liberty2json-python

GitHub Link : https://github.com/mirkat1206/liberty2json-python

2023/1/9

Shiuan-Yun Ding

Liberty User Guides and Reference Manual Suite Version 2017.06

What is Liberty library format?

Liberty library format is the most-widely-used industry standard format in EDA implementation.

A liberty library file contains information of library cells of a particular technology, including timing, power, area, noise, and etc.

General Syntax of Liberty library format

Group Statement	Attribute Statement	Define Statement
group_name (name) { *… statements …* *}*	Simple: *attr_name: attr_value;* Complex: *attr_name (param1, [param2, param3, …]);*	*define (attr_name, group_name, attr_type);*

What is JSON format?

JSON stands for JavaScript Object Notion.

JSON is a text format that is “self-describing” and easy to understand.

JSON is supported by many programming languages, including C/C++, C#, Java, Perl, Python, etc.

In Python, with `*import json*`, you can manipulate JSON format files easily.

Why do we need liberty2json?

In many cases, we might need information in Liberty files.

However, Liberty library format is irregular and hard to process.

In contrast, JSON format is more regular and easy to manipulate.

liberty2json-python

ASIC Design 之初--就先從江Liberty轉成JSON開始吧

https://github.com/erihsu/liberty2json

https://github.com/iLeonSun/Liberty

https://github.com/liyanqing1987/libertyParser

Compared to other existing liberty2json projects, this project has some special features:

1. This project is written in Python.
2. This project is purely based on the general syntax of Liberty library format.
    - Unlike the other projects, this project does not predefine any names (group/attribute/define) nor presume any input orders.
    - Hence, this project can deal with any unexpected names or orders.
3. The code is very clean and elegant.
    - Less than 300 lines of code in *liberty2json.py*, including comments.
4. Another *liberty.py* is also created for parsing the resulted JSON file.
    - Basic methods are created for common needs (ex: get_attributes/get_cells/get_pins)
    - Additional methods can be easily added according to users' needs.

liberty2json-python

Liberty format

```
/*******************************************************************
****                                                           ****
****  This file is created for liberty2json-python project. All ****
****  data is fake and cannot be used for fabrication.         ****
****                                                           ****
*******************************************************************/

library ("saed012rvt_ff34p56v789c") { /*comment*/
    define(driver_model,library,string);
    define(def_sim_opt,library,string);
    define(simulator,library,string);
    define(always_on,cell,string);
    define(is_decap_cell,cell,string);
    define(clk_width,timing,string);
    technology("cmos");
    delay_model : "table_lookup";
    date : "[1999 DECEMBER 6]";
    revision : "1.0000000";
    time_unit : "1ns";
    leakage_power_unit : "1pW";
    voltage_unit : "1V";
    pulling_resistance_unit : "1kohm";
    current_unit : "1uA";
    capacitive_load_unit(1.000000, \
    "ff");
    default_leakage_power_density : 0.000000;
    input_threshold_pct_rise : 87.000000; /*comment*/
```

JSON format

```
{
  "library": {
    "saed012rvt_ff34p56v789c": {
      "technology": "cmos",
      "delay_model": "table_lookup",
      "date": "[1999 DECEMBER 6]",
      "revision": "1.0000000",
      "time_unit": "1ns",
      "leakage_power_unit": "1pW",
      "voltage_unit": "1V",
      "pulling_resistance_unit": "1kohm",
      "current_unit": "1uA",
      "capacitive_load_unit": "1.000000,  ff",
      "default_leakage_power_density": "0.000000",
      "input_threshold_pct_rise": "87.000000",
      "output_threshold_pct_rise": "87.000000",
      "input_threshold_pct_fall": "87.000000",
      "output_threshold_pct_fall": "87.000000",
      "slew_lower_threshold_pct_rise": "69.000000",
      "slew_upper_threshold_pct_rise": "96.000000",
      "slew_lower_threshold_pct_fall": "69.000000",
      "slew_upper_threshold_pct_fall": "96.000000",
      "slew_derate_from_library": "1.000000",
      "default_inout_pin_cap": "0.000000",
```

How to use?

liberty2json.py & main.py

```
$ python3 src/main.py
usage: main.py [-h] [--libfile LIBFILE] [--libdir LIBDIR] [--jsondir JSONDIR]

Turn liberty format file(s) into json format file(s).

optional arguments:
  -h, --help         show this help message and exit
  --libfile LIBFILE  filepath of a liberty format file
  --libdir LIBDIR    dirpath of a directory containing liberty format file(s)
  --jsondir JSONDIR  dirpath for output json file(s)
```

How to use?

liberty.py

```python
import json
from liberty import liberty

lib = liberty(filepath='./test.json')
print('\n----------------------------------------\n')

print(lib.name)
print(lib.list_attributes())
print(lib.get_attribute('technology'))
print(lib.get_attributes(['technology', 'delay_model']))
print(lib.list_cells())
print(lib.get_cells())
print('\n----------------------------------------\n')

cel = lib.get_cell(lib.list_cells()[0])
print(cel.name)
print(cel.list_attributes())
print(cel.get_attribute('cell_footprint'))
print(cel.get_attributes(['cell_footprint', 'area']))
print(cel.list_pins())
print(cel.get_pins())
print(cel.list_input_pins())
print(cel.get_input_pins())
print(cel.list_output_pins())
print(cel.get_output_pins())
print(cel.list_inout_pins())
print('\n----------------------------------------\n')

pinn = cel.get_pin(cel.list_output_pins()[0])
print(pinn.name)
print(pinn.list_attributes())
```

```
----------------------------------------

saed012rvt_ff34p56v789c
['technology', 'delay_model', 'date', 'revision', 'time_unit', 'le
akage_power_unit', 'voltage_unit', 'pulling_resistance_unit', 'cur
rent_unit', 'capacitive_load_unit', 'default_leakage_power_density
', 'input_threshold_pct_rise', 'output_threshold_pct_rise', 'input
_threshold_pct_fall', 'output_threshold_pct_fall', 'slew_lower_thr
eshold_pct_rise', 'slew_upper_threshold_pct_rise', 'slew_lower_thr
eshold_pct_fall', 'slew_upper_threshold_pct_fall', 'slew_derate_fr
om_library', 'default_inout_pin_cap', 'default_cell_leakage_power'
, 'default_input_pin_cap', 'default_output_pin_cap', 'default_fano
ut_load', 'library_features', 'driver_model', 'nom_voltage', 'nom_
temperature', 'nom_process', 'default_wire_load_capacitance', 'def
ault_wire_load_resistance', 'default_wire_load_area', 'default_wir
e_load_mode', 'default_threshold_voltage_group', 'def_sim_opt', 's
imulator', 'voltage_map', 'output_voltage', 'wire_load', 'wire_loa
d_selection', 'power_lut_template', 'operating_conditions', 'lu_ta
ble_template', 'normalized_driver_waveform', 'cell']
cmos
{'technology': 'cmos', 'delay_model': 'table_lookup'}
['AND2X1', 'AND2X2']
{'AND2X1': <liberty.cell object at 0x7f6519a0a8e0>, 'AND2X2': <lib
erty.cell object at 0x7f6519a0a640>}

----------------------------------------

AND2X1
['cell_footprint', 'area', 'cell_leakage_power', 'driver_waveform_
rise', 'driver_waveform_fall', 'leakage_power', 'pg_pin', 'pin']
AND2
{'cell_footprint': 'AND2', 'area': '2.123456'}
['A1', 'A2', 'Y']
{'A1': <liberty.pin object at 0x7f6519a0a8e0>, 'A2': <liberty.pin
object at 0x7f6519a0a5e0>, 'Y': <liberty.pin object at 0x7f6519a0a
580>}
['A1', 'A2']
{'A1': <liberty.pin object at 0x7f6519a0a5e0>, 'A2': <liberty.pin
object at 0x7f6519a0a8e0>}
['Y']
{'A1': <liberty.pin object at 0x7f6519a0a5e0>, 'A2': <liberty.pin
object at 0x7f6519a0a580>}
[]

----------------------------------------

Y
```

Fun Fact

All docstring is generated by ChatGPT.

I only modified few places.

https://chat.openai.com/chat