Πολυπαραμετρική Μοντελοποίηση Βιβλιοθηκών Standard Cells σε Περιβάλλον Synopsys

Συγγραφέας Διπλωματικής Εργασίας
Βλιώρα Λυδία

Επιβλέποντες καθηγητές
Μούντανος Ιωάννης
Ευμορφόπουλος Νέστορας

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών
Τηλεπικοινωνιών και Δικτύων, Πανεπιστήμιο Θεσσαλίας

Σεπτέμβριος 2011

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Ιωάννη Μούντανο αλλά και τον κ. Νέστορα Ευμορφόπουλο διότι η εκπόνηση αυτής της εργασίας δε θα ήταν δυνατη χωρίς τη βοήθειά τους.

Επίσης , θέλω να ευχαριστήσω θερμά τον κ. Γιώργο Σταμούλη για όλη τη βοήθεια, τις συμβουλές και την αμέριστη συμπαράσταση που μου προσέφερε σέ όλη τη διάρκεια των σπουδών μου. Επιπλέον θέλω να δώσω ένα μεγάλο ευχαριστώ στο Μιχάλη Τσιαμπά για τη συμβολή του σε αυτή την εργασία.

Σε αυτό το σημείο θα ήθελα να δώσω τις θερμότερες ευχαριστίες μου στη συμφοιτήτρια και φίλη μου Ευτυχία Μπέλλου με την οποία συνεργαστήκα στην παρούσα εργασία.Ήταν πάντα προθυμη να με συμβουλέψει για το οτιδήποτε και με προέτρεπε συνεχώς να μην αγχώνομαι.

Τέλος, θέλω να ευχαριστήσω εξαιρετικά την οικογένεια μου που στέκεται πάντα δίπλα μου σε όλες τις δύσκολες μου στιγμές.

Table of Contents

ΠΕΡΙΛΗΨΗ

Η σχεδίαση των ολοκληρωμένων κυκλωμάτων στις μέρες μας απαιτεί έναν ισορροπημένο χειρισμό πολύπλοκων ζητημάτων. Ο χρονισμός των κυκλωμάτων παραμένει κρίσιμος, αλλά η ισχύς, ο θόρυβος, η αξιοπιστία και η κατασκευασιμότητα είναι επίσης σημαντικά για την επιτυχία της σχεδίασης.

Όσον αφορά το χρονισμό των κυκλωμάτων, τα φυσικά φαινόμενα και οι τρόποι σχεδίασης παρουσιάζουν νέες προκλήσεις. Η διασύνδεση ανωτέρου επιπέδου παρουσιάζει μεγαλύτερη αντίσταση με μικρότερα πλάτη μετάλλου, με αποτέλεσμα σε κάποιες περιπτώσεις η εμπέδηση διασύνδεσης να είναι πολύ μεγαλύτερη από την αντίσταση οδήγησης του driving cell. Έτσι, απαιτείται η ανάλυση, σε ένα ευρύ φάσμα τιμών, της τάσης V_{dd} για την υποστήριξη των επιδράσεων της δυναμικής πτώσης τάσης (IR drop) καθώς και τρόποι σχεδίασης κυκλωμάτων χαμηλής κατανάλωσης ισχύος που συμπεριλαμβάνουν voltage islands και δυναμική κλιμάκωση τάσης/συχνότητας.

Απαιτείται ένα μοντέλο καθυστέρησης που να είναι ακριβές σε σχέση με την προσομοίωση του κυκλώματος, αλλά να εκτελεί γρήγορους υπολογισμούς για την υποστήριξη επίπεδης ανάλυσης μεγαλύτερων κυκλωμάτων. Το μοντέλο θα πρέπει να υποστηρίζει τον υπολογισμό καθυστέρησης κελιού, καθυστέρησης διασύνδεσης (interconnect delay), χρόνου μετάβασης (pin slew) και χωρητικότητας ακροδέκτη εισόδου για όλα τα στάδια, συμπεριλαμβανομένων των παρασιτικών.

Παράλληλα, όσον αφορά το μοντέλο ισχύος, η βιβλιοθήκη NLPM (Non-Linear Power Models) έχει εξυπηρετήσει το σκοπό της ικανοποιητικά, αλλά είναι ξεκάθαρο ότι το απλό lookup-based μοντέλο πάσχει από σημαντικές ελλείψεις που το εμποδίζουν από το να χρησιμοποιηθεί επιτυχώς σε ένα ευρύ φάσμα εφαρμογών και με προχωρημένες τεχνικές σχεδίασης.

Για όλα τα παραπάνω, εισήχθη το μοντέλο βιβλιοθήκης CCS (Composite Current Source), το οποίο επεκτείνει τα σημερινά μοντέλα βιβλιοθηκών έτσι ώστε να περιλαμβάνουν δεδομένα κυματομορφών ρεύματος. Με αυτό τον τρόπο επιτρέπεται μια πιο ακριβή ανάλυση και ενοποίηση των δεδομένων της βιβλιοθήκης.

Στην παρούσα εργασία, χρησιμοποιήθηκε η βιβλιοθήκη CCS της εταιρείας Nangate για τον σχεδιασμό και τη βελτιστοποίηση του χαρακτηρισμού standard cells για τεχνολογίες κάτω των 45nm σε περιβάλλον Synopsys. Πιο συγκεκριμένα, χρησιμοποιήθηκε το εργαλείο Liberty™ NCX το οποίο παράγει μια βιβλιοθήκη σε Liberty™ μορφή (.lib), από ένα σύνολο μοντέλων SPICE, λειτουργικών περιγραφών κελιών και των αντίστοιχων netlists. Η παραγόμενη βιβλιοθήκη περιέχει δεδομένα χρονισμού και ισχύος, βελτιστοποιήθηκε ως προς το χρόνο μετάβασης ακροδεκτών και μπορεί μετέπειτα να χρησιμοποιηθεί για ανάλυση χρονισμού και ισχύος με διάφορα εργαλεία. Σε αυτό το σημείο πρέπει να αναφερθεί πως η βιβλιοθήκη μας δεν περιέχει δεδομένα θορύβου καθώς το CCS Noise model δεν υποστηρίζεται από τη συγκεκριμένη έκδοση του Liberty™ NCX.

Η παρούσα εργασία οργανώνεται ως εξής: Στο 2^{ο} κεφάλαιο παρουσιάζονται θέματα σχετικά με τη σχεδίαση ολοκληρωμένων κυκλωμάτων μικρότερης κλίμακας καθώς και διάφοροι τύποι μοντέλων βιβλιοθηκών. Στο 3^{ο} κεφάλαιο αναλύεται η μοντελοποίηση CCS Timing και στο 4^{ο} κεφάλαιο η μοντελοποίηση CCS Power. Στο 5^{ο} κεφάλαιο περιγράφεται η λειτουργία του εργαλείου Liberty™ NCX της Synopsys και στο τελευταίο κεφάλαιο η διαδικασία που ακολουθήθηκε για το χαρακτηρισμό των βιβλιοθηκών Standard Cells της Nangate.

Chapter 1: Introduction

Accurate cell library models are essential for IC design implementation and sign-off tools. To ensure high accuracy at technology nodes of 90-nm and below, library models must accurately capture the complex transistor behavior of cells. Current-based modeling was introduced and proven in industry in the last few years as an effective way to model nanometer timing effects. However, accurate timing models alone are no longer sufficient because the effects are inter-dependent between timing, noise and power. To comprehensively deal with the inter-dependent timing, noise, and power effects in nanometer ICs, designers need a single current-based library model that enables the concurrent analysis and optimization of issues in all three categories.

The Composite Current Source (CCS) modeling technology is the first in the industry to deliver a complete open-source current based modeling solution for timing, noise and power. Along with the available parsers, characterization/validation tools, and guidelines, this open-source Liberty™ modeling format enables efficient characterization for cell library creators. For IC designers, the CCS modeling technology enables comprehensive nanometer design analysis and optimization for the first time. Designers can reduce design margins and speed design closure by eliminating iterations.

Chapter 2:
Nanometer issues

The widely available Non-Linear Delay and Power Models (NLDM/NLPM) have served the IC design industry for over 5 years. These models consist of tables capturing a cell's delay or power for each combination of input slew and output load. At process geometries of 90-nm and below, many new effects can no longer be modeled using this approach. They include:

- High impedance interconnect
- Miller effect
- Dynamic IR-drop
- Multi-voltage, and Dynamic Voltage and Frequency Scaling (DVFS) design
- Driver weakening
- Temperature inversion
- Increasing variations

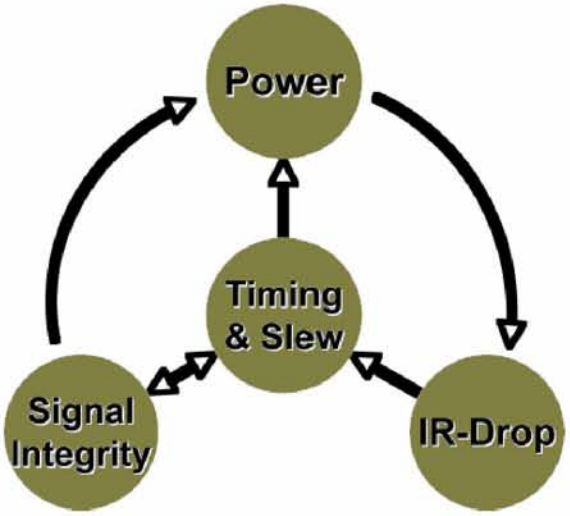


Figure 1

To make matters worse, some of these effects are inter-dependent between timing, noise and power. Figure 1 illustrates some of these inter-dependencies. For example, timing and slew rates affect power, which impacts IR-drop, which in turn changes timing. Also, timing impacts signal-integrity, which in turn can because crosstalk induced delay shifts. Another often overlooked fact is that signal integrity can impact power, which in turn impacts IR-drop and timing. Typically designers overlook glitches in a design that do not propagate to a register to cause a functional failure. However, a large number of glitches in a design increase power consumption.

Alternative models

The issues discussed so far, along with many others, prompted the development of the CCS modeling technology. Note that other models have addressed some of these issues using current-based methods, but always relating to timing or noise, or power separately–never all three. These ad-hoc models are typically closed, proprietary and do not provide the comprehensive approach that is vital for improving nanometer design modeling capabilities.

Moreover, some current-based timing models store voltage values for the driver. This provides only limited information about the actual current waveform. In between two (voltage, time) points, only the average current value is known. The true shape of the current waveform cannot be preserved unless a large number of voltage samples are used. Accurate current-based models should rely directly upon current values and include a method for reducing the number of (current, time) points while still retaining accuracy for the every time step in the output pin response.

The CCS solution for today and tomorrow

The Composite Current Source modeling technology addresses today's existing and emerging design requirements—the physical effects of nanometer designs as well as the needs of design strategies such as multiple-voltage domains. In one model, this open-source Liberty™ format combines the cell data needed to support timing, noise, and power analyses that are efficient yet accurate because they begin with current values that are characterized for the relevant nanometer dependencies. Because the open CCS format is extensible, these models constitute a foundation that can be enhanced as needed to meet future requirements such as variation-aware or statistical timing analysis.

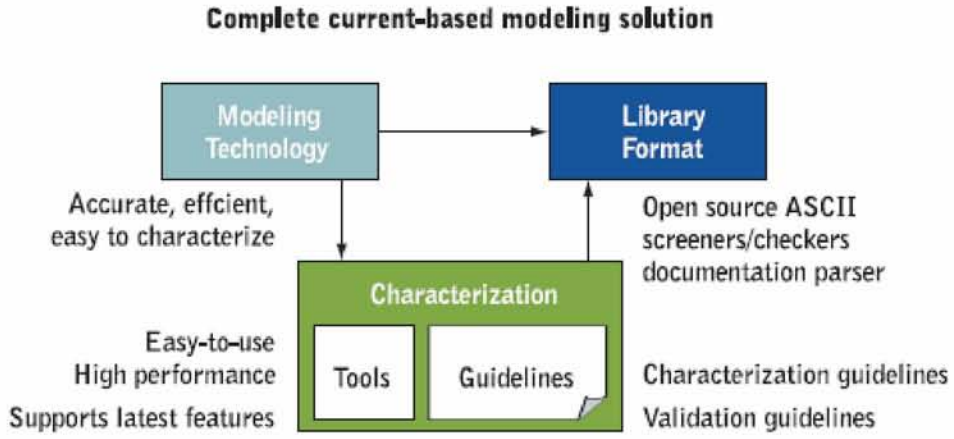


Figure 2

Chapter 3:
CCS Timing solution

CCS Timing consists of a driver model and a receiver model. The driver model describes how a timing arc propagates a transition from input to output, and how it can drive arbitrary RC networks. The receiver model describes the capacitance that an input pin presents to driving cells.

CCS Timing delay calculation uses advanced interpolation technology to determine a current waveform when the input slew and/or output load values do not match those used during cell characterization. Additionally, interpolation is used for intermediate values of VDD and temperature by using data from multiple libraries.

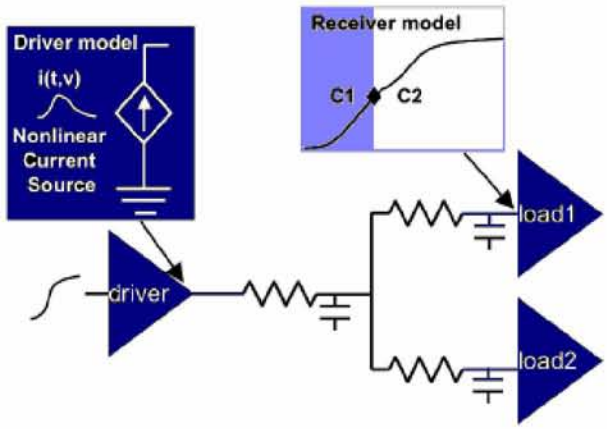


Figure 3

Characterization for CCS Timing

Characterizing a cell timing arc for CCS Timing is very similar to characterization for nonlinear delay models (NLDM): an input stimulus is chosen to produce a specific input slew time (S_{inp}); a load capacitance (C_{out}) is connected to the output pin; and a circuit simulation is run in the same way as for NLDM. But instead of measuring voltage thresholds at the output pin, current is measured through the load capacitor and into the input pin. The current through C_{out} is used for the driver model, and the current into the input pin is used to determine the receiver model.

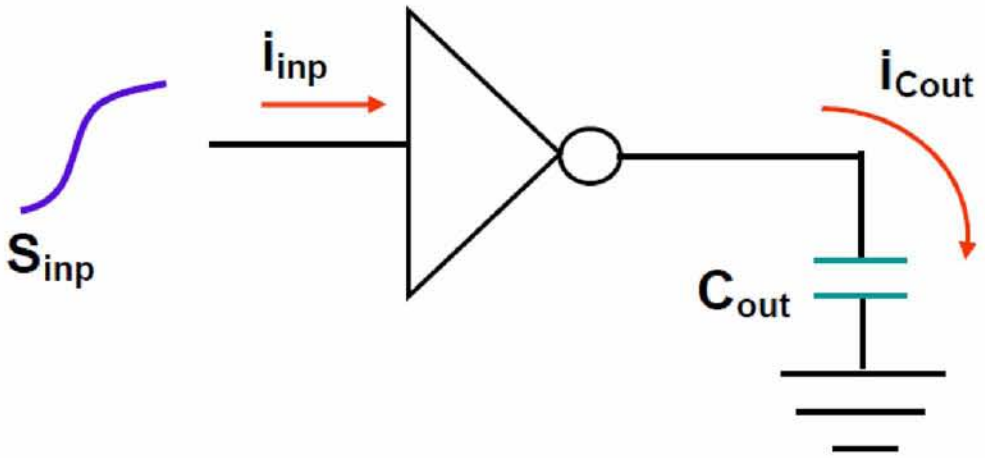


Figure 4 : CCS Timing characterization measurements

These characterization experiments are repeated for a table of different S_{inp} and C_{out} combinations. The current through C_{out} is saved for every circuit simulation timestep and then reduced to a much smaller set of current and time (i, t) points. The points are chosen such that $V_{out(t)}$ can be accurately reproduced for every timestep during the transition. Figure 5 shows an example of the complete i(t) waveform and a reduced set of points.

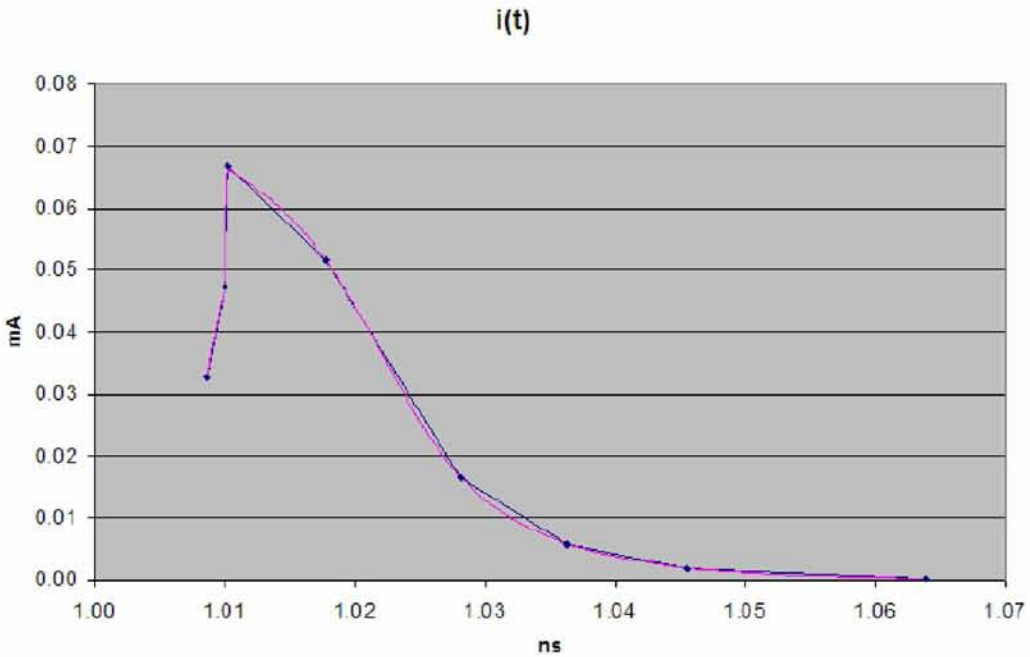


Figure 5 : Current waveform from circuit simulation, and reduced current points

The current and voltage at the input pin are saved and then used to determine C1 and C2 values such that gate-level delay calculation can closely match times to the delay threshold and to the second slew threshold at the input pin.

An additional piece of information, input reference time, is needed in order to calculate cell delays. The reference time is the simulation time at which the waveform at the input pin crosses the rising or falling delay threshold (often this is 50% of VDD).

Benefits of CCS Timing

- The CCS Timing delay calculation provides a high accuracy response for cell delay, interconnect delay, and pin slew.

- The CCS Timing receiver model produces excellent results on single-stage cells with large Miller effect. Furthermore, the stage delay and slew results are typically within 2% of the golden circuit simulation values.

- The CCS Timing enables scaling for intermediate VDD and temperature values. Synopsys delay calculation with CCS Timing includes powerful nonlinear V_{dd} scaling for timing check arcs. This results in better correlation to circuit simulation than with simple linear interpolation approaches.

- The current waveforms are expected to consume larger space in terms of data size compared to the NLDM models. Therefore, a "Compact CCS" is used to represent the current waveforms in a very compact form. The compact CCS takes advantage of similarity of I/V curves in the library. The compact CCS modeling uses a common set of I/V curves (known as base-curves) for the entire library and each instantiation of the current waveform is derived from one of these base curves. This technique allows for high accuracy while reducing the library size by up to 3 to 4x compared to the expanded (non-compact) CCS timing library.

- CCS timing also allows, for accurate representation of current characteristics of the library subjected to the process, variation. The variation-aware extension of CCS timing captures the current waveforms as the cell is subjected to process variation with respect to the process parameters.

Chapter 4:
CCS Power Library model

The CCS Power Library model represents the physical circuit properties more closely to the simulated data obtained during characterization with Spice. It is a current-based power model that contains the following features:

- One library format suitable for a wide range of applications (power analysis and optimization; reliability analysis)
- Allows power analysis with much higher time resolution compared to NLPM
- Equivalent parasitics necessary to perform IR-drop analysis
- Standard-cell and macro-cell modeling

CCS Power Characterization

The CCS Power characterization process is very similar to NLPM characterization:

1. First, the leakage currents are measured with simple DC analysis.
2. Next, the dynamic current waveforms are acquired with a transient analysis.
3. Finally, the equivalent parasitics is measured with fast AC and DC analysis runs.

Most of the characterization for timing and power can be performed simultaneously to reduce the characterization runtime.

Leakage Current

The typical simulation setup for leakage characterization is shown in Figure 6. This measurement is usually performed as part of the simulation setup for timing analysis.

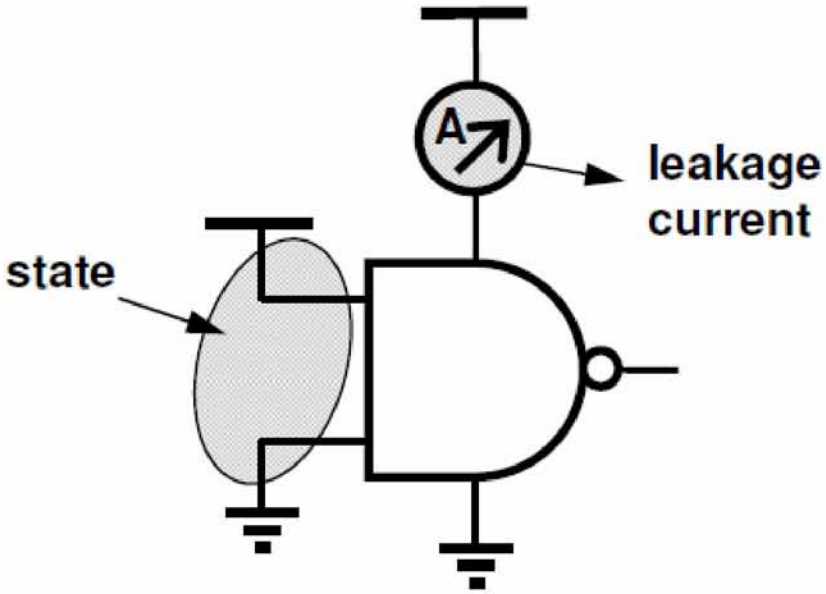


Figure 6 : Leakage Current Measurement

Dynamic Current

The dynamic current waveforms are acquired by performing a transient analysis as shown in Figure 7. This setup is identical to that used for timing characterization.

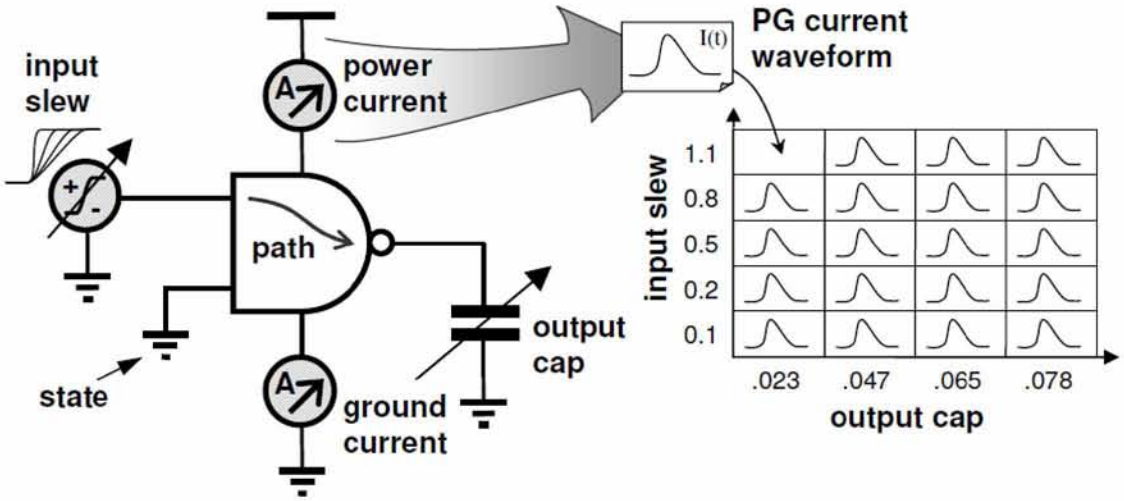


Figure 7 : Dynamic Current Waveform Measurements

Equivalent Parasitics

You can obtain the equivalent capacitance by performing a simple AC analysis in the DC operating points obtained during SD leakage analysis. By applying an AC voltage source to the PG pin and measuring the resulting AC current, you can derive the capacitance (shown in Figure 8 a).

An easy way to measure the equivalent resistance is to perform a second DC analysis with a slightly different voltage on the PG pin and with the output tied to a voltage source with the original DC output voltage, as shown in figure 8.b. Calculate the equivalent resistance from the current supplied by the voltage source at the output.

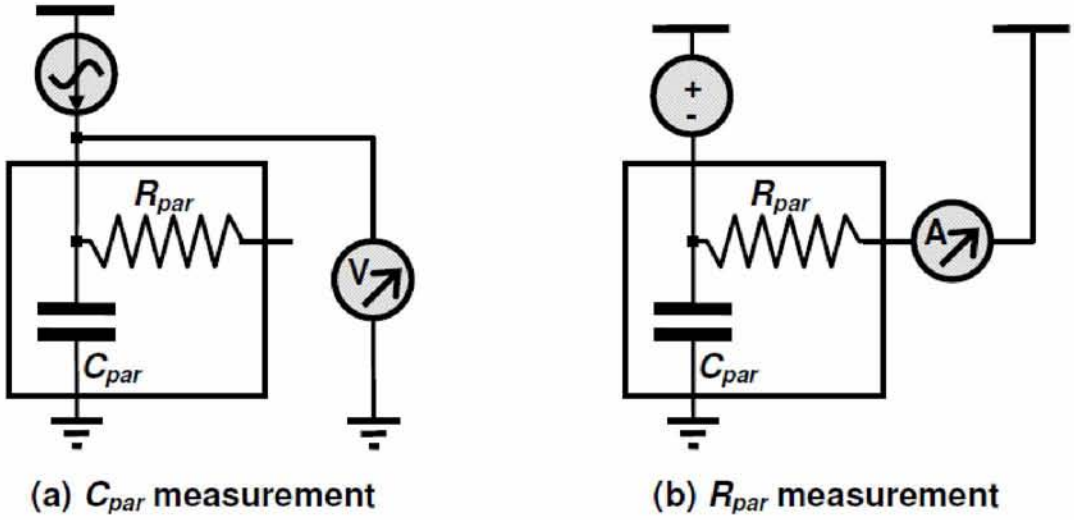


Figure 8 : Characterization of equivalent parasitics

Chapter 5:
Liberty™ NCX Operation Description

Liberty NCX is a software tool that generates a library in Liberty™ (.lib) format from a set of SPICE models, cell functional descriptions, and associated netlists. The generated library can then be used for timing, power, and noise analysis with compatible tools such as Library Compiler, IC Compiler, Design Compiler, and PrimeTime. The tool can generate CCS, NLDM, and NLPM models for the library.

Figure 9 shows the data files used for characterization. The procedure starts with either an existing seed library or a set of template files containing high-level descriptions of each cell, a set of SPICE netlist files for the cells to be characterized, and a set of SPICE model files for the circuit elements used in the cell netlists.

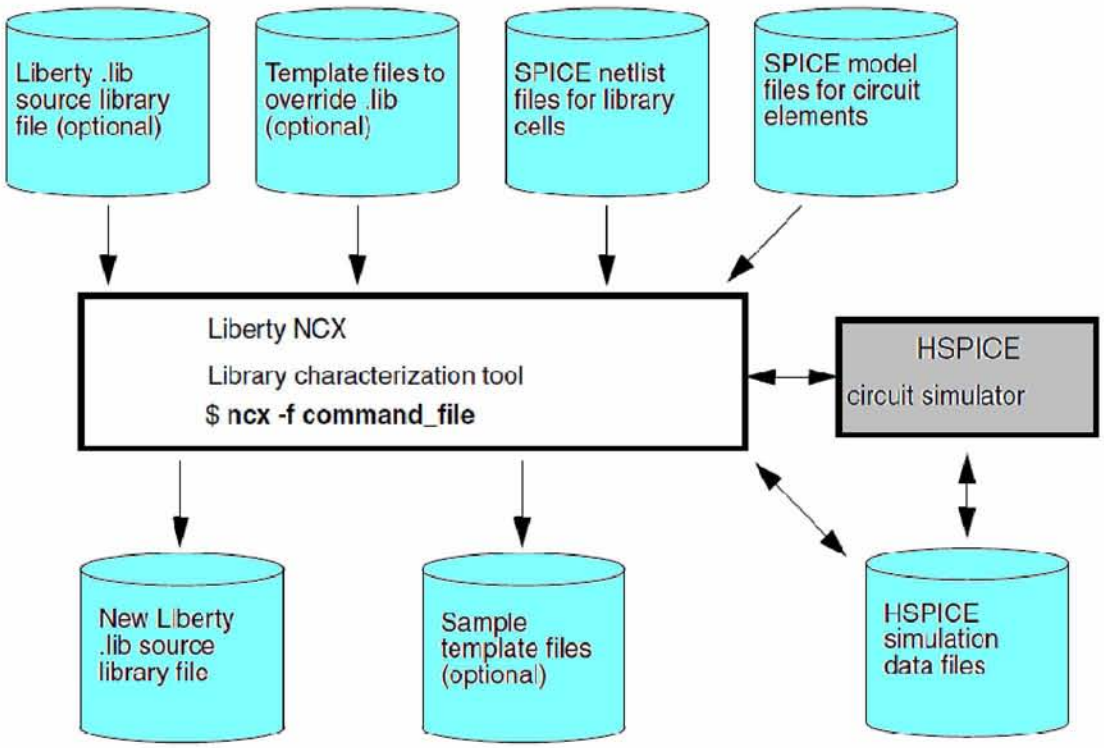


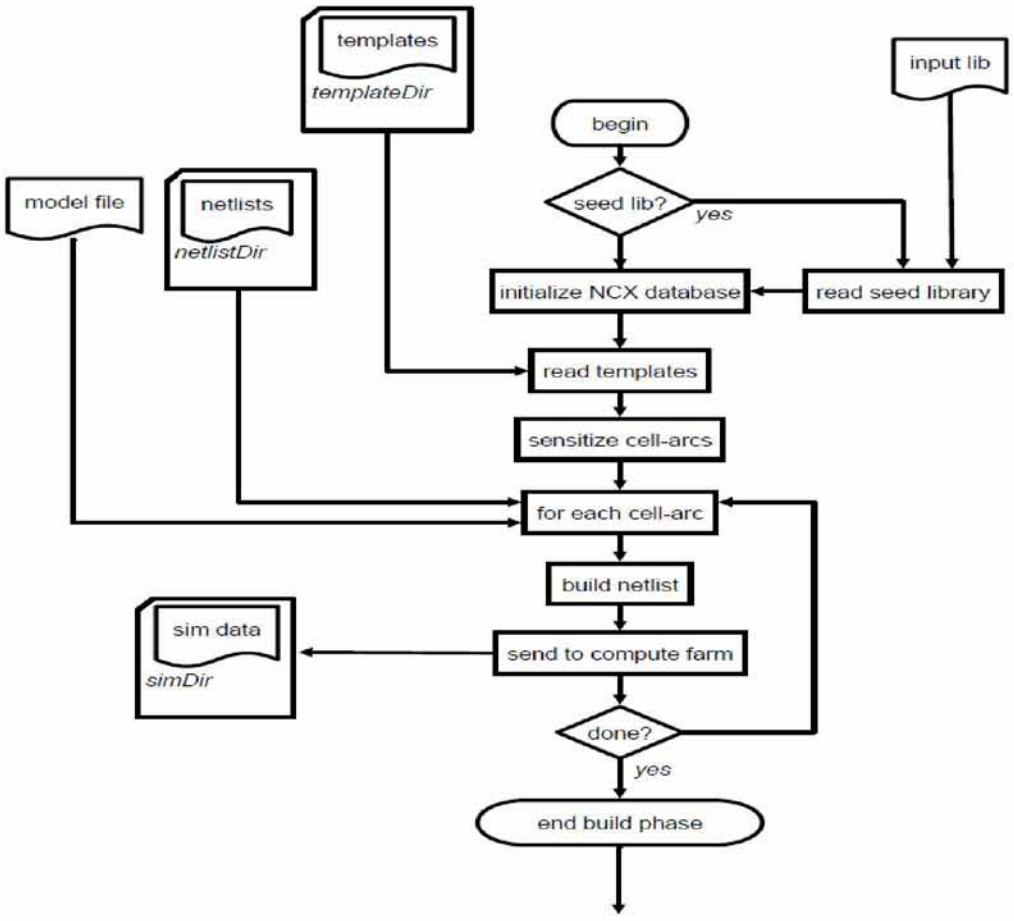
Figure 9 : Liberty NCX Characterization Data

Liberty™ NCX:

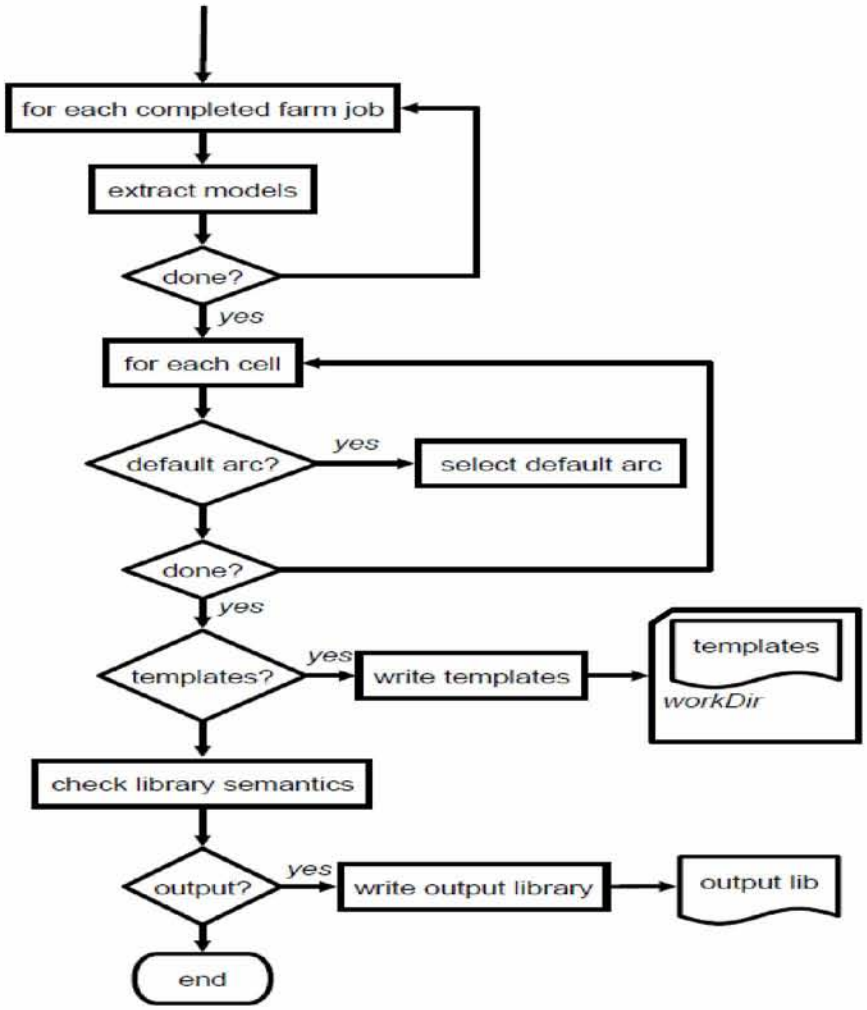
- reads cell descriptions from the seed library or cell template files
- reads SPICE netlists and models
- sets up SPICE simulations of the cells
- runs the simulations using HSPICE or a compatible simulator
- gathers the simulation results
- generates the library cell characterization data
- writes out a .lib library file containing the newly characterized cells.

Liberty™ NCX gets the cell functionality information from the descriptions contained in the existing .lib seed library or the user-supplied template files. More cells can be added or the parameters for the existing cells can be overridden by providing a library template file for global parameters or individual cell template files for the cells and cell parameters being characterized.

Figure 10: **Liberty™ NCX Operation**



Continue to the next figure



Liberty™ NCX Usage Flows

Liberty™ NCX can be used to perform two kinds of tasks: characterization and library formatting. Characterization means creating a new library or adding new cell models to an existing library. Library formatting means converting the data in an existing library to a different format. The types of library formatting are:

- compacting/expanding CCS models
- variation-aware library merging
- CCS Noise model merging

- CCS-to-ECSM conversion
- and CCS-to-NLDM conversion.

In characterization there are two flows, creation and re-characterization. The library creation flow uses a library template, a library index file and cell level templates to generate a new library. The templates can be created from scratch or generated by Liberty™ NCX from a similar seed library. The templates would have to be modified as required to get the desired library. To characterize a cell, it runs SPICE simulations of the cell under various conditions and records the cell behavior into its database. Then it writes out the cell model in Liberty format. Library Compiler or a similar tool can then be used to compile the .lib description into a form that can be used by timing, power, and noise analysis tools.

The re-characterization flow reads in an existing seed library and recharacterizes the timing tables, the power tables, the design rules (max-capacitance, max-transition), the pin capacitances, and the setup/hold constraints as instructed with specific PVT (Process Voltage Temperature) corners. In this thesis we are dealing with the creation flow in different corner each time which is described in the next chapter.

Chapter 6:
Standard Cells Characterization with Liberty™ NCX

The NanGate 45nm Open Cell Library is an open-source, standard-cell library provided for the purposes of testing and exploring EDA flows. NanGate has developed it for open use. The library is intended to aid university research programs and organizations in developing flows, developing circuits and exercising new algorithms. In this release the Open Cell Library contains 33 different functions ranging from buffers to scan flip-flops with set and reset. With multiple drive strength variants, the library includes over 100 different standard cells.

The library was generated using NanGate's Library Creator™ and the 45nm FreePDK Base Kit from North Carolina State University (NCSU) and characterized with NanGate's NanSpice™ using the Predictive Technology Model (PTM) from Arizona State University (ASU).

The 45nm Open Cell Library contains the following views:

- Liberty (.lib) formatted libraries with CCS Timing, ECSM Timing and NLPM/NLDM data (slow, typical, fast, low temperature and worst low temperature corners)
- Geometric library in Library Exchange Format (LEF)
- Simulation libraries in Verilog and Spice (pre and post parasitic extracted netlists)
- Cell layouts in GDSII
- Schematics in EDIF and PNG formats
- Library databook in HTML format
- OpenAccess database containing layouts and netlists

In this thesis we use as inputs:

- the Liberty formatted library with CCS Timing(typical corners)
- SPICE netlist files for each cell being characterized (post-layout parasitic parameters)
- SPICE model file for the circuit elements used in the cell netlists
- Configuration files that will be described later.

Our goal is to produce, through the characterization process, a Liberty formatted library that contains CCS Timing and Power models which can then be used for timing

and power analysis with compatible tools such as Library Compiler, IC Compiler, Design Compiler and Prime Time.

The file below is the first configuration file (run1):

```
set input_library     ./NangateOpenCellLibrary_typical_conditional_ccs.lib
set work_dir       mywork
set output_library     ./NangateOpenCellLibrary_typical_conditional_25_0.6_CCS.lib
set templates     true
set output_template_dir     templates
set compact     false
set ccs_timing     false
set ccs_power     false
set farm_type     nofarm
```

The following table briefly describes the characterization settings in the first configuration file:

Command setting	Description	Used Value
input_library	The name of the existing Liberty™ (.lib) source file. Liberty™ NCX uses this library to gather cell function descriptions. If no input library is specified, Liberty™ NCX generates an entirely new library using the library and the cell templates.	
work_dir	The name of the working directory used by Liberty™ NCX to write all working files.	
output_library	The name of the new Liberty (.lib) library file created by Liberty™ NCX.	
templates	When this attribute is set to	true

	true causes the generation of sample templates for the library and cells. The generated template files can be modified to specify the characterization parameters for future runs of Liberty™ NCX. The files are written to the directory defined by output_template_dir.	
output_template_dir	The name of the directory where the output template files are to be written. These files can be used as input template files in future runs of Liberty™ NCX.	templates
compact	If CCS models are being generated, this attribute causes generation of driver models in compact form. Compact models contain the same current-source information as standard models, but with the information encoded as a set of current-versus-voltage waveform parameters to reduce the size of the characterized library. When the compact command is set to true (the default), Liberty™ NCX performs compaction for CCS power and CCS timing models.	false
ccs_timing	This attribute causes acquisition of CCS timing models.	false

ccs_power	This attribute causes acquisition of CCS power models. When the ccs_power command is set to true (the default), Liberty™ NCX generates CCS power models.	false
farm_type	This attribute specifies the multiprocessor farm system. Can be set to LSF (Platform Computing LSF), SGE (Sun Grid Engine), user, or nofarm (local processor usage only). For LSF or SGE, you must run Liberty NCX on a submit machine that can accept bsub (LSF) or qsub (SGE) job submission commands. Setting user allows you to specify your own script for managing the farm.	nofarm

In the second step we use the following configuration file (run2):

```
set model_file    ./model_nom.typ
set netlist_dir    ./netlists
set netlist_suffix    .sp
set work_dir    mywork
set output_library    ./NangateOpenCellLibrary_typical_conditional_25_0.6_CCS.lib
set input_template_dir    ./templates
set simulator_exec    ./hspice
set timing    true
set ccs_timing    true
set nldm    false
```

set fix_nldm_timing none

set power true

set ccs_power true

set compact false

set farm_type nofarm

The following table describes the settings that are not included in the first configuration file:

Command setting	Description	Used Value
model_file	This attribute specifies the name of the SPICE model file containing the process and device models for the circuit elements used in the cell netlists.	
netlist_dir	This attribute specifies the directory containing the SPICE netlists for the cells being characterized. There must be a separate SPICE netlist file for each cell being characterized.	
netlist_suffix	This attribute specifies the file name extension used by Liberty™ NCX to recognize the SPICE netlist files in the cell netlist directory. If the SPICE netlist file names had the extension .spc the value .spc should be used.	.sp
input_template_dir	The name of the directory containing the input template files. These files specify the cells that need to be characterized and the library and cell characterization parameters.	
simulator_exec	This attribute specifies the absolute path to the SPICE circuit simulator executable. The supported simulators are hspice, eldo, and spectre.	
timing	This attribute causes acquisition of	true

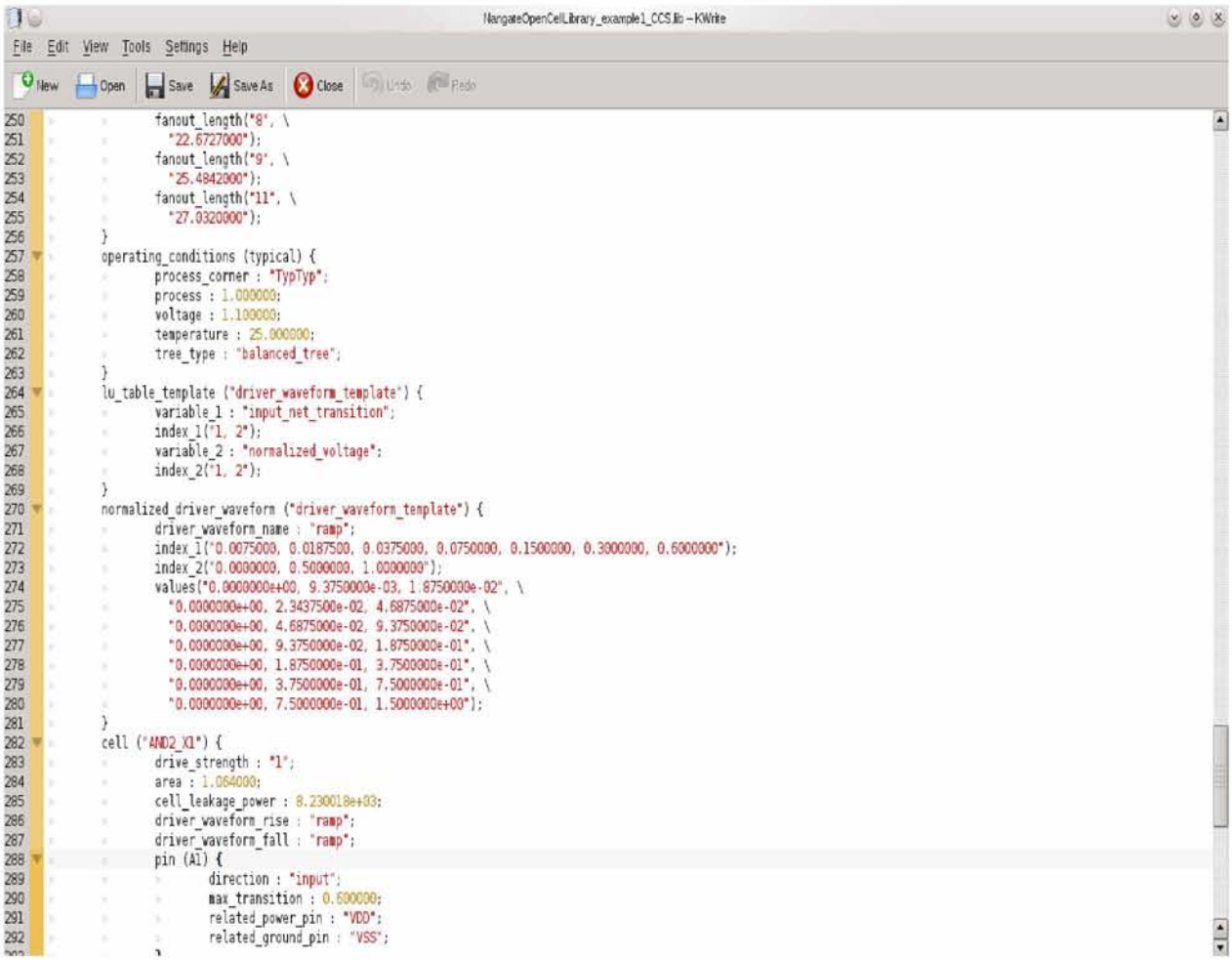
	NLDM and CCS timing models.	
power	This attribute causes acquisition of NLPM and CCS power models.	true
fix_nldm_timing	This attribute ensures monotonically increasing cell delays with increasing output load capacitance in the generated NLDM timing models. This command accepts the following values: delay, slew, both, and none. When fix_nldm_timing is set to delay, the tool ensures monotonically increasing cell delays with increasing output load capacitance in the generated NLDM timing models. When the command is set to slew, the tool ensures monotonically increasing cell transitions with increasing output load capacitance in the generated NLDM timing models. When the command is set to both, the tool ensures monotonicity for cell delays and cell transition tables in the generated NLDM models. When the command is set to none, the tool does not ensure monotonicity for cell delay and transition tables. For backward compatibility, the tool internally translates true to delay and false to none.	none
nldm	When this command is set to true, it causes acquisition of NLDM timing models.	

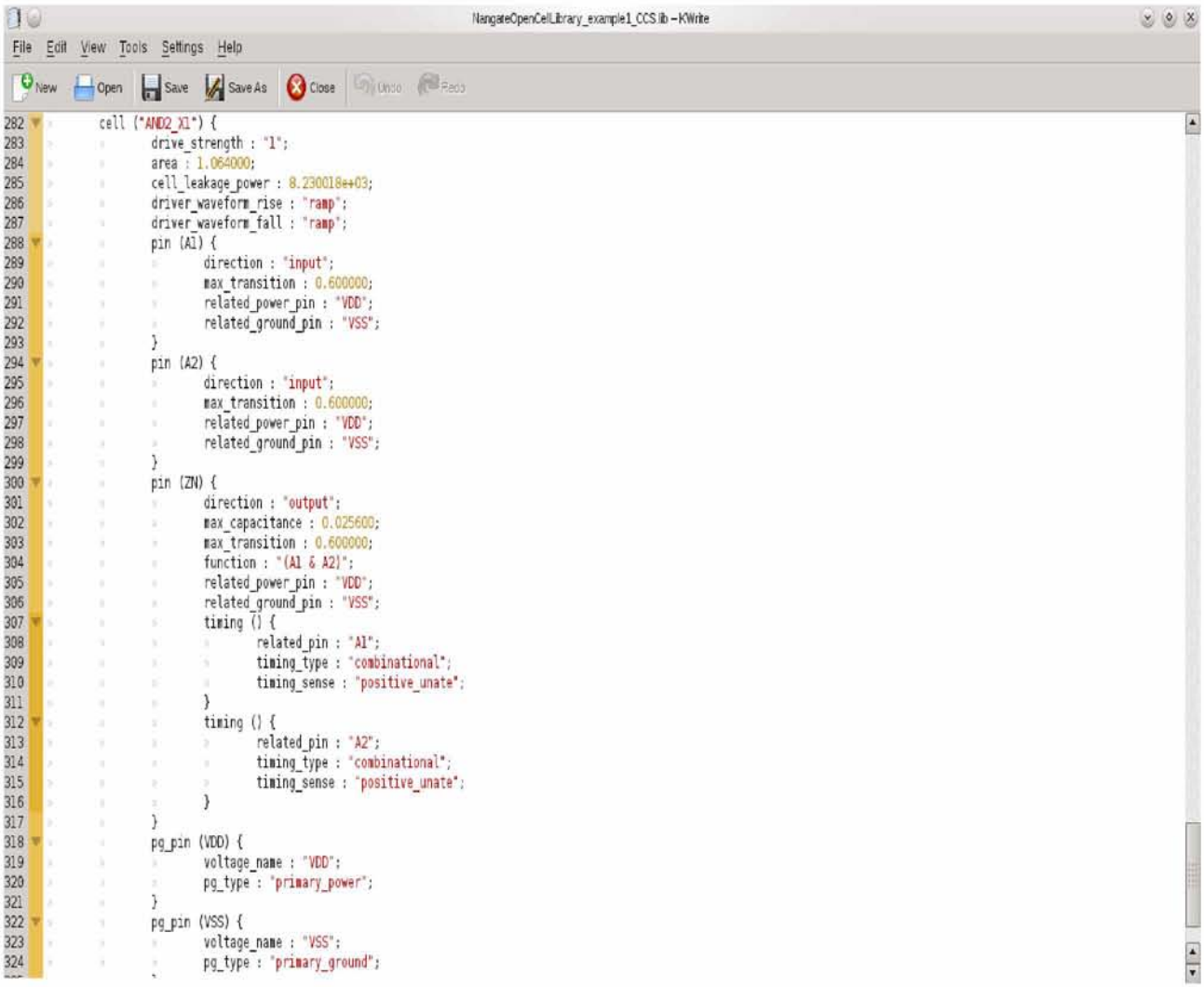
We set to true the commands:

- timing
- ccs_timing
- power
- ccs_power

because we want the desired output library to contain CCS Timing and Power models.

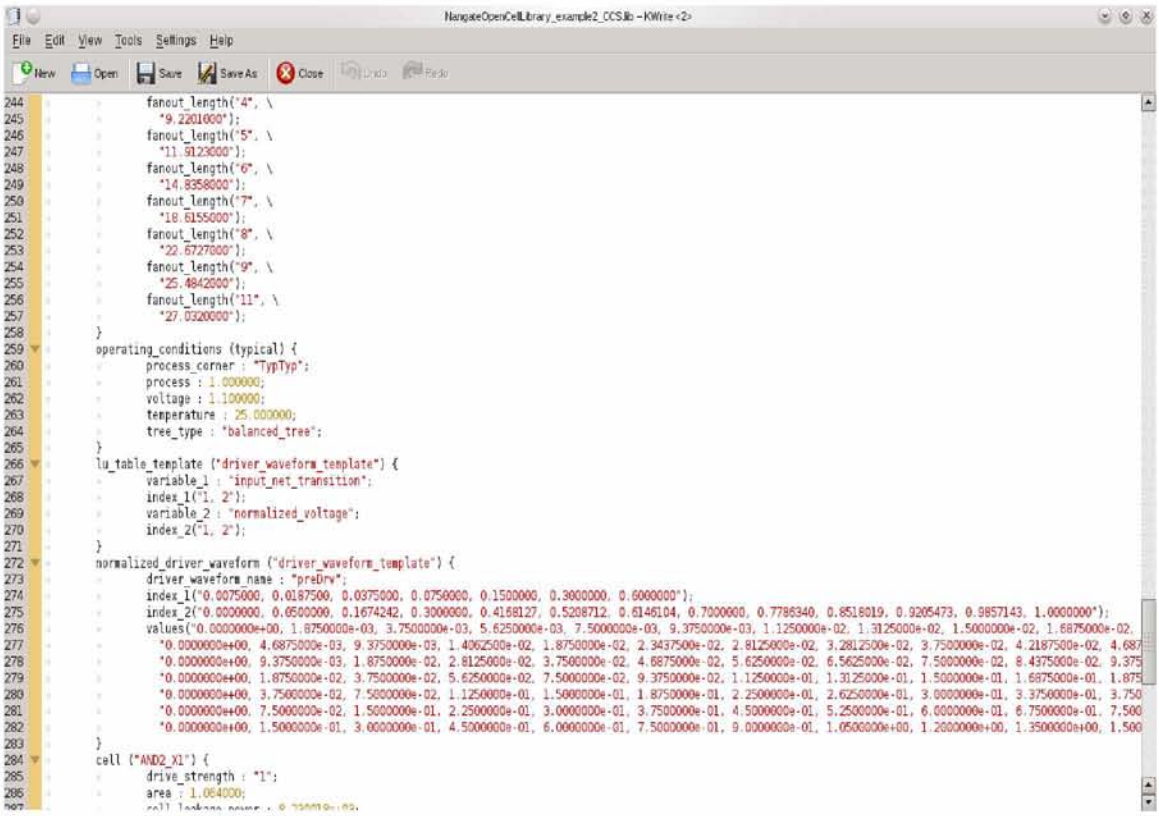
If we set the above commands to false, then the output library does not support neither CCS nor NLDM/NLPM models as we see below.



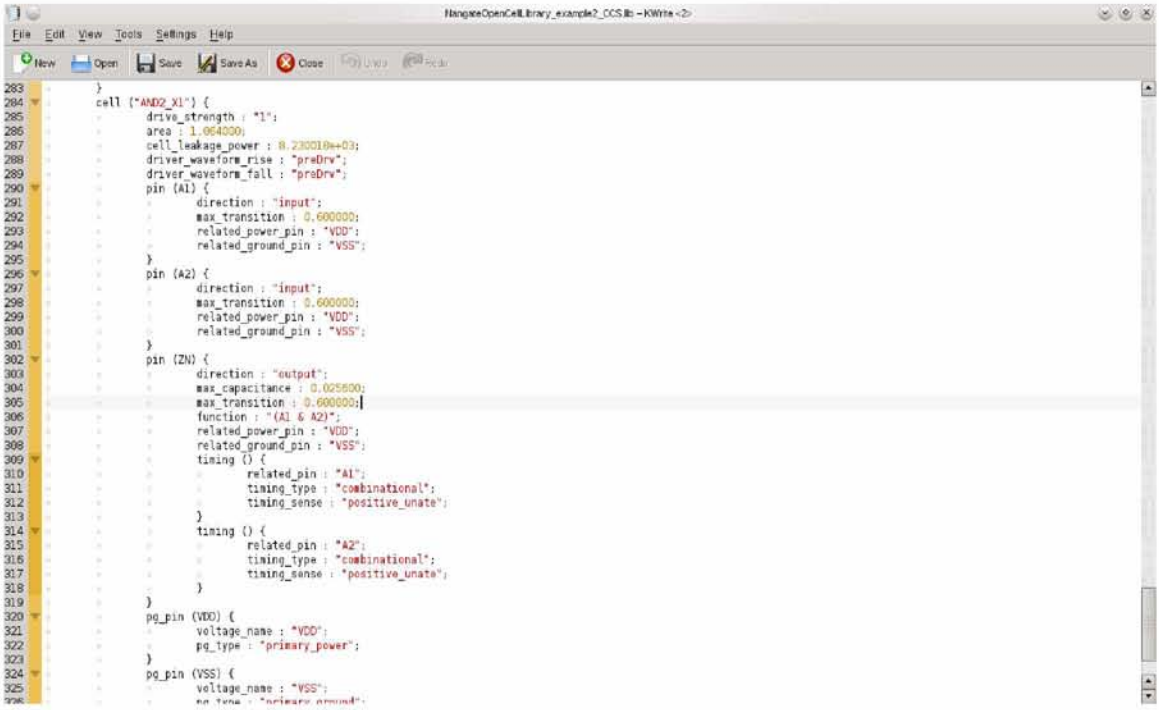


Example 1

Moreover, if we set the commands timing and power to true and the commands ccs_timing and ccs_power to false, then the output library supports both CCS and NLDM/NLPM models.

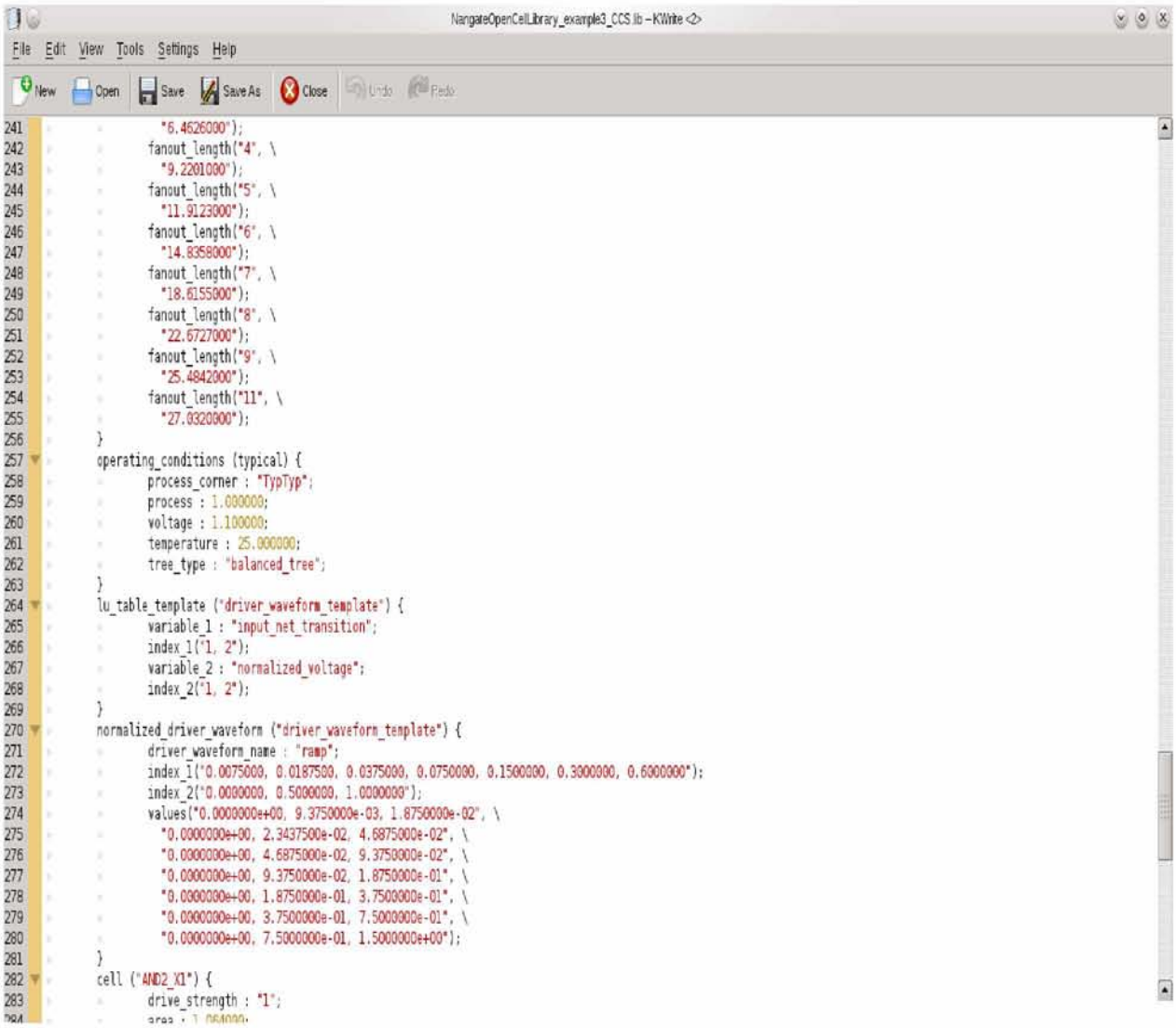


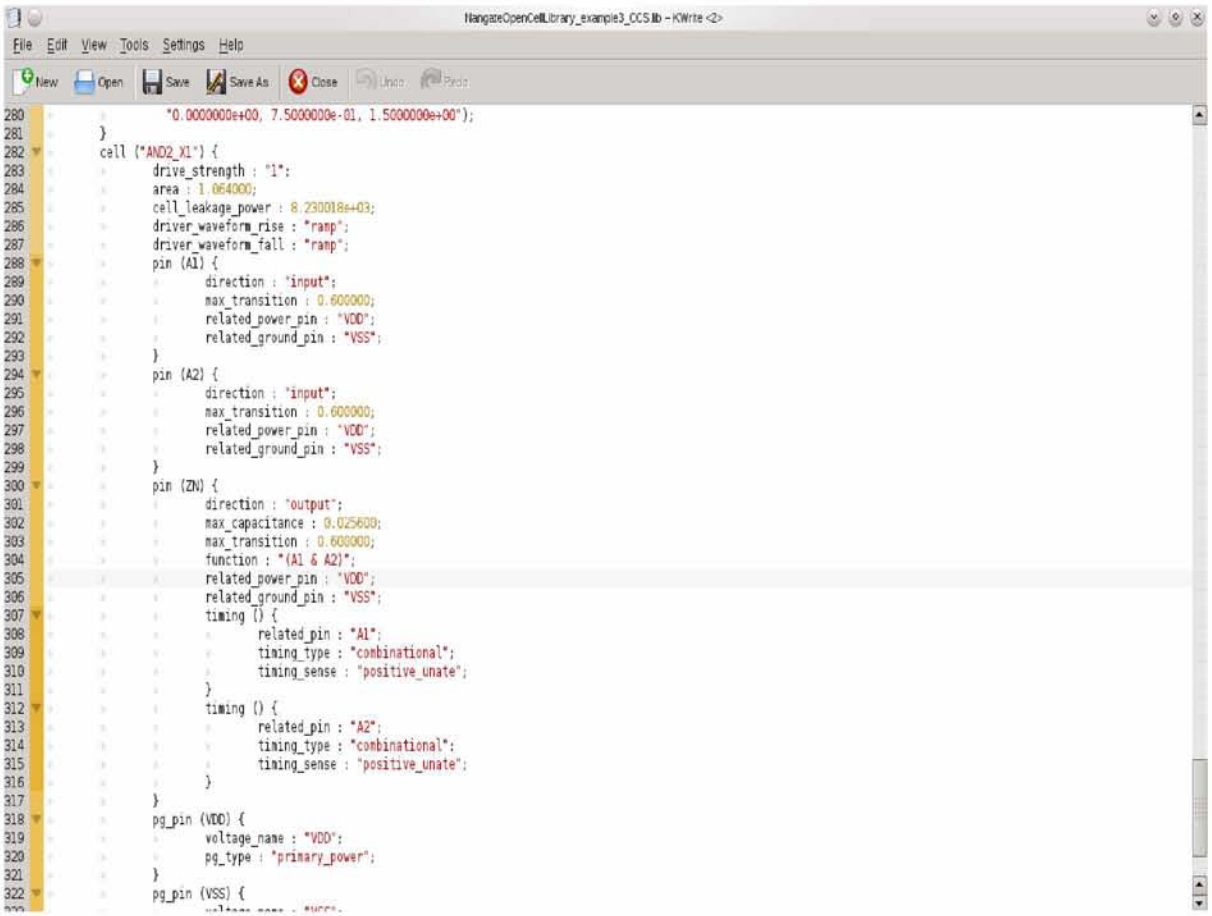
-



Example 2

Also, we set nldm command to false in order not to cause acquisition of NLDM timing models. If we set it to true and set the commands timing, power, ccs_timing, ccs_power to false we have the following output library.





Example 3

In order to execute the first configuration file the following command must be typed in the Linux bash prompt:

- ncx –f run1

This step produces three types of template files which are stored in the templates directory. These are the library template files, the cell template files and the library index file. A library template file specifies parameters such as units, delay thresholds, slew thresholds and derating factors. A cell template file specifies cell characterization parameters, such as scaling factors, area and sensitization.

The library template file that is produced at this first step is the following:

```
* Generated by Liberty NCX vD-2010.06
Date : "Fri 17 Jul 2009, 20:52:36" ;
revision : "revision 1.0" ;
comment : "Copyright (c) 2004-2008 Nangate Inc. All Rights Reserved." ;
technology : cmos ;
delay_model : table_lookup ;
in_place_swap_mode : match_footprint ;
library_features : report_delay_calculation report_power_calculation ;
time_unit : 1ns ;
leakage_power_unit : 1pW ;
voltage_unit : 1V ;
current_unit : 1uA ;
pulling_resistance_unit : 1kohm ;
capacitive_load_unit : 1 pf ;
nom_process : 1.0000000 ;
nom_temperature : 25.0000000 ;
nom_voltage : 1.1000000 ;
default_operating_conditions : typical ;
slew_lower_threshold_pct_fall : 30.0000000 ;
slew_lower_threshold_pct_rise : 30.0000000 ;
slew_upper_threshold_pct_fall : 70.0000000 ;
slew_upper_threshold_pct_rise : 70.0000000 ;
slew_derate_from_library : 1.0000000 ;
input_threshold_pct_fall : 50.0000000 ;
input_threshold_pct_rise : 50.0000000 ;
output_threshold_pct_fall : 50.0000000 ;
output_threshold_pct_rise : 50.0000000 ;
default_leakage_power_density : 0.0000000e+00 ;
default_cell_leakage_power : 0.0000000e+00 ;
default_inout_pin_cap : 1.0000000 ;
default_input_pin_cap : 1.0000000 ;
```

```
default_output_pin_cap : 0.0000000e+00 ;
default_fanout_load : 1.0000000 ;
default_wire_load : 5K_hvratio_1_1 ;
ncx_use_pg_pins : true ;
ncx_nlpm_mode : aggregate ;
operating_conditions typical {
  define process_corner : TypTyp ;
  process : 1.0000000 ;
  voltage : 1.1000000 ;
  temperature : 25.0000000 ;
  tree_type : balanced_tree ;
}
wire_load 1K_hvratio_1_4 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0755970 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.3207000 ;
  fanout_length : 2 2.9813000 ;
  fanout_length : 3 5.1135000 ;
  fanout_length : 4 7.6639000 ;
  fanout_length : 5 10.0334000 ;
  fanout_length : 6 12.2296000 ;
  fanout_length : 8 19.3185000 ;
}
wire_load 1K_hvratio_1_2 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0762066 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.3216000 ;
  fanout_length : 2 2.8855000 ;
  fanout_length : 3 4.6810000 ;
```

```
  fanout_length : 4 6.7976000 ;
  fanout_length : 5 9.4037000 ;
  fanout_length : 6 13.0170000 ;
  fanout_length : 8 24.1720000 ;
}
wire_load 1K_hvratio_1_1 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0765020 ;
  slope : 6.2836880 ;
  fanout_length : 1 1.3446000 ;
  fanout_length : 2 2.8263000 ;
  fanout_length : 3 4.7581000 ;
  fanout_length : 4 7.4080000 ;
  fanout_length : 5 10.9381000 ;
  fanout_length : 6 15.7314000 ;
  fanout_length : 8 29.7891000 ;
}
wire_load 3K_hvratio_1_4 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0799410 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.8234000 ;
  fanout_length : 2 4.5256000 ;
  fanout_length : 3 7.5342000 ;
  fanout_length : 4 10.6237000 ;
  fanout_length : 5 13.5401000 ;
  fanout_length : 6 16.3750000 ;
  fanout_length : 7 18.6686000 ;
  fanout_length : 8 19.4348000 ;
  fanout_length : 10 20.9672000 ;
}
```

```
wire_load 3K_hvratio_1_2 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0800407 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.6615000 ;
  fanout_length : 2 3.9827000 ;
  fanout_length : 3 6.6386000 ;
  fanout_length : 4 9.6287000 ;
  fanout_length : 5 12.8485000 ;
  fanout_length : 6 16.4145000 ;
  fanout_length : 7 20.0747000 ;
  fanout_length : 8 22.6325000 ;
  fanout_length : 10 21.7173000 ;
}
wire_load 3K_hvratio_1_1 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0811883 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.5771000 ;
  fanout_length : 2 3.9330000 ;
  fanout_length : 3 6.6217000 ;
  fanout_length : 4 9.7638000 ;
  fanout_length : 5 13.5526000 ;
  fanout_length : 6 18.1322000 ;
  fanout_length : 7 22.5871000 ;
  fanout_length : 8 25.1074000 ;
  fanout_length : 10 30.1480000 ;
}
wire_load 5K_hvratio_1_4 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
```

```
  area : 0.0803598 ;
  slope : 5.0000000 ;
  fanout_length : 1 2.0449000 ;
  fanout_length : 2 4.4094000 ;
  fanout_length : 3 7.2134000 ;
  fanout_length : 4 10.4927000 ;
  fanout_length : 5 13.9420000 ;
  fanout_length : 6 18.0039000 ;
  fanout_length : 7 23.9278000 ;
  fanout_length : 8 30.8475000 ;
  fanout_length : 9 34.9441000 ;
  fanout_length : 11 43.1373000 ;
}
wire_load 5K_hvratio_1_2 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0802320 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.6706000 ;
  fanout_length : 2 3.7951000 ;
  fanout_length : 3 6.2856000 ;
  fanout_length : 4 9.1309000 ;
  fanout_length : 5 12.1420000 ;
  fanout_length : 6 15.6918000 ;
  fanout_length : 7 20.1043000 ;
  fanout_length : 8 24.2827000 ;
  fanout_length : 9 27.3445000 ;
  fanout_length : 11 35.3421000 ;
}
wire_load 5K_hvratio_1_1 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0815905 ;
```

```
  slope : 5.0000000 ;
  fanout_length : 1 1.7460000 ;
  fanout_length : 2 3.9394000 ;
  fanout_length : 3 6.4626000 ;
  fanout_length : 4 9.2201000 ;
  fanout_length : 5 11.9123000 ;
  fanout_length : 6 14.8358000 ;
  fanout_length : 7 18.6155000 ;
  fanout_length : 8 22.6727000 ;
  fanout_length : 9 25.4842000 ;
  fanout_length : 11 27.0320000 ;
}

include NangateOpenCellLibrary_typical_conditional_25_0.6_CCS.indexes ;


do {
  TLAT_X1
  SDFF_X2
  SDFF_X1
  SDFFS_X2
  SDFFS_X1
  SDFFR_X2
  SDFFR_X1
  SDFFRS_X2
  SDFFRS_X1
  LOGIC1_X1
  LOGIC0_X1
  FILLCELL_X8
  FILLCELL_X4
  FILLCELL_X32
  FILLCELL_X2
  FILLCELL_X16
```

FILLCELL_X1
DLL_X2
DLL_X1
DLH_X2
DLH_X1
DFF_X2
DFF_X1
DFFS_X2
DFFS_X1
DFFR_X2
DFFR_X1
DFFRS_X2
DFFRS_X1
CLKGATE_X8
CLKGATE_X4
CLKGATE_X2
CLKGATE_X1
CLKGATETST_X8
CLKGATETST_X4
CLKGATETST_X2
CLKGATETST_X1
AND2_X1
AND2_X2
AND2_X4
AND3_X1
AND3_X2
AND3_X4
AND4_X1
AND4_X2
AND4_X4
ANTENNA_X1
AOI211_X1
AOI211_X2

AOI211_X4
AOI21_X1
AOI21_X2
AOI21_X4
AOI221_X1
AOI221_X2
AOI221_X4
AOI222_X1
AOI222_X2
AOI222_X4
AOI22_X1
AOI22_X2
AOI22_X4
BUF_X1
BUF_X16
BUF_X2
BUF_X32
BUF_X4
BUF_X8
CLKBUF_X1
CLKBUF_X2
CLKBUF_X3
FA_X1
HA_X1
INV_X1
INV_X16
INV_X2
INV_X32
INV_X4
INV_X8
MUX2_X1
MUX2_X2
NAND2_X1

NAND2_X2
NAND2_X4
NAND3_X1
NAND3_X2
NAND3_X4
NAND4_X1
NAND4_X2
NAND4_X4
NOR2_X1
NOR2_X2
NOR2_X4
NOR3_X1
NOR3_X2
NOR3_X4
NOR4_X1
NOR4_X2
NOR4_X4
OAI211_X1
OAI211_X2
OAI211_X4
OAI21_X1
OAI21_X2
OAI21_X4
OAI221_X1
OAI221_X2
OAI221_X4
OAI222_X1
OAI222_X2
OAI222_X4
OAI22_X1
OAI22_X2
OAI22_X4
OAI33_X1

```
 OR2_X1
 OR2_X2
 OR2_X4
 OR3_X1
 OR3_X2
 OR3_X4
 OR4_X1
 OR4_X2
 OR4_X4
 TBUF_X1
 TBUF_X16
 TBUF_X2
 TBUF_X4
 TBUF_X8
 TINV_X1
 XNOR2_X1
 XNOR2_X2
 XOR2_X1
 XOR2_X2
}
```

If we use the above library template file as input in the second characterization step and run the congifuration file 2, the produced library had some errors, as far as the reference time is concerned. The reference_time attribute represents the time at which the input waveform crosses the rising or falling input delay threshold. The reference_time attribute must be identical for the same input transition time and edge (either rise or fall) across all capacitive loads. This ensures that the input characterization waveform is consistent across all characterization loads.

These errors are highlighted in the output library extract that follows:

```
3456        values("0.0008546663, 4.963764, 4.133261, 2.613386, 0.7445581, 0.2246704, -0.02369");
3457    }
3458    vector ("ccs_power_output_switching_0") {
3459        reference_time : 1.022544;
3460        index_1("0.0187500");
3461        index_2("0.0004000");
3462        index_3("1.0000000, 1.0053457, 1.0160370, 1.0267284, 1.0481111, 1.0588025, 1.0908766, 1.3581605, 1.4543828, 1.5185309,
3463        values("0.0008546665, 2.662641, 4.076142, 3.878189, 5.616803, 3.763288, 2.85431, 1.284617, 0.5381629, 0.2319812, 0.116
3464    }
3465    vector ("ccs_power_output_switching_0") {
3466        reference_time : 1.022544;
3467        index_1("0.0187500");
3468        index_2("0.0008000");
3469        index_3("1.0000000, 1.0066003, 1.0330014, 1.0462019, 1.0594025, 1.0990041, 1.4026167, 1.5214216, 1.5874244, 1.6798282,
3470        values("0.0008546663, 2.954142, 4.17976, 5.731007, 3.997799, 3.028922, 1.563103, 0.6301195, 0.3200116, 0.1506251, 0.047
3471    }
3472    vector ("ccs_power_output_switching_0") {
3473        reference_time : 1.022540;
3474        index_1("0.0187500");
3475        index_2("0.0016000");
3476        index_3("1.0000000, 1.0084695, 1.0254086, 1.0423476, 1.0762257, 1.6013364, 1.7199098, 1.8046051, 1.8893003, 2.0332823")
3477        values("0.0008546663, 3.1356, 3.805115, 5.463246, 3.655042, 1.184876, 0.5292718, 0.306631, 0.1710553, 0.04755661");
3478    }
3479    vector ("ccs_power_output_switching_0") {
3480        reference_time : 1.022539;
3481        index_1("0.0187500");
3482        index_2("0.0032000");
3483        index_3("1.0000000, 1.0127390, 1.0382170, 1.1146510, 1.7261228, 2.0063808, 2.1592487, 2.2611607, 2.5286796");
3484        values("0.0008546663, 3.227767, 4.996208, 3.621795, 2.049557, 0.6841255, 0.3120988, 0.1972339, 0.02004733");
3485    }
3486    vector ("ccs_power_output_switching_0") {
3487        reference_time : 1.022540;
3488        index_1("0.0187500");
3489        index_2("0.0064000");
3490        index_3("1.0000000, 1.0192710, 1.0578131, 1.0963551, 2.2526165, 2.5994950, 2.8307472, 2.9849154, 3.3510649");
3491        values("0.0008546663, 3.49686, 5.161347, 4.020049, 1.991259, 0.8615484, 0.4236318, 0.2816415, 0.04626827");
```

Line: 1 Col: 1 | INS | LINE | C | NangateOpenCellLibrary_typical_conditional_25_0.6_error_CCS.lib

Example 4

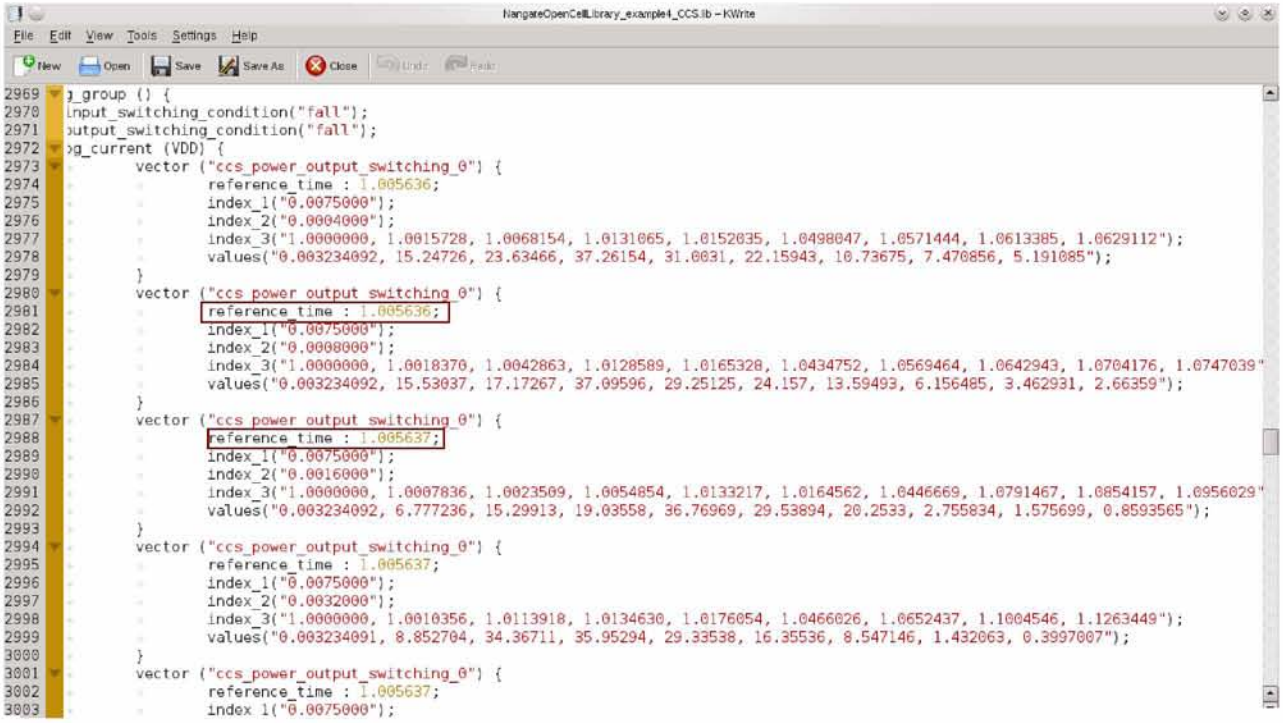
This extract shows that for an input transition time of 0.0187500ns, the reference_time for a load of 0.0008000pf is 1.022544ns, is 1.022540ns for a load of 0.0016000ns and is 1.022539ns for a load of 0.0032000pf.

In order to correct these errors we made some modifications and added some attributes with different values each time. Some of the attempts that we made, before we conclude in the final format of the library template file, are shown below.

- First we tried different values for the following attributes and left the other ones with the default values that are shown in the sample template file above.

Attribute	Description
slew_lower_threshold_pct_fall	The lower threshold for

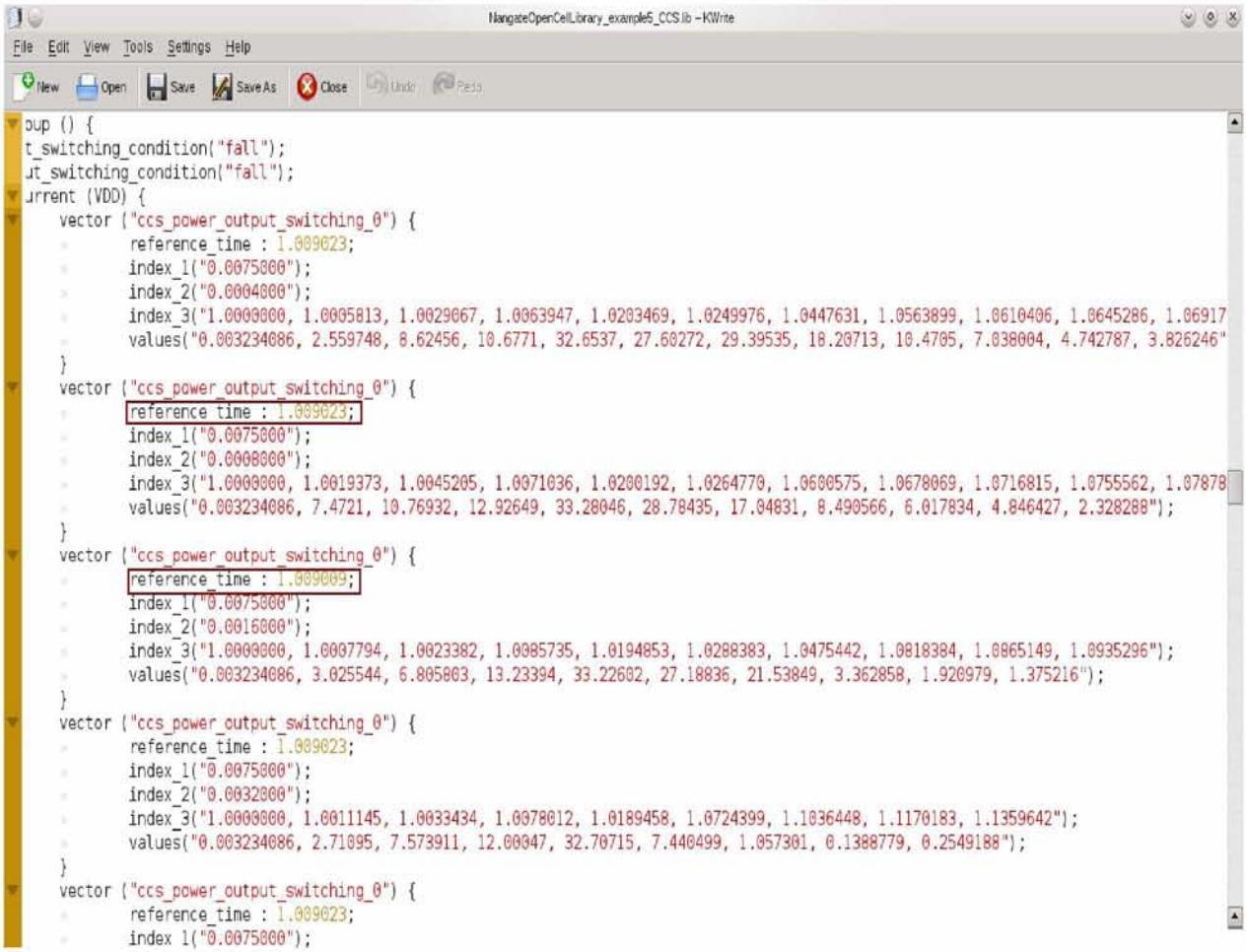
	determining the slew of a falling signal, expressed as a percentage of the rail voltage; typically used only in the library template.
slew_lower_threshold_pct_rise	The lower threshold for determining the slew of a rising signal, expressed as a percentage of the rail voltage; typically used only in the library template.
slew_upper_threshold_pct_fall	The upper threshold for determining the slew of a falling signal, expressed as a percentage of the rail voltage; typically used only in the library template.
slew_upper_threshold_pct_rise	The upper threshold for determining the slew of a rising signal, expressed as a percentage of the rail voltage; typically used only in the library template.



Example 5: output library extract for slew_lower_thresholds=20 and slew_upper_thresholds=80

- Then, we tried to change the value for

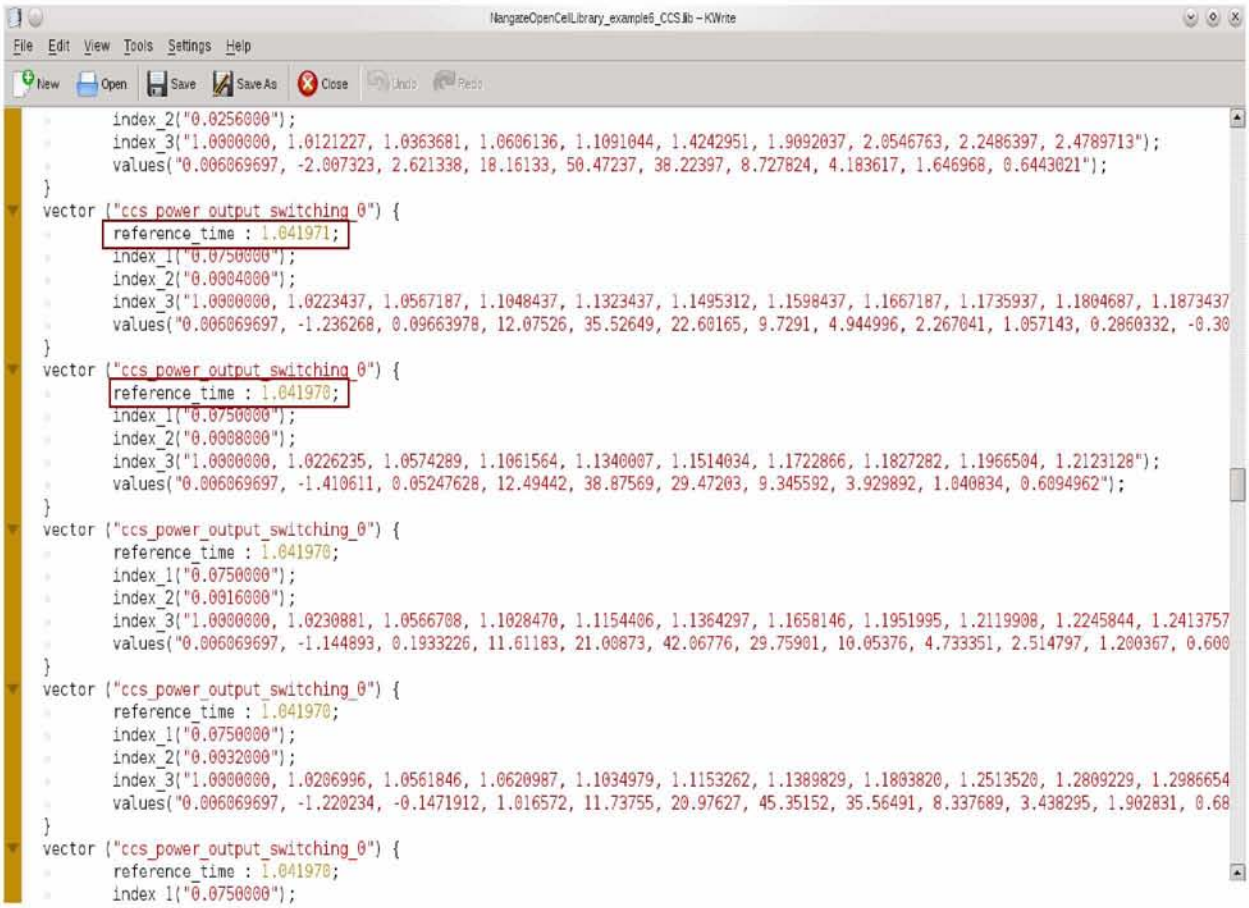
Attribute	Description
slew_derate_from_library	slew_derate_from_library = $\frac{simulation\,trip\,point\,range}{extrapolated\,trip\,point\,range}$



Example 6: output library extract for slew_derate_from_library=0.0

- In addition, we tried some values for

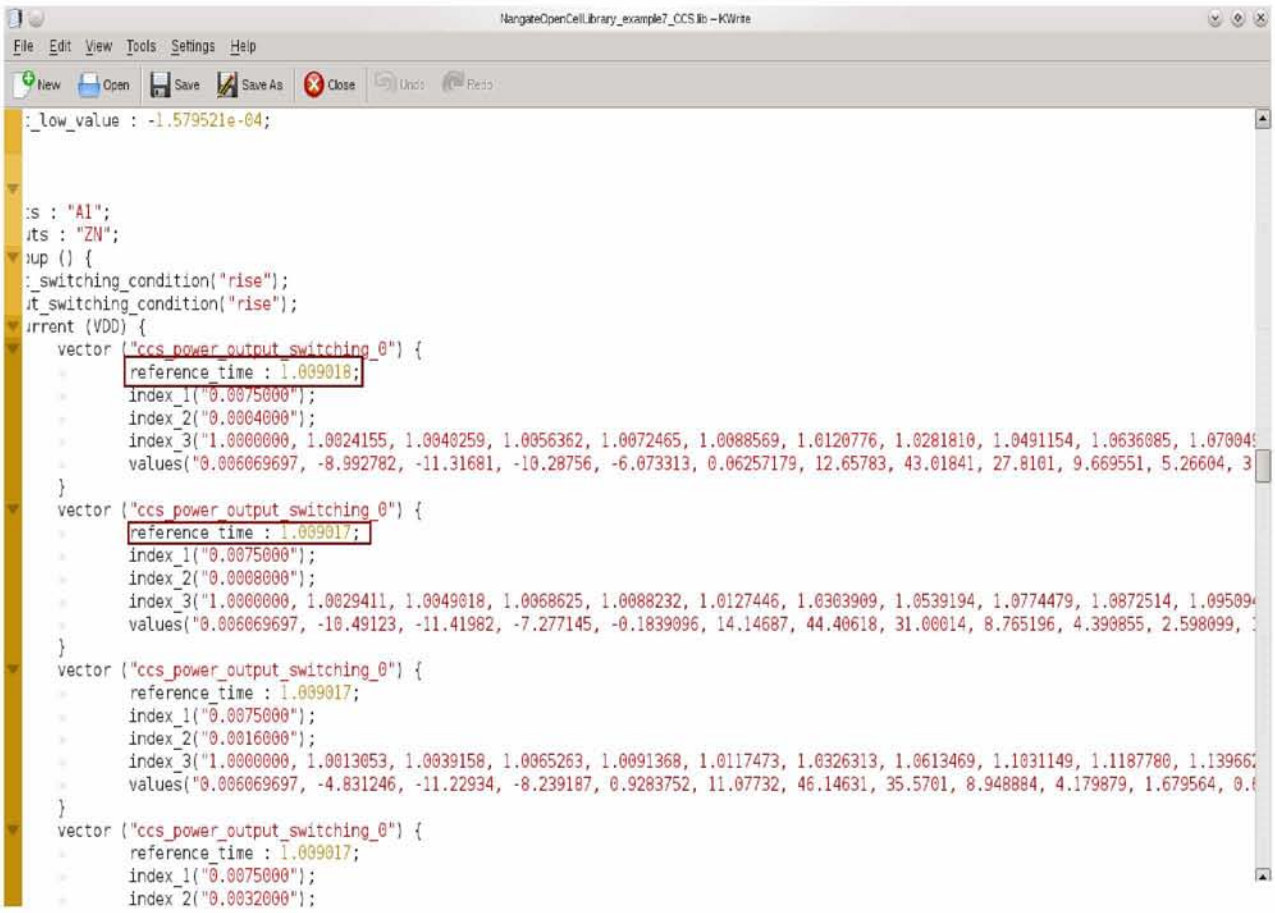
Attribute	Description
input_threshold_pct_fall	The timing threshold on input falling edge, as a percentage of the rail voltage
input_threshold_pct_rise	The timing threshold on input falling edge, as a percentage of the rail voltage.
output_threshold_pct_fall	The timing threshold on output falling edge, as a percentage of the rail voltage.
output_threshold_pct_rise	The timing threshold on output rising edge, as a percentage of the rail voltage.



<u>Example 7</u>: output library extract for input_threshold_pct_fall=80, input_threshold_pct_rise=20, output_threshold_pct_fall=20, output_threshold_pct_rise=80

- Also, we added the attribute min_pulse_width_slew

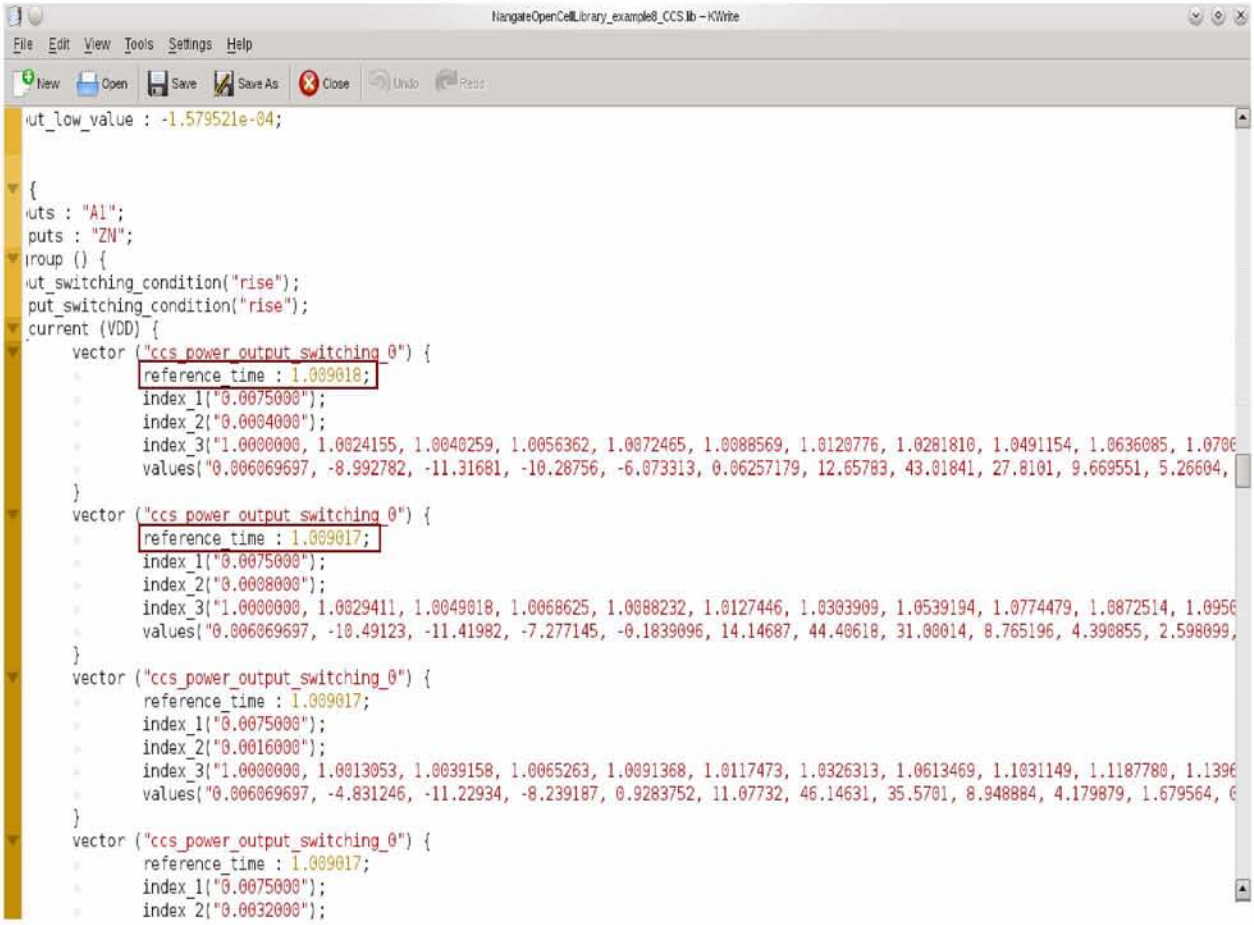
Attribute	Description
min_pulse_width_slew	Allows you to assign the input slew for pin-based minimum pulse width acquisition. You can define the value in library time units. The default is 0.1 ps. This attribute affects only pin-based minimum pulse width, not arc-based minimum pulse width.



Example 8: output library extract for min_pulse_width_slew=0.05 ps

- After, we tried different values for the attribute

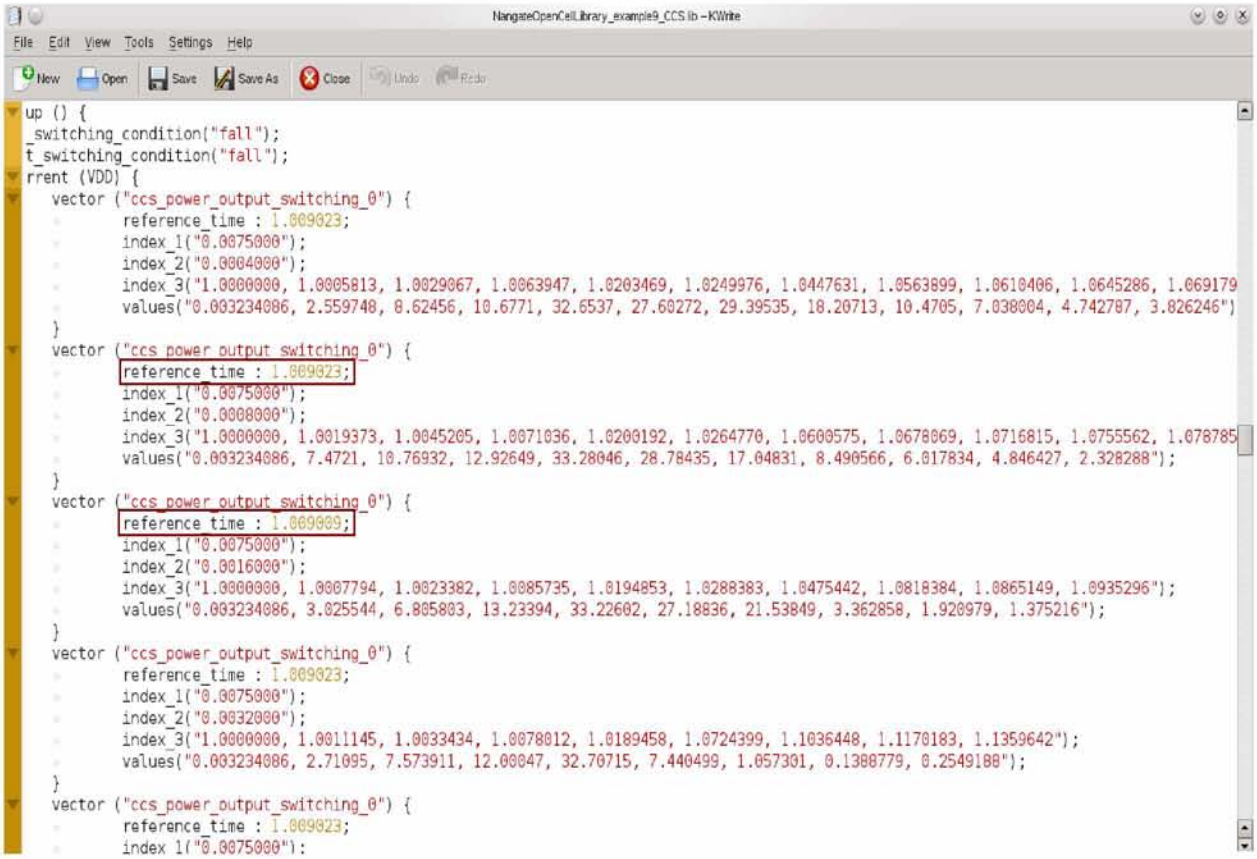
Attribute	Description
ccs_delay_tol	The acceptable difference between measured delay from simulation and delay obtained from the CCS waveform, expressed as a fraction of the measured delay from simulation.



Example 9: output library extract for ccs_delay_tol=0.0

- Also, we tried some values for the attributes

Attribute	Description
clk_tran	Clock transition time, in seconds, used for generating the input stimulus for a cell being characterized. This also represents the transition time for input signal edges.
clk_width	Clock width, in seconds, used for generating the input stimulus for a cell being characterized. This represents the minimum time between input signal toggles. This attribute might need to be set to a larger number in the case of failing constraint acquisitions for a cell, at the cost of more runtime.



Example 10: output library extract for clk_tran=0e-12, clk_width=0e-9

Finally, we concluded in the following combination of attributes and values which are the bold ones.

* Generated by Liberty NCX vD-2010.06

date : "Fri 17 Jul 2009, 20:52:36" ;

revision : "revision 1.0" ;

comment : "Copyright (c) 2004-2008 Nangate Inc. All Rights Reserved." ;

technology : cmos ;

delay_model : table_lookup ;

in_place_swap_mode : match_footprint ;

library_features : report_delay_calculation report_power_calculation ;

time_unit : 1ns ;

leakage_power_unit : 1pW ;

voltage_unit : 1V ;

current_unit : 1uA ;

pulling_resistance_unit : 1kohm ;

capacitive_load_unit : 1 pf ;

nom_process : 1.0000000 ;

nom_temperature : 25.0000000 ;

nom_voltage : 0.6000000 ;

default_operating_conditions : typical ;

slew_lower_threshold_pct_fall : 10.0000000 ;

slew_lower_threshold_pct_rise : 10.0000000 ;

slew_upper_threshold_pct_fall : 90.0000000 ;

slew_upper_threshold_pct_rise : 90.0000000 ;

slew_derate_from_library : 0.5000000 ;

input_threshold_pct_fall : 90.0000000 ;

input_threshold_pct_rise : 10.0000000 ;

output_threshold_pct_fall : 10.0000000 ;

output_threshold_pct_rise : 90.0000000 ;

default_leakage_power_density : 0.0000000e+00 ;

default_cell_leakage_power : 0.0000000e+00 ;

default_inout_pin_cap : 1.0000000 ;

```
default_input_pin_cap : 1.0000000 ;
default_output_pin_cap : 0.0000000e+00 ;
default_fanout_load : 1.0000000 ;
default_wire_load : 5K_hvratio_1_1 ;
ncx_use_pg_pins : true ;
clk_tran : 145e-12 ;
clk_width : 145e-9 ;
operating_conditions typical {
  define process_corner : TypTyp ;
  process : 1.0000000 ;
  voltage : 0.6000000 ;
  temperature : 25.0000000 ;
  tree_type : balanced_tree ;
}
wire_load 1K_hvratio_1_4 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0755970 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.3207000 ;
  fanout_length : 2 2.9813000 ;
  fanout_length : 3 5.1135000 ;
  fanout_length : 4 7.6639000 ;
  fanout_length : 5 10.0334000 ;
  fanout_length : 6 12.2296000 ;
  fanout_length : 8 19.3185000 ;
}
wire_load 1K_hvratio_1_2 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0762066 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.3216000 ;
```

```
  fanout_length : 2 2.8855000 ;
  fanout_length : 3 4.6810000 ;
  fanout_length : 4 6.7976000 ;
  fanout_length : 5 9.4037000 ;
  fanout_length : 6 13.0170000 ;
  fanout_length : 8 24.1720000 ;
}
wire_load 1K_hvratio_1_1 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0765020 ;
  slope : 6.2836880 ;
  fanout_length : 1 1.3446000 ;
  fanout_length : 2 2.8263000 ;
  fanout_length : 3 4.7581000 ;
  fanout_length : 4 7.4080000 ;
  fanout_length : 5 10.9381000 ;
  fanout_length : 6 15.7314000 ;
  fanout_length : 8 29.7891000 ;
}
wire_load 3K_hvratio_1_4 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0799410 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.8234000 ;
  fanout_length : 2 4.5256000 ;
  fanout_length : 3 7.5342000 ;
  fanout_length : 4 10.6237000 ;
  fanout_length : 5 13.5401000 ;
  fanout_length : 6 16.3750000 ;
  fanout_length : 7 18.6686000 ;
  fanout_length : 8 19.4348000 ;
```

```
  fanout_length : 10 20.9672000 ;
}
wire_load 3K_hvratio_1_2 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0800407 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.6615000 ;
  fanout_length : 2 3.9827000 ;
  fanout_length : 3 6.6386000 ;
  fanout_length : 4 9.6287000 ;
  fanout_length : 5 12.8485000 ;
  fanout_length : 6 16.4145000 ;
  fanout_length : 7 20.0747000 ;
  fanout_length : 8 22.6325000 ;
  fanout_length : 10 21.7173000 ;
}
wire_load 3K_hvratio_1_1 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0811883 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.5771000 ;
  fanout_length : 2 3.9330000 ;
  fanout_length : 3 6.6217000 ;
  fanout_length : 4 9.7638000 ;
  fanout_length : 5 13.5526000 ;
  fanout_length : 6 18.1322000 ;
  fanout_length : 7 22.5871000 ;
  fanout_length : 8 25.1074000 ;
  fanout_length : 10 30.1480000 ;
}
wire_load 5K_hvratio_1_4 {
```

```
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0803598 ;
  slope : 5.0000000 ;
  fanout_length : 1 2.0449000 ;
  fanout_length : 2 4.4094000 ;
  fanout_length : 3 7.2134000 ;
  fanout_length : 4 10.4927000 ;
  fanout_length : 5 13.9420000 ;
  fanout_length : 6 18.0039000 ;
  fanout_length : 7 23.9278000 ;
  fanout_length : 8 30.8475000 ;
  fanout_length : 9 34.9441000 ;
  fanout_length : 11 43.1373000 ;
}
wire_load 5K_hvratio_1_2 {
  capacitance : 1.7740000e-04 ;
  resistance : 0.0035714 ;
  area : 0.0802320 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.6706000 ;
  fanout_length : 2 3.7951000 ;
  fanout_length : 3 6.2856000 ;
  fanout_length : 4 9.1309000 ;
  fanout_length : 5 12.1420000 ;
  fanout_length : 6 15.6918000 ;
  fanout_length : 7 20.1043000 ;
  fanout_length : 8 24.2827000 ;
  fanout_length : 9 27.3445000 ;
  fanout_length : 11 35.3421000 ;
}
wire_load 5K_hvratio_1_1 {
  capacitance : 1.7740000e-04 ;
```

```
  resistance : 0.0035714 ;
  area : 0.0815905 ;
  slope : 5.0000000 ;
  fanout_length : 1 1.7460000 ;
  fanout_length : 2 3.9394000 ;
  fanout_length : 3 6.4626000 ;
  fanout_length : 4 9.2201000 ;
  fanout_length : 5 11.9123000 ;
  fanout_length : 6 14.8358000 ;
  fanout_length : 7 18.6155000 ;
  fanout_length : 8 22.6727000 ;
  fanout_length : 9 25.4842000 ;
  fanout_length : 11 27.0320000 ;
}

include NangateOpenCellLibrary_typical_conditional_25_0.6_CCS.indexes ;

do {
  TLAT_X1
  SDFF_X2
  SDFF_X1
  SDFFS_X2
  SDFFS_X1
  SDFFR_X2
  SDFFR_X1
  SDFFRS_X2
  SDFFRS_X1
  LOGIC1_X1
  LOGIC0_X1
  FILLCELL_X8
  FILLCELL_X4
  FILLCELL_X32
  FILLCELL_X2
```

FILLCELL_X16
FILLCELL_X1
DLL_X2
DLL_X1
DLH_X2
DLH_X1
DFF_X2
DFF_X1
DFFS_X2
DFFS_X1
DFFR_X2
DFFR_X1
DFFRS_X2
DFFRS_X1
CLKGATE_X8
CLKGATE_X4
CLKGATE_X2
CLKGATE_X1
CLKGATETST_X8
CLKGATETST_X4
CLKGATETST_X2
CLKGATETST_X1
AND2_X1
AND2_X2
AND2_X4
AND3_X1
AND3_X2
AND3_X4
AND4_X1
AND4_X2
AND4_X4
ANTENNA_X1
AOI211_X1

AOI211_X2
AOI211_X4
AOI21_X1
AOI21_X2
AOI21_X4
AOI221_X1
AOI221_X2
AOI221_X4
AOI222_X1
AOI222_X2
AOI222_X4
AOI22_X1
AOI22_X2
AOI22_X4
BUF_X1
BUF_X16
BUF_X2
BUF_X32
BUF_X4
BUF_X8
CLKBUF_X1
CLKBUF_X2
CLKBUF_X3
FA_X1
HA_X1
INV_X1
INV_X16
INV_X2
INV_X32
INV_X4
INV_X8
MUX2_X1
MUX2_X2

NAND2_X1
NAND2_X2
NAND2_X4
NAND3_X1
NAND3_X2
NAND3_X4
NAND4_X1
NAND4_X2
NAND4_X4
NOR2_X1
NOR2_X2
NOR2_X4
NOR3_X1
NOR3_X2
NOR3_X4
NOR4_X1
NOR4_X2
NOR4_X4
OAI211_X1
OAI211_X2
OAI211_X4
OAI21_X1
OAI21_X2
OAI21_X4
OAI221_X1
OAI221_X2
OAI222_X1
OAI22_X1
OAI33_X1
OR2_X1
OR2_X2
OR2_X4
OR3_X1

OR3_X2
OR3_X4
OR4_X1
OR4_X2
OR4_X4
TBUF_X1
TBUF_X16
TBUF_X2
TBUF_X4
TBUF_X8
TINV_X1
XNOR2_X2
XOR2_X1
XOR2_X2
}

In this thesis we characterized the library for the default temperature of 25°C and the voltage range is from 0.55 to 0.9 increased each time by 0.05

When we finish modifying the template file, we executed the final step with the command ncx -f run2 to perform characterization:

- using as cell netlists the post-layout parasitic parameters. The output library contained some cells that were not correct when it comes to the reference time. The problematic cells were the following:

1. OAI221_X4
2. OAI222_X2
3. OAI222_X4
4. OAI22_X2
5. OAI22_X4
6. XNOR2_X1

- Then, we assumed that the cause of the problem is in the connectivity of the parasitics with the transistors of the misbehaved cells. So we used the pre-layout parasitics netlists which do not contain R, C elements. During the process, we noticed that we had to remove from the DO list of the library template file the ANTENNA cell because it is incompatible with the Liberty NCX tool using the pre-layout netlits. In the output library the above six cells were corrected but many other cells were wrong (different reference time for the same input transition).

- Next try was to check the cell netlist schematics of the different drive strength variants in the same function and the compatibility between the cell netlist files and the related schematics that are provided by Nangate. We noticed that in almost every netlist, the polarity of some transistors was reversed. Thus we corrected the netlists of the cells that were wrong in the previous try and we run the characterization using the pre-layouts. Again the output library had many problematic cells.

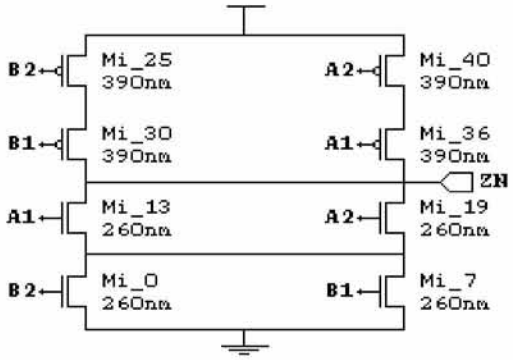


Figure 11: Schematic of cell OAI22_X2

The wrong netlist for the above cell that we took from Nangate is shown below (the red lines show the transistors whose drain and source are reversed):

.SUBCKT OAI22_X2 A1 A2 B1 B2 ZN VDD VSS

M_i_0 VSS B2 net_000 VSS NMOS_VTL

M_i_7 net_000 B1 VSS VSS NMOS_VTL

M_i_13 ZN A1 net_000 VSS NMOS_VTL

M_i_19 net_000 A2 ZN VSS NMOS_VTL

M_i_25 net_001 B2 VDD VDD PMOS_VTL

M_i_30 ZN B1 net_001 VDD PMOS_VTL

M_i_36 net_002 A1 ZN VDD PMOS_VTL

M_i_40 VDD A2 net_002 VDD PMOS_VTL

.ENDS

Now we can see the corrected netlist:

.SUBCKT OAI22_X2 A1 A2 B1 B2 ZN VDD VSS

M_i_0 net_000 B2 VSS VSS NMOS_VTL

M_i_7 net_000 B1 VSS VSS NMOS_VTL

M_i_13 ZN A1 net_000 VSS NMOS_VTL

M_i_19 ZN A2 net_000 VSS NMOS_VTL

M_i_25 net_001 B2 VDD VDD PMOS_VTL

M_i_30 ZN B1 net_001 VDD PMOS_VTL

M_i_36 ZN A1 net_002 VDD PMOS_VTL

M_i_40 net_002 A2 VDD VDD PMOS_VTL

.ENDS

- Finally, we corrected the post-layout netlists of the six cells mentioned in the first try and the result was a library with the least number of wrong cells among all tries. The problematic cells were the following:

    1. OAI222_X4
    2. OAI22_X2
    3. XNOR2_X1

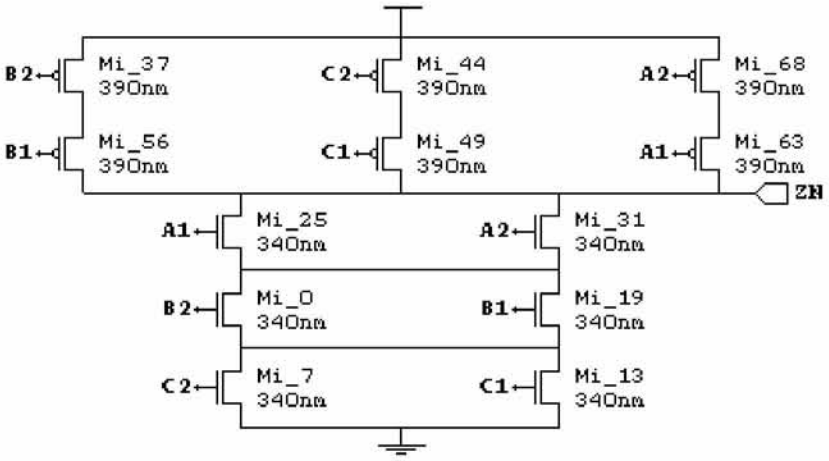


Figure 12: Schematic of cell OAI222_X2

The wrong netlist for the above cell that we took from Nangate is shown below (the red lines show the transistors whose drain and source are reversed):

.SUBCKT OAI222_X2 VSS VDD B2 C2 C1 ZN B1 A1 A2

M_M6 N_VDD_M0_d N_B2_M0_g N_5_M0_s VDD PMOS_VTL

M_M7 noxref_13 N_C2_M1_g N_VDD_M0_d VDD PMOS_VTL

M_M8 N_ZN_M2_d N_C1_M2_g noxref_13 VDD PMOS_VTL

M_M9 N_5_M3_d N_B1_M3_g N_ZN_M2_d VDD PMOS_VTL

M_M10 noxref_14 N_A1_M4_g N_ZN_M4_s VDD PMOS_VTL

M_M11 N_VDD_M5_d N_A2_M5_g noxref_14 VDD PMOS_VTL

M_M0 N_6_M6_d N_B2_M6_g N_4_M6_s VSS NMOS_VTL

M_M1 N_VSS_M7_d N_C2_M7_g N_6_M6_d VSS NMOS_VTL

M_M2 N_6_M8_d N_C1_M8_g N_VSS_M7_d VSS NMOS_VTL

M_M3 N_4_M9_d N_B1_M9_g N_6_M8_d VSS NMOS_VTL

M_M4 N_ZN_M10_d N_A1_M10_g N_4_M9_d VSS NMOS_VTL

M_M5 N_4_M11_d N_A2_M11_g N_ZN_M10_d VSS NMOS_VTL

.ENDS

Now we can see the corrected netlist:

```
.SUBCKT OAI222_X2 VSS VDD B2 C2 C1 ZN B1 A1 A2
M_M6 N_5_M0_s N_B2_M0_g N_VDD_M0_d VDD PMOS_VTL
M_M7 noxref_13 N_C2_M1_g N_VDD_M0_d VDD PMOS_VTL
M_M8 N_ZN_M2_d N_C1_M2_g noxref_13 VDD PMOS_VTL
M_M9 N_ZN_M2_d N_B1_M3_g N_5_M3_d VDD PMOS_VTL
M_M10 N_ZN_M4_s N_A1_M4_g noxref_14 VDD PMOS_VTL
M_M11 noxref_14 N_A2_M5_g N_VDD_M5_d VDD PMOS_VTL
M_M0 N_4_M6_s N_B2_M6_g N_6_M6_d VSS NMOS_VTL
M_M1 N_6_M6_d N_C2_M7_g N_VSS_M7_d VSS NMOS_VTL
M_M2 N_6_M8_d N_C1_M8_g N_VSS_M7_d VSS NMOS_VTL
M_M3 N_4_M9_d N_B1_M9_g N_6_M8_d VSS NMOS_VTL
M_M4 N_ZN_M10_d N_A1_M10_g N_4_M9_d VSS NMOS_VTL
M_M5 N_ZN_M10_d N_A2_M11_g N_4_M11_d VSS NMOS_VTL
.ENDS
```

After the Liberty™ NCX tool finishes, the ncx.log file is produced where we can see that defined cells were acquired sequentially. The input slews and output loads used during acquisition are defined in the cell templates. If there are any failures, they will be detailed at the end of the log file.

References

[1] Synopsys Inc., CCS Timing Technical White Paper version 2.0, 2006

[2] Synopsys Inc., CCS Power Technical White Paper version 3.0, 2006

[3] Synopsys Inc., Liberty™ NCX User Guide version D-2010.06, June 2010

[4] Low Power Cells Liberty Characterization with Liberty NCX, http://www.synopsys.com.cn/information/snug/2009/low-power-cells-liberty-characterization-with-liberty-ncx

[5] Nangate Open Cell Library, http://www.nangate.com/?page_id=22