchies of schematics.

`gnetlist` has a very flexible architecture. The main program, which is written in C, reads in a schematic (using routines from libgeda) and creates an internal representation of the schematic data. This internal representation is then manipulated by a backend which is responsible for writing the various netlist formats. The backend for each netlist format is written in scheme (specifically Guile). This architecture not only allows for an infinite number of netlist formats, but also allows the netlister to generate other reports (like bill of material lists).

As of 20001006 gnetlist has scheme backends to support the following netlist formats:

1. PCB & PCBboard - UNIX PCB netlist format.
2. Allegro netlist format
3. BAE netlist format
4. BOM & BOM2 - Bill of Material generators
5. DRC - Start of a design rule checker
6. gEDA - the native format of gEDA, mainly used for testing
7. Gossip netlist format

8. PADS netlist format
9. ProtelII netlist format
10. Spice compatible netlist format
11. Tango netlist format
12. Verilog code
13. VHDL code
14. VIPEC netlist format
15. VAMS - VHDL-AMS netlist format

This list is constantly growing. Several lacking features (as of 20001006) are: no support for buses, error detection and reporting is fairly limited, and ... (many more).

3 Installation

Hopefully by now you have already installed gnetlist on your machine. This document does not cover installation. You can verify the installation by running:

```
libgeda-config --version
gesym-config --version
which gnetlist
ldd `which gnetlist`
```

The first two should return the version of the installed tools (libgeda and the symbol library) and the next command should return the path to the gnetlist binary. The final command (only on Unix-like operating systems which include the `ldd` utility for listing dynamic dependencies of executable files or shared objects) will return which libraries are linked to gnetlist; all of the request libraries must be found for gnetlist to run. If these commands do not return the expected results, then most likely the gEDA tools are not installed properly. Please see the appropriate INSTALL docs (which came with the distribution) for more info on installing the gEDA tools.

4 Running gnetlist

It is very easy to run gnetlist. gnetlist is a pure command line interface so there is no pesky GUI to get in the way :-) For a list of command line arguments please run "`gnetlist -h`".

You need to specify the following two parameters to run gnetlists:

- `-g proc` (this specifies which backend to run against the schematics)
- `filename.sch` (this specifies the schematic files)

You can specify multiple schematics on the command line. The default filename for the generated netlist goes into "`output.net`" You can change this default location by using the `-o filename` option.

A few examples on running gnetlist:

```
gnetlist -g geda -o stack.net stack\_1.sch
```

(output netlist (in `stack.net`) for `stack_1.sch` using the gEDA native format)

There are also a few debugging flags. The first one is the `-v` flag which enables verbose mode. Verbose mode outputs a whole bunch of information on what gnetlist is doing as well a dump of the internal representation. The `-i` flag which puts gnetlist into a interactive mode is very useful in debugging scheme backends and typically is not used by the end user.

For a detailed list of command line arguments please see the gnetlist man page.

5 Schematic / symbol requirements

This section describes what schematics/symbols need to have to be usable with gnetlist. In order for gnetlist to work correctly, both the schematics and supporting symbols must be correct. Basically these requirements consist of attribute specification. Attributes are used through out the gEDA system to represent information. Attributes are the only way of adding information to components, nets, pins, etc... For more detailed information about the attributes mentioned in this document, please see the `attributes.txt` document (Master attribute list).

5.1 Symbol requirements

- All symbols must have a device= attribute.

- All pins must have the `pin#=#` attribute. This attribute will eventually change form, but for now it is required as `pin#=#`
- All pin should also have a `pinlabel=` attribute.
- For symbols which are slotted you also need the `slot=` attribute, for each slot a `slot#=#` attribute, and the `numslots=#` attribute. Slotting will also change in the near future, but for now it should be specified as above.
- For any power/gnd/arbitrary you need to put `net=` attributes inside the symbol. See the netattrib.txt document for more info.
- You can supply default values for various parameters (this is dependent on which backend you use) by taking advantage of the attribute "promotion" mechanism. See below for more info as well as the gschem documentation.
- For symbols which you want the netlister to completely ignore use the `graphical=1` attribute
- For more tips on symbols, please see the symbol creation document.

5.2 Schematic requirements

- Most importantly, every component you want to show up in a netlist must have a `refdes=` attribute. This is *VERY* important. gnetlist should warn you if you have a component which doesn't have a `refdes=`, but there have been bugs which do not cause this warning.
- You can label all nets using the `label=` attribute. You only need to attach this label to one net segment (of an electrically connected net) for all the net segments to inherit the label.
- You can have multiple schematics in a design (which is actually a confusing term since it means many different things to people). To use multiple schematics to create a single netlist, just specify them on the gnetlist command line.
- If you name nets the same, then these nets will be electrically connected. Same net names spawn all the specified schematics.
- There are quite a few issues that deal with hierarchy please see the hierarchy section below.

5.3 Random notes

- Attributes which are not attached to anything and are inside a symbol are "promoted" to the outside of the symbol when the symbol is placed inside a schematic (in gschem). These promoted attributes are always looked

at/for first before going into the symbol. So, in other words, if there is an attribute with the same name is inside a symbol and attached to the outside of the instantiated component, then the outside attribute takes precedence.

6 Hierarchy Support

TBA

7 Specific backend info

TBA

8 Scheme backend API

TBA