

SILVACO

AccuCell QuickStart

Silvaco

4701 Patrick Henry Drive, Bldg. 2
Santa Clara, CA 95054
Phone (408) 567-1000
Web <http://www.silvaco.com>

March 26, 2010

AccuCell QuickStart
Copyright 2010

Silvaco
4701 Patrick Henry Drive, Bldg #2
Santa Clara, CA 95054

Phone: (408) 567-1000
Web: <http://www.silvaco.com>

1.1: Running AccuCell

ACCUCELL uses TCL as the command-processing framework, so script files will have the extension `.tcl`. You can create a TCL script in a text editor of your choice. For example, if the script file is called `run.tcl`, from the prompt in c-shell you would enter the following command:

```
accucell run.tcl |& tee log.txt
```

This command invokes ACCUCELL using `run.tcl` as input, and saves ACCUCELL output in file `log.txt`. ACCUCELL will analyze the cells through the creation of a library. All the information describing the operation of the analysis will be sent to standard output and standard error streams.

1.1.1: Tutorial Step 1: Getting Started

Running AccuCell

Type the following to ensure the tool is ready to be executed:

```
%accucell
```

At this point, something similar to the following should be displayed.

Example

```
Loading ByteCode Loader.....
AccuCell(R) 2.3.84.C, Wed Mar 10 14:45:06 PST 2010, Proprietary and
Confidential Software.
Copyright (c) 1984 - 2010, Silvaco Inc.

AccuCell>
```

The tool should now be in the interactive prompt mode. If it is not, then see your System Administrator otherwise, type `exit` to end:

```
AccuCell> exit
```

Tutorial Data Files

The first step to run ACCUCELL's tutorial is to make sure that the example included with the ACCUCELL release is available. The data files should be in the `examples/accucell` subdirectory where Silvaco tools were installed. This subdirectory includes a sample cell library used for tutorial purposes. Please see your System Administrator if you have problems locating it.

Three example directories:

- `combinational`
- `sequential`
- `special`

contain the various groups of cell examples. At the time this quickstart is created, there are 11 cells in 3 groups included with installation. More cells might be added with newer versions.

`combinational` contains cell subdirectories:

- `inv`
- `mux2`
- `nand2`
- `nor2`

- `xnor2`
- `xor2`

and a `models` subdirectory containing the `model.inc` file with the SPICE models.

`special` contains cell directories:

- `pad`
- `tristate`

and the same symbolic links or files for `models`, `.cfg` and `.tcl`.

`sequential` contains cell directories:

- `dff`
- `dff2`
- `latch`

and the same symbolic links or files for `models`, `.cfg` and `.tcl`.

Make a copy of the entire ACCUCELL tutorial to a working directory, as follows:

```
cp -r <install_pathname>/examples/accucell/combinational
  <working_directory>/.
cp -r <install_pathname>/examples/accucell/sequential <working_directory>/.
cp -r <install_pathname>/examples/accucell/special <working_directory>/.
```

In the following steps we outline the basic operation of ACCUCELL using these sample library.

1.1.2: Tutorial Step 2: Setting Up the Library Hierarchy

This tutorial demonstrates characterization using a library made up of eleven cells. Of the eleven cells, six are combinational (`inv`, `mux2`, `nand2`, `nor2`, `xor2`, `xnor2`), two special cells (`pad`, `tristate`), and three sequential cells (`dff`, `dff1`, `latch`). It is important to note that each cell must have its own directory within the library level. The hierarchical directory structure may be verified by typing:

```
% ls cell_name
```

at least the following two files are required to exist in each cell subdir (e.g., for `xor2`):

- `xor2.cfg`: Configuration file for the cell
- `xor2.spi`: SPICE file for the cell

1.1.3: Tutorial Step 3a: Setting Up the Library-Level Configuration File

Become familiar with the library level configuration file by entering:

```
%vi tutorial.cfg
```

1.1.4: Tutorial Step 3b: Setting Up the TCL Command File

To execute ACCUCELL in batch mode, you must supply a TCL based command file (`run.tcl`). Most often, this file contains the single command:

```
gen_lib <cell_list_filename> <configuration_filename>
```

In our example, the list of cells in the library is contained in the file `cell.list`, and the command is contained in the `run.tcl` file.

Become familiar with this file by entering:

```
% vi run.tcl
```

Notice that in this case the command is:

```
gen_lib cell.list tutorial.cfg
```

1.1.5: Tutorial Step 3c: Setting Up the Library List File

The library list file contains the list of cells for ACCUCELL to characterize. Verify that the list contains all cells for the tutorial by entering:

```
% vi cell.list
```

1.1.6: Tutorial Step 4: Setting Up the Cell Level Configuration Files

As stated above, each cell must have its own directory within the library level. Further, each cell directory should also contain the SPICE netlist for the cell, the individual configuration file (<cell_name>.cfg), and any other cell specific files.

Also, become familiar with each cell configuration file by entering:

```
% cd <cell_name> ; vi <cell_name>.cfg
```

1.1.7: Tutorial Step 5: Running AccuCell to Characterize a Library

ACCUCELL is executed from the Unix command line. When running in batch mode, it is also good practice to direct the standard output and standard error streams into a runtime log file, which may be done in c-shell with the tee command as follows:

```
% accucell run.tcl |& tee log.txt
```

1.1.8: Tutorial Step 6: Checking Results

When the run is complete the message:

```
library generation completed
```

should appear and you should be returned to the Unix prompt. Review the log file (log.txt in this case) for errors.

Note: Some warnings are OK and are only informational. Others, depending on the situation, might indicate a need for corrective action.

Cell directories will contain individual cell results. The library level directory will contain the combined results of all cells for that run. <lib_name>_sps.lib is the liberty file with timing and optionally power and leakage results. Results are in NLDM and optionally CCS formats. Optional verilog simulation models may also exist depending on selected model options as <lib_name>_vlg.v.

[This page intentionally left blank.]