Cadence Liberate
Library Characterization Tool Tutorial

Digital VLSI Chip Design CAD manual addendum

The Digital VLSI Chip Design book describes in detail the process of characterizing a cell library. That is, take the extracted layout for a set of standard cells, and run analog simulations on those cells to characterize the electrical/timing behavior in way that can be used by HDL synthesis. This means understanding the propagation delays, the internal power, the input capacitance, and many other aspects of the behavior of the cells. The result is a "liberty format" .lib file that describes all this electrical and timing behavior.

This is described in Chapter 8 of the book. Section 8.1 describes the Liberty format used by HDL synthesis, and place & route tools for timing and power information. Section 8.2 describes the use of the Encounter Library Characterizer (ELC) tool. Unfortunately, that tool is no longer supported by Cadence. The new library characterizer tool from Cadence is called Liberate. This tutorial replaces section 8.2 for use with Liberate. Section 8.2.1 describes how to generate the input netlist for the characterization. This process is the same for the new tool. Up to the creation of the netlist using the ADE-L tool (page 233) you can follow this exact process from the book.

The differences start with the modifications to that netlist file to make it compatible with Liberate. The main difference is that Liberate seems to strongly prefer vss as the name of the ground node in the circuit. Also, Liberate allows the use the "global" parameter in the Spectre input file to define certain nodes (typically the power and ground nodes) as global in scope. With the vss and vdd nodes described as global, the individual "subckt" descriptions no longer need to include the vdd and vss arguments. An example of a Liberate-modified netlist for two cells (INVX1 and NAND2X1) replacing Figure 8.17 is:

```
simulator lang=spectre
global vss vdd

subckt INVX1 A Y
M1 (Y A vdd vdd) ami06P w=6u l=600n as=9e-12 ad=9e-12 ps=15.0u \
    pd=15.0u m=1 region=sat
M0 (Y A vss vss) ami06N w=3u l=600n as=4.5e-12 ad=4.5e-12 ps=9u \
    pd=9u m=1 region=sat
ends INVX1

subckt NAND2X1 A B Y
M2 (Y A vdd vdd) ami06P w=6u l=600n as=9e-12 ad=9e-12 ps=15.0u \
    pd=15.0u m=1 region=sat
M3 (Y B vdd vdd) ami06P w=6u l=600n as=9e-12 ad=9e-12 ps=15.0u \
    pd=15.0u m=1 region=sat
M0 (Y B net12 vss) ami06N w=6u l=600n as=9e-12 ad=9e-12 \
    ps=15.0u pd=15.0u m=1 region=sat
M1 (net12 A vss vss) ami06N w=6u l=600n as=9e-12 ad=9e-12 \
    ps=15.0u pd=15.0u m=1 region=sat
ends NAND2X1
```

There is a script in the F17 bin directory called scs2liberate that should automatically make the modifications to the netlist generated by ADE-L to make it compatible with Liberate. The use f this script is:

```
scs2liberate <input-file> <output-file>
```

You should check the output of this tool carefully to make sure it looks right. It has been tested, but not on a huge number of input files! To make things easier later in the process, you should name the output file **libcells.scs**.

Now everything is different from the book! The new tool does the same thing as ELC, but it is a completely different tool with a different interface.

In your VLSI directory (or wherever you want run Liberate from), make a recursive copy of the **Liberate** directory from **/uusoc/facility/cad_common/local/class/6710/F17/cadence**. This will copy a number of directories and setup files. The following Linux commands do the trick (make sure you include the "dot" at the end of the cp command).

```
cd ~/VLSI
cp -r /uusoc/facility/cad_common/local/class/6710/F17/cadence/Liberate .
```

In that new **Liberate** directory you will see the following:

1. A **lib** directory – that's where the generated **<library>.lib** file will be generated.
2. A **netlist** directory – put your **libcells.scs** file in this directory.
3. A **templates** directory – edit the **UofU_Cell_Defs.tcl** file in this directory to include cell descriptions for each of the cells you want to characterize. The **UofU_Templates.tcl** file has definitions for the timing and power templates that will be used for characterization. You probably don't need to modify these unless you are using a different technology than ON Semi C5N.
4. A **UofU_Cells.tcl** file – edit this file to make a list of the cells that you want to characterize in this run. This could be a list of every cell described in **templates/UofU_Cell_Defs.tcl**, or it could be a subset if you just want to try a few.
5. A **userdata** directory – edit the **userdata.lib** file to reflect the areas and footprints of each of the cells in your library.
6. A **tcl** directory – Edit the **UofU_Char.tcl** file in this directory to change the name of the library that the tool generates. If you don't modify this, the tool will generate a file named **Lib6710_XX.lib** by default. The **settings.tcl** file in this directory has configuration commands for the Spectre simulator. You won't need to modify this file at all.
7. A **models** directory that has the Spectre model files for the ami06N and ami06P transistors used by in your netlist.
8. A **run.sh** shell script that calls the cad-lib Liberate script with the appropriate input files, and makes a copy of the log information in a **Liberate.log** file.

Let's take a closer look at each of these steps.

Characterization Netlist: Start with your **libcells.scs** file that describes the transistor-level netlist for the cells you'd like to characterize (the equivalent of Figure 8.17 in the book). Put this file into the **Liberate/netlist** directory.

Cell Descriptions: Edit **templates/UofU_Cell_Defs.tcl** to add definitions for each cell that you will be characterizing. You can continue to add cells to this file as you add cells to your library. This is the central place where the cell interfaces are described. The actual cells being characterized on any particular run of the tool are only those listed in the **UofU_Cells.tcl** file. As an example, here are the descriptions for INVX1 and NAND2X1. This is essentially the same as the interfaces extracted by ELC and shown in Figure 8.19 in the book, but you'll have to type them in by hand.

```
if {[ALAPI_active_cell "INVX1"]} {
     define_cell \
               -input { A } \
              -output { Y } \
              -pinlist { A Y } \
             -delay delay_template_5x5_X1 \
             -power power_template_5x5_X1 \
            INVX1
}

if {[ALAPI_active_cell "NAND2X1"]} {
     define_cell \
               -input { A B } \
               -output { Y } \
               -pinlist { A B Y } \
              -delay delay_template_5x5_X1 \
              -power power_template_5x5_X1 \
              NAND2X1
}
```

The "if" statement says to ignore the cell description if it's not currently active in the tool. This is for efficiency of execution. After the "if" each cell is described by a **define_cell** command where the inputs, the outputs, the argument list (in the order used in the netlist), the delay and power templates to use, and the name of the cell are described. The delay and power templates are defined in the **UofU_Templates.tcl** file. You can look in that file to see what they are and how they're defined. Basically there are versions of the delay_template and power_template that are used for X1, X2, X4, and X8 (and larger) cells. The difference is that the cells with larger drive strengths are characterized over a larger range of capacitive loads.

Sequential cells are further characterized for constraints such as setup and hold times. The template for constraints is called **constraint_template_5x5**. An example of a flip flop definition is:

```
if {[ALAPI_active_cell "DCBX1"]} {
      define_cell \
              -clock { CLK } \
              -async { CLR } \
              -input { D } \
              -output { Q QB } \
              -pinlist { CLK CLR D Q QB } \
             -delay delay_template_5x5_X1 \
             -power power_template_5x5_X1 \
             -constraint constraint_template_5x5 \
             DCBX1
}
```

You need to add a description for each new cell that you add to your library.

Now you need to add the cells that you want to be characterized on this run of the tool to the **`UofU_Cells.tcl`** file. This is just a list of cell names. The list could be all the cells described in the **`templates/UofU_Cell_Defs.tcl`** file, or just a subset of them (if you want to try out a new cell without characterizing the whole library again, for example). The format of this file is:

```
#
# List the cells that you want to include on
# this characterization run.
# The cell definitions should be in the
# templates/UofU_Cell_Defs.tcl file.
set cells {
       INVX1
       NAND2X1
}
```

User data: This file (**`userdata/userdata.lib`**) has information about the area and footprint of your cells. This is similar information to the **`footprints.def`** file described in the book on page 242, but in a different format. The area is straightforward – it's the area of the cell. I measure the area from the center of the lower left contact in the GND line to the center of the upper right contact in the VDD line. The footprint is used to group cells into similar functionality.

As described in the book, cells with the same footprint can actually have different areas. For example, all your INVXn cells should have the same footprint (INV), but they'll each have different areas. The footprint information lets the place & route tool choose suitable replacements as required for achieving speed and power goals. An example of the userdata.lib file is:

```
library (LIB6710_XX) {
     cell (INVX1) {
            area : 129.6;
           cell_footprint : "INV";
      }
      cell (NAND2X1) {
             area : 194.4;
            cell_footprint : "NAND2";
      }
}
```

Characterization commands: The commands that drive the tool are in the two files in the tcl directory: **`settings.tcl`** with deep dark secret settings for the tool (I got this file from Cadence and haven't dissected exactly what every line means), and **`UofU_char.tcl`** which drives the actual characterization run with your files. The only thing you probably have to change in this file is the name of the library you want to end up with. Even this isn't essential – you can always edit the default **`Lib6710_XX.lib`** file to be a different name. But, it only takes a second to modify the LIBNAME field in this file. You can also see the sequence of commands that will drive the actual characterization.

The only other thing you might want to change in this file is to select a different set of model files for the transistors. By default you'll get the "typical" parameters for the ON Semi C5N process (called ami06N and ami06P because ON Semi used to be named AMI Semiconductor). This is almost always what you want.

But, you could also try simulations with best-case and worst-case conditions. You could change the PROCESS variable to "ff" for "fast-fast" to get the best-case performance, and "ss" for "slow-slow" to get the worst-case behavior as described by ON Semi. You could also go to mosis.com and get the extracted models for a recent run on the process and use that model if you like. That might be a more up to date version of the model parameters. An example of a mosis.com extracted model is also in the **`models/spectre`** directory (**`T89Y.scs`**).

Running Characterization: Now that you've modified all the relevant files you can actually run the Liberate tool. You could run it directly from the cad-lib script, but there's a **`run.sh`** script in your new **`Liberate`** directory that runs the tool, with the **`UofU_Char.tcl`** as the input file, and forking the console output to a **`Liberate.log`** file so you can look at the log and make sure you didn't have any serious errors or warnings in the process. You can run this shell script, but you'll probably have to say explicitly where it is using the dot notation (which says to look for it in your current directory):

```
./run.sh
```

If everything was specified correctly, this will result in a **`Lib6710_XX.lib`** (or whatever named you changed it to) in the lib directory of your Liberate run directory. This .lib file can be used directly by the place & route tools, but needs to be converted to binary format for the Synopsys HDL compiler Design Compiler.

As with the older ELC tool, this characterization will spawn a bunch of Spectre simulations for various conditions, and automatically check the timing and power of the cell. These data are collected into the proper Liberty format and eventually output into the .lib file. Because a large number of Spectre simulations are used for the characterization, this can take a while for a large library with a lot of complex cells.

Converting the .lib file: To convert the .lib into a binary .db file for Design Compiler, you'll use a tool called Library Compiler from Synopsys. Call this with the **`syn-lc`** script. The process is to read the .lib file, then output that library as a binary version in a .db file. This is as described in the CAD book in Section 8.4, but use the **`syn-lc`** library compiler script directly rather than going through the design compiler interface. This is the recommended flow in the latest Synopsys tools. Here's a transcript of the simple 2-cell version:

```
[elb@lab2-20 lib]$ syn-lc
Using setup-synopsys from S17
Assuming your OS is amd64
You are now set up to run the synopsys tools.

Working directory is /home/elb/VLSI/Liberate/lib

                              Library Compiler (TM)
                                  DesignWare (R)
                Version M-2016.12-SP3 for linux64 - Apr 13, 2017
                      Copyright (c) 1988 - 2017 Synopsys, Inc.
    This software and the associated documentation are proprietary to Synopsys,  Inc.
This software may only be used in accordance with the terms and conditions
  of a written license agreement with Synopsys, Inc. All other use, reproduction,
             or distribution of this software is strictly prohibited.

Initializing...
lc_shell> read_lib Lib6710_XX.lib
Reading '/home/elb/VLSI/Liberate/lib/Lib6710_XX.lib' ...
Warning: Line 1, The 'default_inout_pin_cap' attribute is not specified. Using 1.00. (LBDB-172)
Warning: Line 1, The 'default_input_pin_cap' attribute is not specified. Using 1.00. (LBDB-172)
Warning: Line 1, The 'default_leakage_power_density' attribute is not specified. Using 0.00. (LBDB-172)
Warning: Line 105, Cell 'INVX1', The cell_leakage_power attribute of the 'INVX1' cell is redundant
        and not used in the leakage_power modeling. (LBDB-644)
Warning: Line 215, Cell 'INVX1', pin 'A', The pin 'A' does not have a internal_power group. (LBDB-607)
Warning: Line 229, Cell 'NAND2X1', The cell_leakage_power attribute of the 'NAND2X1' cell is redundant
        and not used in the leakage_power modeling. (LBDB-644) Technology library
'Lib6710_XX' read successfully
1
lc_shell> write_lib Lib6710_XX -o Lib6710_XX.db Wrote the
'Lib6710_XX' library to '/home/elb/VLSI/Liberate/lib/Lib6710_XX.db' successfully
1
lc_shell> exit Memory usage for this session 19 Mbytes.
CPU usage for this session 0 seconds ( 0.00 hours ).

Thank you...
[elb@lab2-20 lib]$
```

You'll see that there are a number of warnings – typically related to the cell leakage power calculation. You can safely ignore these warnings because the leakage power in our process is negligible anyway. The Liberate tool also seems to not generate every possible attribute that the library compiler understands. You can safely ignore this too.

You can now use your converted library with Synopsys HDL synthesis as described in the CAD book as described in Chapter 9.