Library characterizer for open-source VLSI design

Shinichi Nishizawa, Waseda Univ.

Toru Nakura, Fukuoka Univ.

Outline

- Background
- Proposed characterizer
  - Characterization flow
  - Characterize combinational cells
  - Characterize sequential cells
- Experimental results
- Conclusion and future work

Open source VLSI design

- Design chips using open source EDAs
  - Google + Skywater (United States)
  - MakeLSI Project (Japan)

☺ : Anyone can join this project, design their own circuits

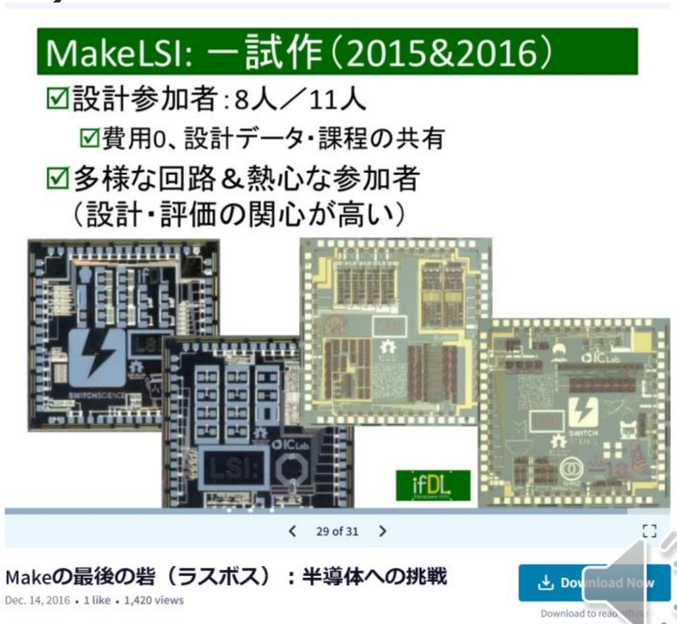
☹ : Some important tools are not ready

Google + skywater TECHNOLOGY

FOSS 130nm Production PDK
github.com/google/skywater-pdk

LSI:

MakeLSI Project (Japan)
Organized by Prof. Akita,
Kanazawa Univ. ⇨



MakeLSI: 一試作（2015&2016）

☑設計参加者：8人／11人

☑費用0、設計データ・課程の共有

☑多様な回路＆熱心な参加者
（設計・評価の関心が高い）

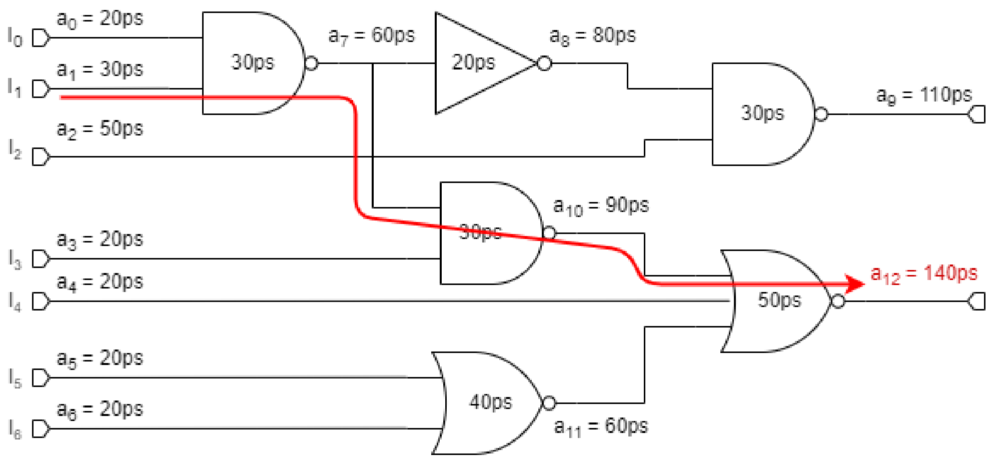
Makeの最後の砦（ラスボス）：半導体への挑戦

Dec. 14, 2016 • 1 like • 1,420 views

https://www.slideshare.net/junichiakita9/make-70153385

Digital circuit design and STA

- Static Timing Analysis (STA): estimation of path delay
  - Calculate max/min path delay integrating the cells delay
    - Max delay of ith node: : $a_i = \max_{j \in \text{fanin}(i)}\{a_j\} + t_{\text{pd},i}$
  - Need both timing information (.lib) calculation engine (STA)
    - STA: OpenSTA (Open-ROAD), sta (yosis)
    - .lib: (Need characterizer)



Related works

- (Based on our survey) no good candidate is available
  1. Google-Skywater-PDK：brank for characterizer
     - Timing library in JSON (why?)
  2. pharosc[8]
     - From website *The Art of Standard Cell Library Design*
     - Binary file for each logic (?), WinSpice3
  3. AutoLibGen[9] in *Intl. Conf. on Electronic Design*
     - Binary file for each logic (?), HSPICE
  4. LiChEn[10] in *Euromicro Conf. on Digital System Design*
     - Analyze logic function, C++ source in GitHub，Spector
- DFF：2 support non-set/reset FF，3&4 not support

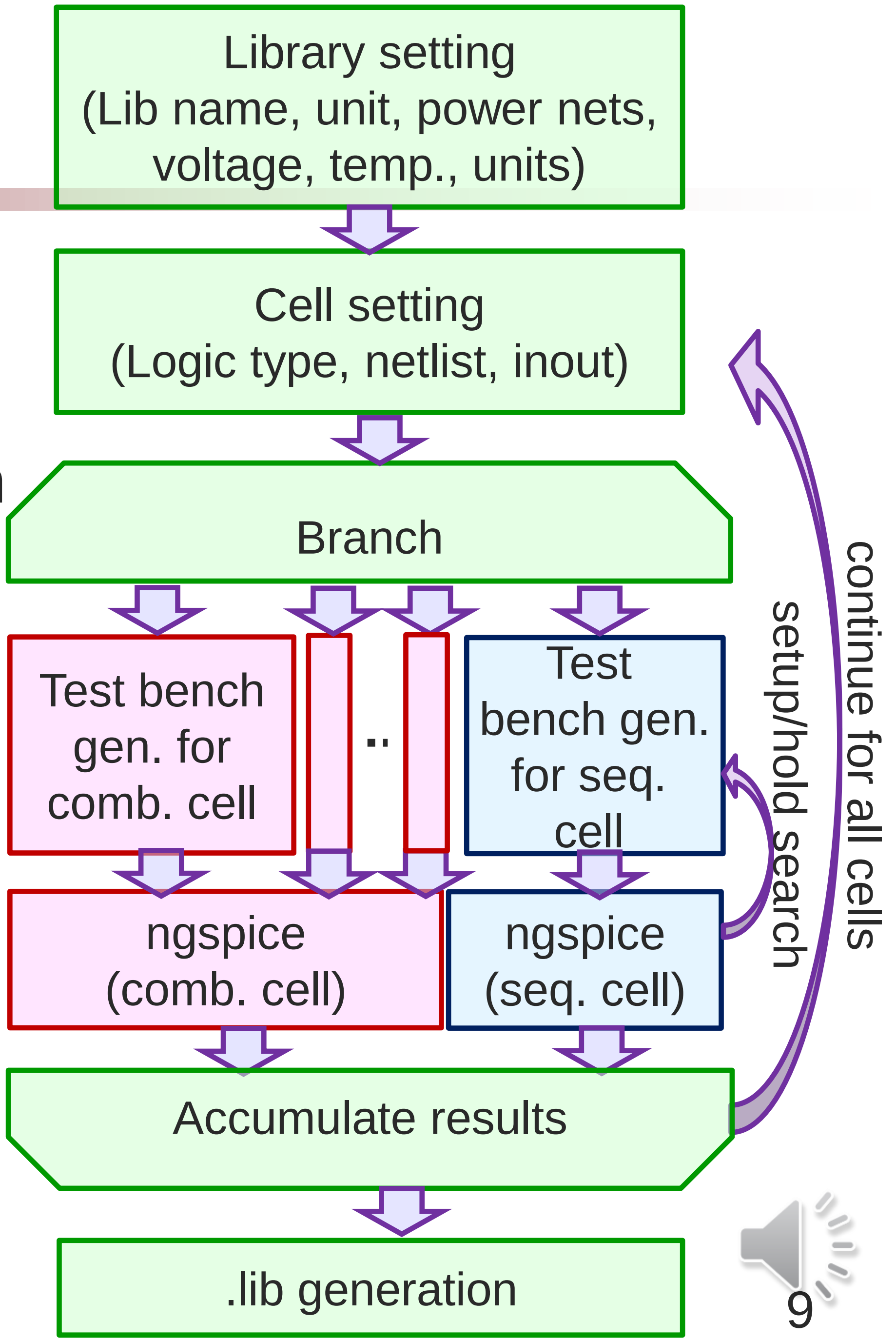
[8] G. Bronstein, et al.,, “Asic standard cell library design by graham petley.,” 1991.
[9] I.K. Rachit and M.S. Bhat, “AutoLibGen: An open source tool for standard cell library characterization at 65nm technology,” Intl. Conf. on Electronic Design, 2008.
[10] M.T. Moreira, et al., “LiChEn: Automated electrical characterization of asynchronous standard cell libraries,” Euromicro Conf, on Digital System Design, pp.933–940, 2013.

Proposed characterizer

- Open source characterizer
  - Language: Python3
  - Simulator: ngspice, hspice
- Advantage
  - Characterize both combinational and Flip-Flops
    - Support Flip-Flops w/ async. set/reset
  - Easy to add function
    - Two analysis engine: for combinational and sequential
      - Do not prepare engines for each logic function
      - Use truth table to handle different logic functions

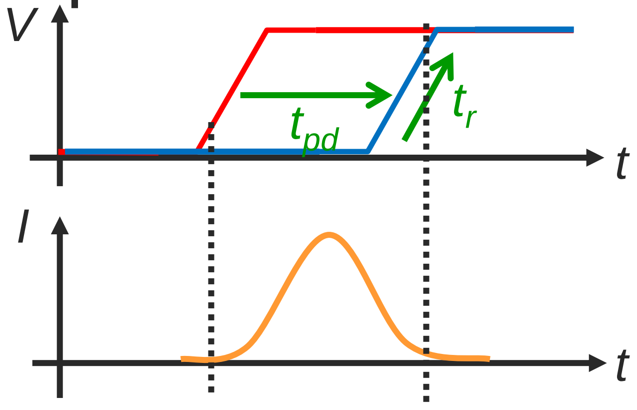
Operation flow

- Setup library and cells
- Branch to comb. or seq.
  - Generate test bench based on the target logic function
  - Launch simulator
- Seq. cell need iteration
  - Find min. delay in setup/hold search



Characterize: combinational cell

- Propagation delay: input 50% to output 50%
- Transition delay: output 20% to output 80%
- Dynamic power*1 : integrate current from input start 1% to output end 99%
- Static power : integ. current at the beginning of sim*2
- Set simulation end time, time step
  - Set by designer, or use auto set

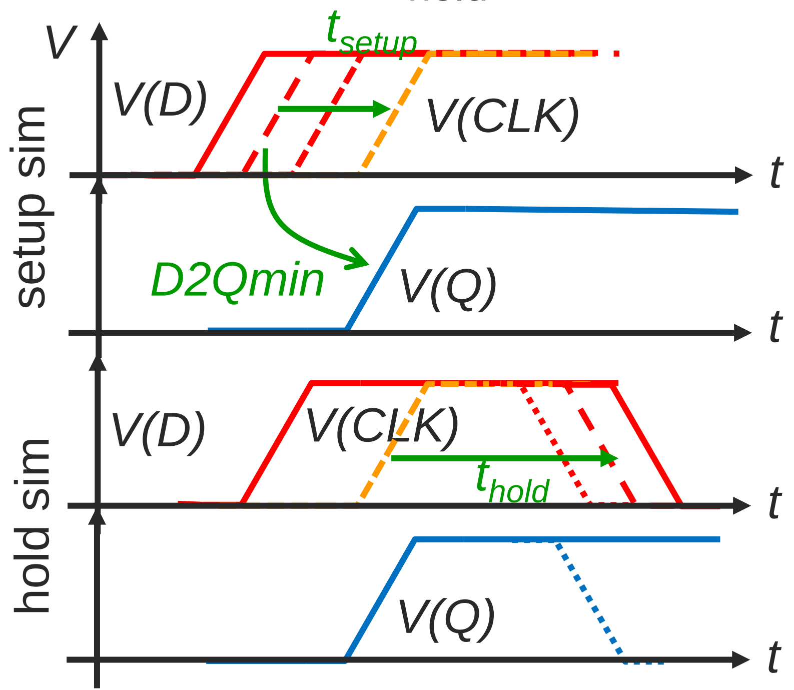


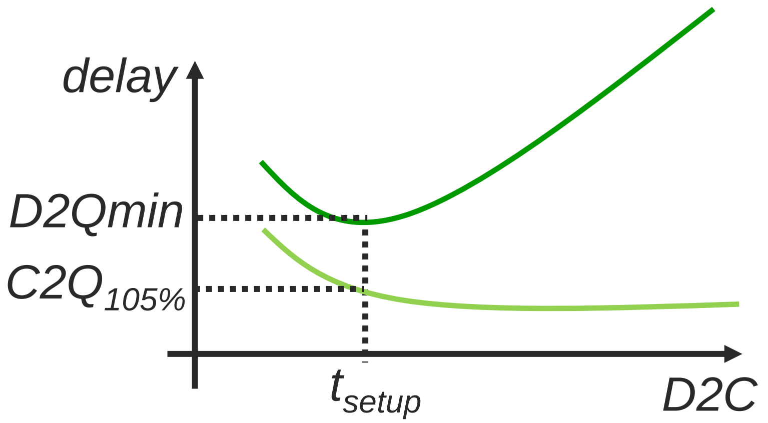
$$P_d = \int_{t@V_i=1\%}^{t@V_o=99\%} I\,dt \times V_{dd}$$

* Parameters can be changed by designer. Initial parameters comes from Liberty User Guide V1
*1: This should be in energy
*2: Don't meas. input dependency

Characterize: sequential cell delay

- C2Q delay, t_{setup}, t_{hold} : calc. by min. D2Q delay
  - Change D2C, find min. D2Q
    - D2C= t_{setup}, C2Q = D2Q – D2C
  - Set D2C= t_{setup}, change C2D and find min C2D
    - Min. C2D = t_{hold}

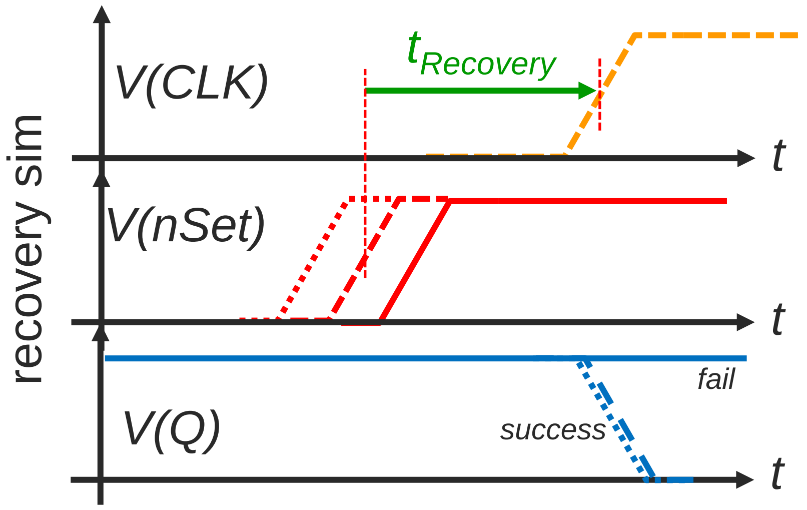


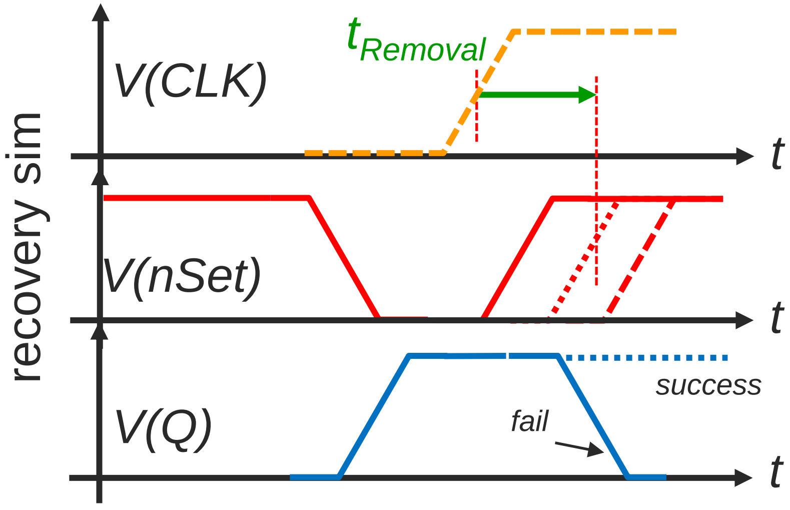


Two def. for setup
1. D2C when C2Q increase 3~5%
2. D2C when D2Q is minimum

Characterize: recovery removal

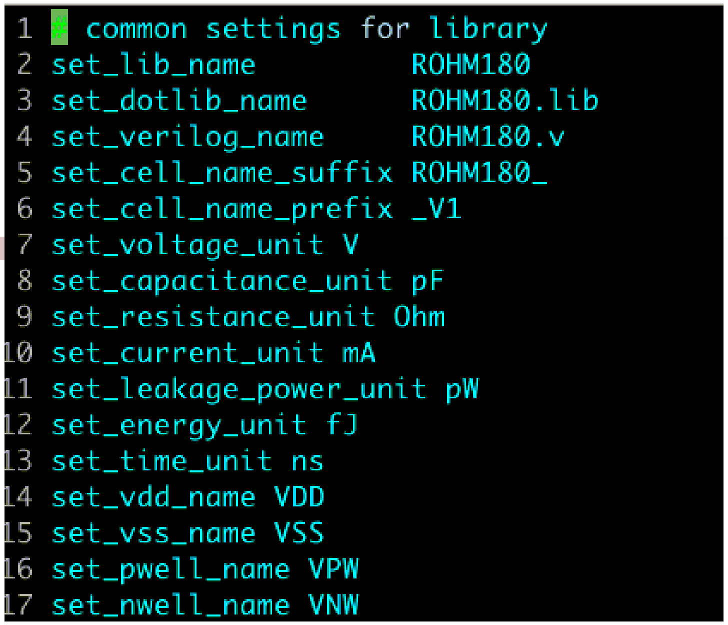
- Timing for asynchronous set/reset
  - Important for asynchronous circuit design
  - Recovery: operation correctness after the release of set/reset
    - min. time from set/reset to inactivate state to clock
  - Removal: operation correctness of set/reset during the clock
    - Min. time clock at active state to set/reset in inactive state



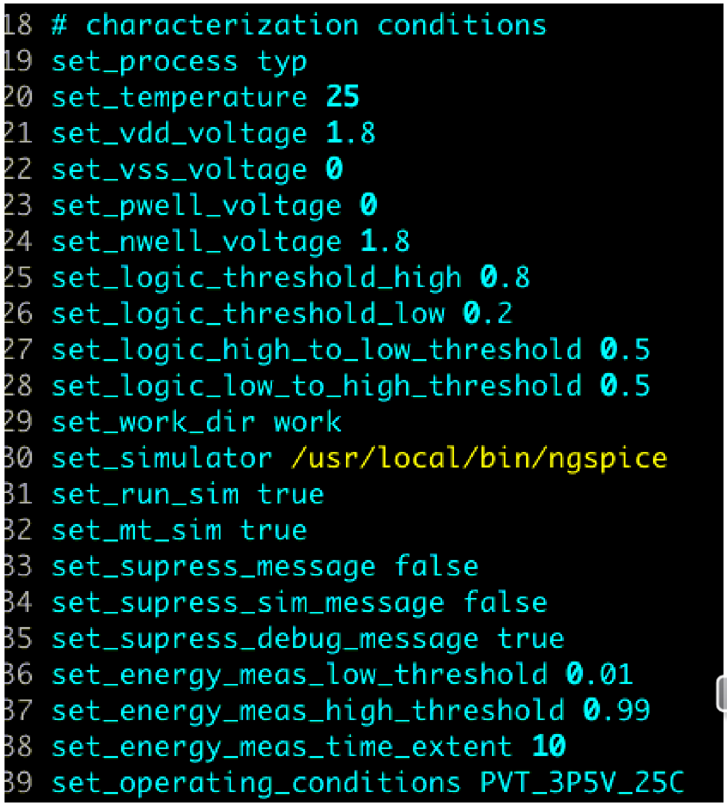


Library setting

- Common setting for lib.
  - Library name
  - Prefix, suffix of cells
  - Units (volt, current, cap.)
  - Power name
  - Temperature
  - Voltage
  - Logical threshold
  - High/low threshold
  - Operation directory
  - Simulator
  - …



```
# common settings for library
set_lib_name          ROHM180
set_dotlib_name       ROHM180.lib
set_verilog_name      ROHM180.v
set_cell_name_suffix ROHM180_
set_cell_name_prefix _V1
set_voltage_unit V
set_capacitance_unit pF
set_resistance_unit Ohm
set_current_unit mA
set_leakage_power_unit pW
set_energy_unit fJ
set_time_unit ns
set_vdd_name VDD
set_vss_name VSS
set_pwell_name VPW
set_nwell_name VNW
```

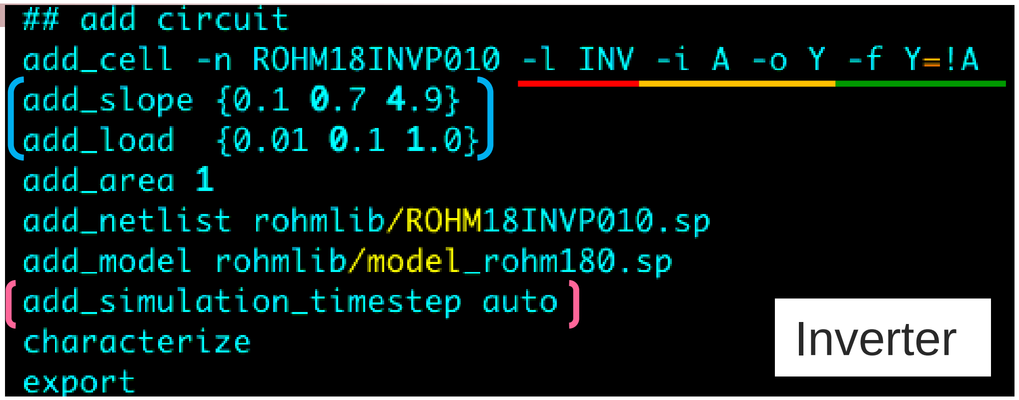


```
# characterization conditions
set_process typ
set_temperature 25
set_vdd_voltage 1.8
set_vss_voltage 0
set_pwell_voltage 0
set_nwell_voltage 1.8
set_logic_threshold_high 0.8
set_logic_threshold_low 0.2
set_logic_high_to_low_threshold 0.5
set_logic_low_to_high_threshold 0.5
set_work_dir work
set_simulator /usr/local/bin/ngspice
set_run_sim true
set_mt_sim true
set_supress_message false
set_supress_sim_message false
set_supress_debug_message true
set_energy_meas_low_threshold 0.01
set_energy_meas_high_threshold 0.99
set_energy_meas_time_extent 10
set_operating_conditions PVT_3P5V_25C
```

Setting for each cell

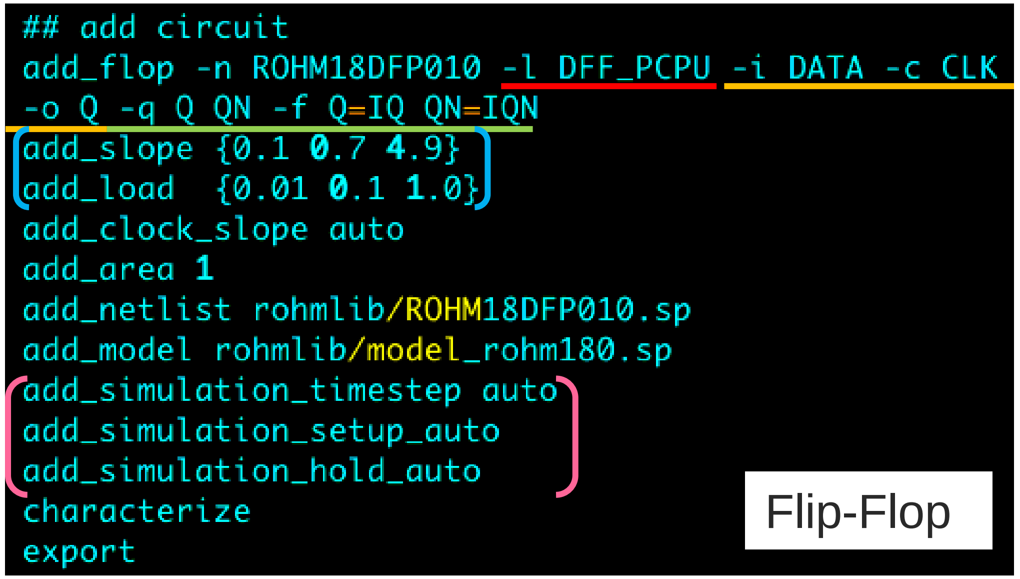
- Characterize cond.
  - Add cell
  - Input slope (in array)
  - Output load (in array)
  - Netlist
  - Timestep*. sim. end*
- Flip-Flop need
  - Clock slope*
  - Setup unit time*
  - Hold unit time*

* Parameters can use auto set



```
## add circuit
add_cell -n ROHM18INVP010 -l INV -i A -o Y -f Y=!A
add_slope {0.1 0.7 4.9}
add_load  {0.01 0.1 1.0}
add_area 1
add_netlist rohmlib/ROHM18INVP010.sp
add_model rohmlib/model_rohm180.sp
add_simulation_timestep auto
characterize
export
```

Inverter



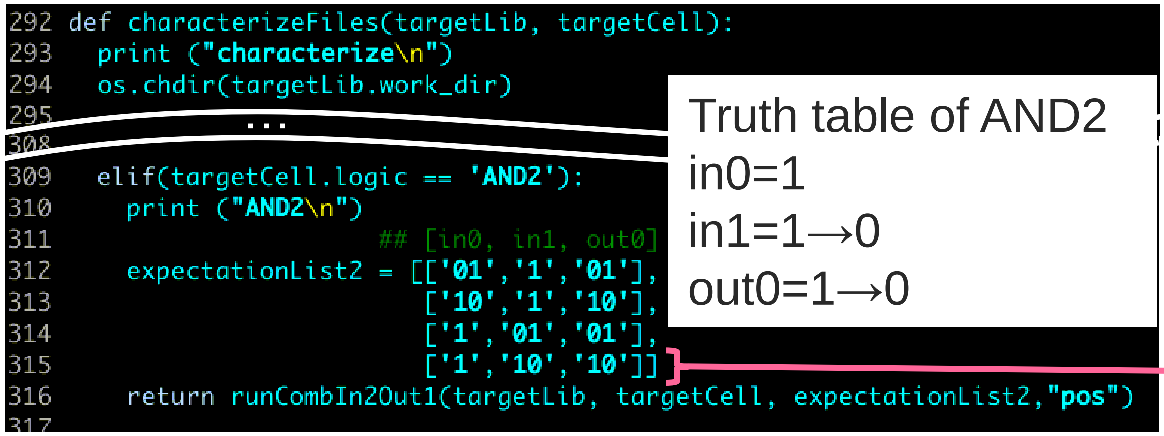
```
## add circuit
add_flop -n ROHM18DFP010 -l DFF_PCPU -i DATA -c CLK
-o Q -q Q QN -f Q=IQ QN=IQN
add_slope {0.1 0.7 4.9}
add_load  {0.01 0.1 1.0}
add_clock_slope auto
add_area 1
add_netlist rohmlib/ROHM18DFP010.sp
add_model rohmlib/model_rohm180.sp
add_simulation_timestep auto
add_simulation_setup_auto
add_simulation_hold_auto
characterize
export
```

Flip-Flop

Command example: logic function, in/out pins, storage, logic expression, slew/cap index, simulation time step

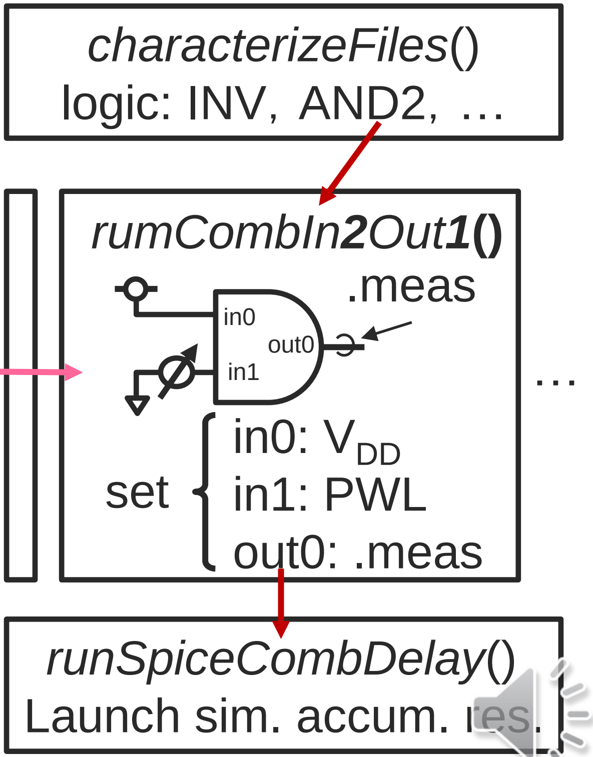
Netlist gen. and simulation (comb. cell)

- *characterizeFiles*() : def. of logic func., its truth table
- *runCombIn**n**Out**m***() : set input/output pin setting
- *runSpiceCombDelay*() : generate netlist, run spice



```
def characterizeFiles(targetLib, targetCell):
  print ("characterize\n")
  os.chdir(targetLib.work_dir)
            ...
  elif(targetCell.logic == 'AND2'):
    print ("AND2\n")
                     ## [in0, in1, out0]
    expectationList2 = [['01','1','01'],
                        ['10','1','10'],
                        ['1','01','01'],
                        ['1','10','10']]
    return runCombIn2Out1(targetLib, targetCell, expectationList2,"pos")
```

Truth table of AND2
in0=1
in1=1→0
out0=1→0

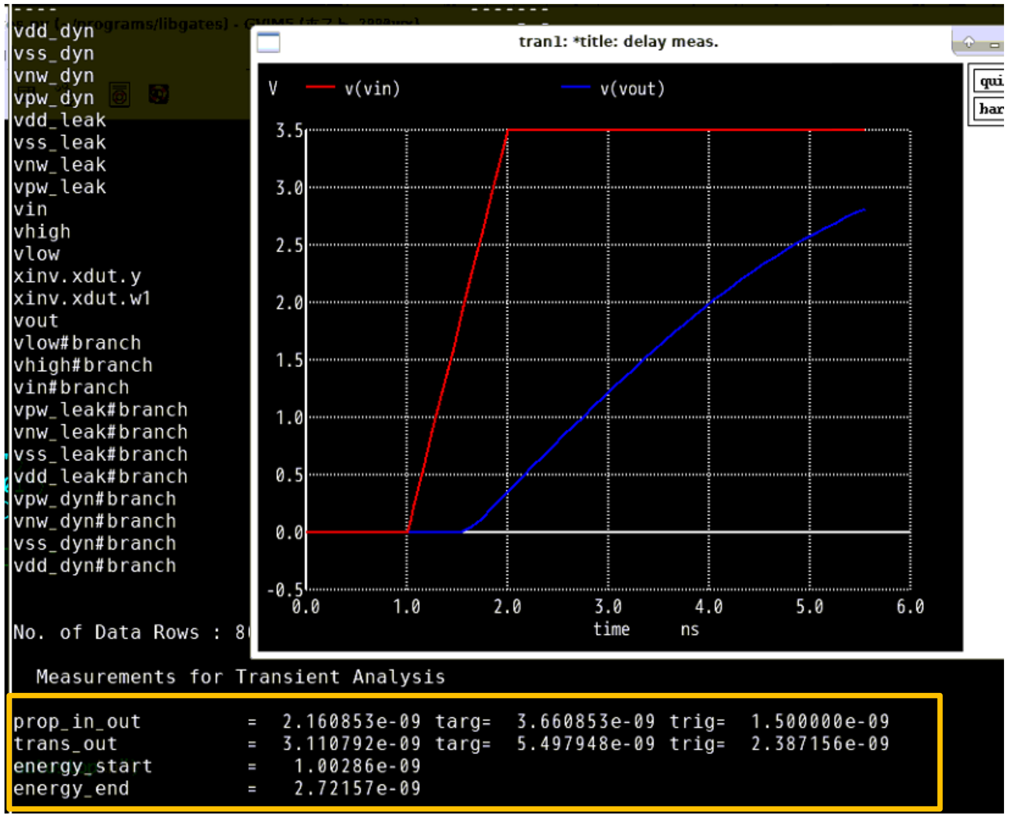


Use same function for netlist gen., spice run, analysis (*runSpiceCombDelay*())

- Pin order is analyzed and connect proper voltage sources and spice measure statements

Simulation result (combinational)

- ngspice example at in0(1) , in1(0→1) and out0(0→1)
  - (Waveform for debug)
- Read result of .measure statements, create timing and power LUT

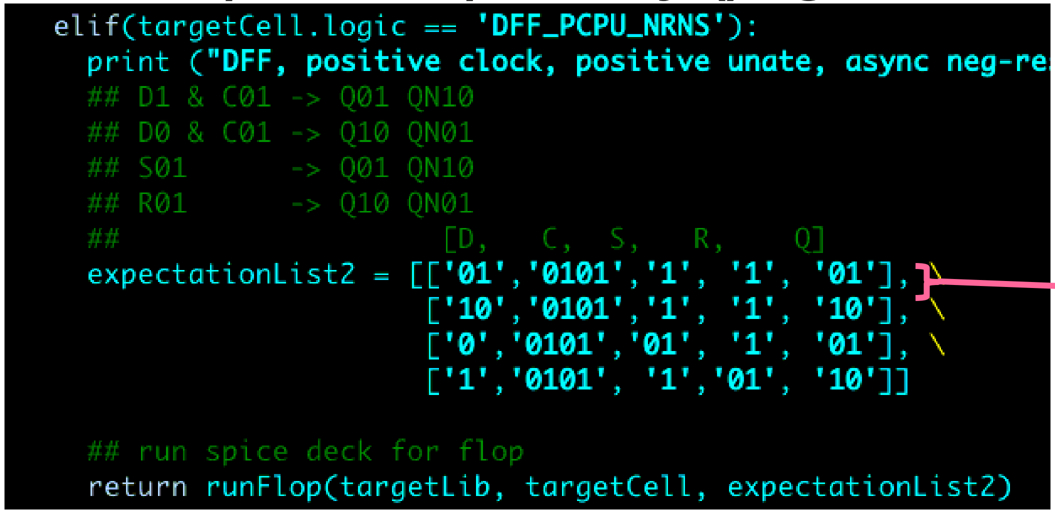


```
vdd_dyn
vss_dyn
vnw_dyn
vpw_dyn
vdd_leak
vss_leak
vnw_leak
vpw_leak
vin
vhigh
vlow
xinv.xdut.y
xinv.xdut.w1
vout
vlow#branch
vhigh#branch
vin#branch
vpw_leak#branch
vnw_leak#branch
vss_leak#branch
vdd_leak#branch
vpw_dyn#branch
vnw_dyn#branch
vss_dyn#branch
vdd_dyn#branch

No. of Data Rows : 8

 Measurements for Transient Analysis

prop_in_out          =  2.160853e-09 targ=  3.660853e-09 trig=  1.500000e-09
trans_out            =  3.110792e-09 targ=  5.497948e-09 trig=  2.387156e-09
energy_start         =   1.00286e-09
energy_end           =   2.72157e-09
```

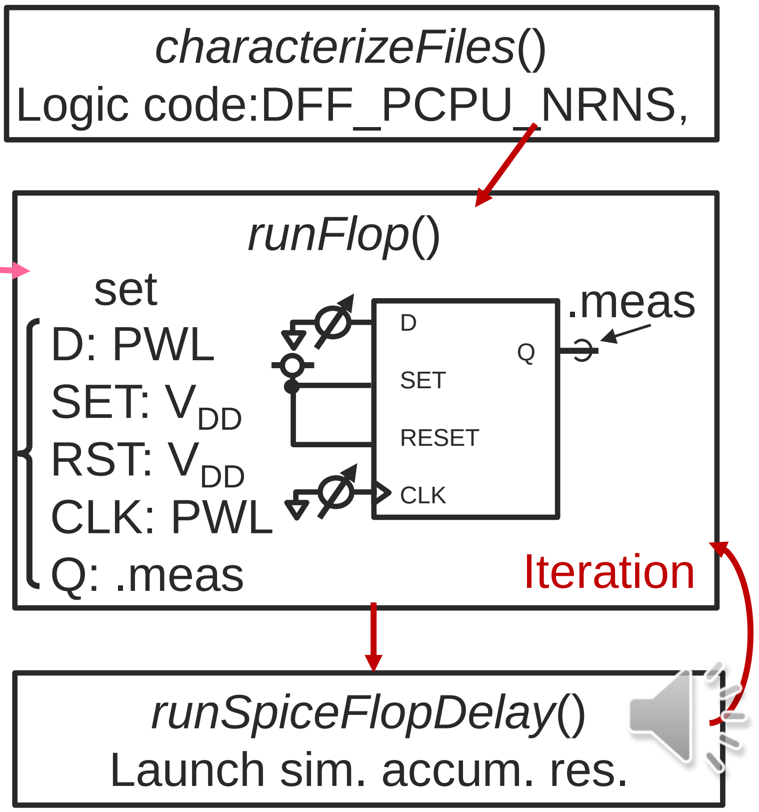
Result of .measure statement

Netlist gen. and simulation (seq. cell)

- *characterizeFiles*() : def. of logic func., its truth table
- *runFlop*() : set input/output pin setting
- *runSpiceFlopDelay*() : generate netlist, run spice



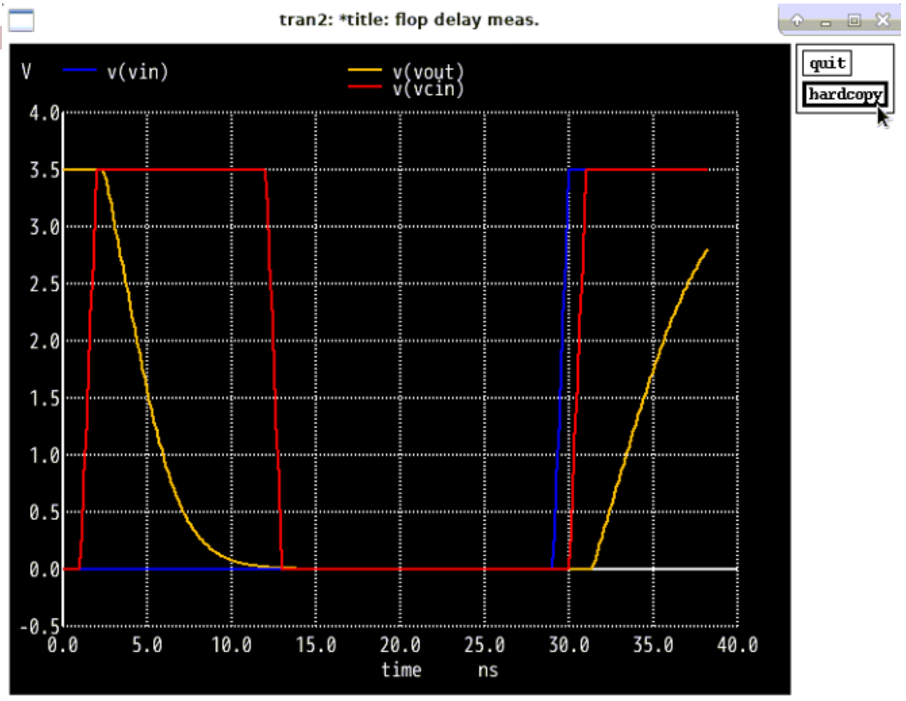
```
  elif(targetCell.logic == 'DFF_PCPU_NRNS'):
    print ("DFF, positive clock, positive unate, async neg-re
    ## D1 & C01 -> Q01 QN10
    ## D0 & C01 -> Q10 QN01
    ## S01      -> Q01 QN10
    ## R01      -> Q10 QN01
    ##                  [D,    C,   S,    R,     Q]
    expectationList2 = [['01','0101','1', '1', '01'],\
                         ['10','0101','1', '1', '10'],\
                         ['0','0101','01', '1', '01'], \
                         ['1','0101', '1','01', '10']]

    ## run spice deck for flop
    return runFlop(targetLib, targetCell, expectationList2)
```



Truth table of code: DFF_PCPU_NRNS
(pos. clk, pos. unate, neg. rst, neg. set)
D=0→1, SET=1, RST=1
CLK= 0→1 → 0→1
Q=0→1

Simulation result (sequential)

- ngspice example at D(0→1) , CLK(0→1 → 0→1), and Q(X→0→1)
  - 1st CLK: set initial value, 2nd clock: measure D2Q delay
  - (Waveform for debug)
- Read result of .measure statements, create timing and power LUT



Registered logic family

- Support simple logic functions and several Flip-Flops

Family	Logic function
Inv/Buf	Inverter, Buffer
NAND	NAND2, NAND3, NAND4
NOR	NOR2, NOR3, NOR4
AND	AND2, AND3, AND4
OR	OR2, OR3, OR4
And-Or-Inv.	AOI21, AOI22
Or-And-Inv.	OAI21, OAI22
Exclusive	XOR2, XNOR2
Selector	SEL2

DFF code	clk	porality*1	set	rst
DFF_PCPU *2	pos.	pos.		
DFF_PCNU	pos.	neg.		
DFF_NCPU	pos.	neg.		
DFF_NCNU	neg.	neg.		
DFF_PCPU_NR	pos.	pos.		neg.
DFF_PCPU_NRNS	pos.	pos.	neg.	neg.

*1: Pos.: in/out are same direction (H/H,L/L). neg. in/out are opposite (H/L,L/H)

*2: D-Flip-Flop w/ pos. clock edge, positive porality

Cent OS 7.8
Ryzen 2990wx 3GHz 32core
MEM 96GB，SSD 3TB

Evaluation setup

- Use commercial 180nm for evaluation

	Proposed characterizer		SilconSmart
Simulator	ngspice	hspice	
#parallelism	1	1	32
Condition	Typical (TT, 1.8V, 25℃)		
Input slew	0.01ns, 0.1ns, 1.0ns		
Output load	0.1pF, 0.7pF, 4.9pF		
High/low threshold	1.44V, 0.36V(80%, 20%)		
Logic threshold	0.9V (50%)		
Device under test	Inverter, DFF w/ pos. edge clock		
Runtime	147min.	65 min. (2.3x)	41 sec. (215x)

- Need large runtime: low parallelism, poor search algorithm, separate timing and power sim. (ngspice do not support nesting of .meas)

Result (comb. cell)

- No difference in simulator (ngspice, HSPICE)
- Characterizers show some different result (HSPICE)
  - Propagation delay and transient delay are almost same (Max error 0.5%, 24%)
  - Intl. energy: large difference at large slew case (Max 418%)
  - Leak power : Matches
  - Input cap. : 10% difference

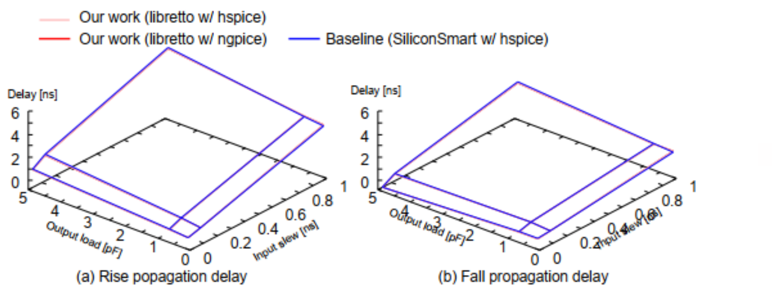


Fig. 6: Propagation delay of Inverter.

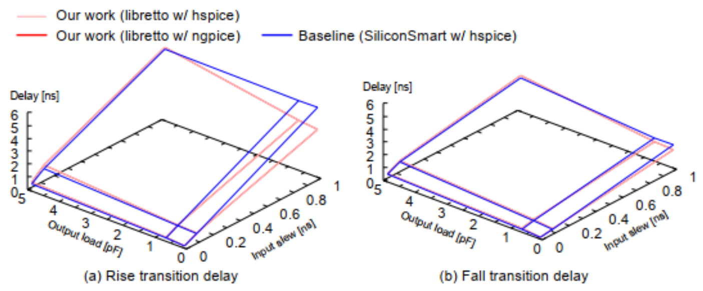


Fig. 7: Transition delay of Inverter.

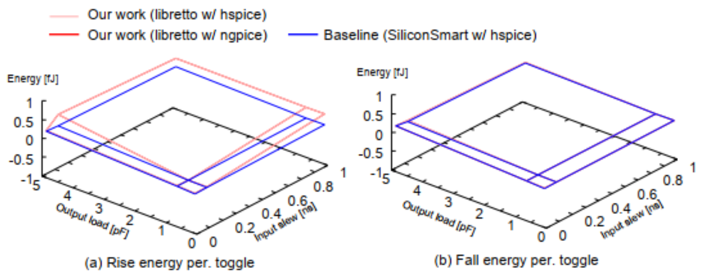


Fig. 8: Internal energy of Inverter.

TABLE II: Capacitance and leakage power of Inverter.

Characterizer	**libretto**		SiliconSmart
Simulator	ngspice	hspice	
Leakage power [pW]	9.862	9.862	9.862
Input capacitance [fF]	4.120	4.063	4.599

Result (seq. cell)

- Different result by characterizer
  - C2Q delay：1125%, 2407% pessimistic
  - Cause：Setup/hold interdependence
    - Tight setup/hold worse other performance (C2Q, hold/setup)
    - Method we use: find min. setup then measure others
    - SiliconSmart：balance setup/hold
    - Need to balance setup/hold

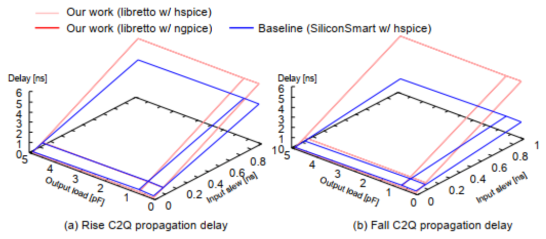


Fig. 9: C2Q propagation delay of Flip-Flop.

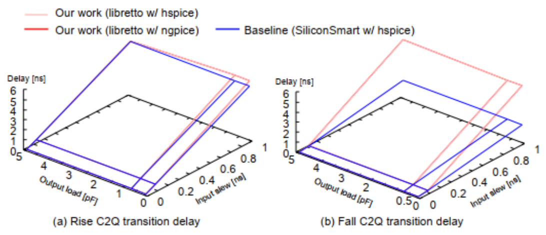


Fig. 10: C2Q transition delay of Flip-Flop.

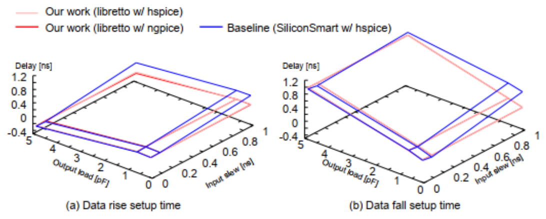


Fig. 11: Setup time of Flip-Flop.

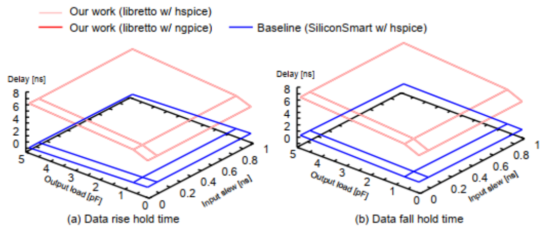


Fig. 12: Hold time of Flip-Flop.

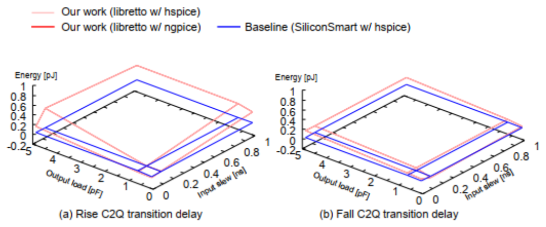


Fig. 13: Energy consumption of Flip-Flop.

Conclusion

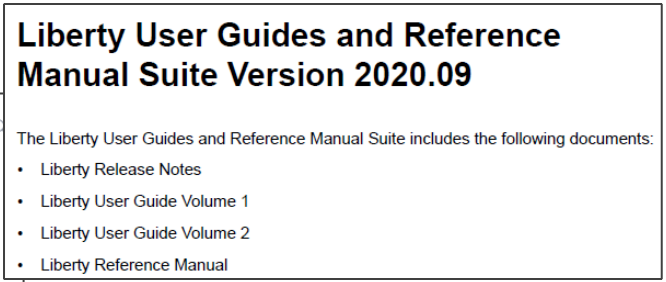
- Full open characterizer
  - Generate timing/power library as .lib
  - Used for timing analysis (simulation, STA)
- Support both combinational and sequential cells
  - Combinational cell: register the function as truth table
  - Sequential cell: support asynchronous set/reset
- Evaluate delay, energy, and performance
  - Slow processing speed (1/215), need to use parallelism
  - Combinational cell: delay and energy are close
  - Sequential cell: large gap
- Checked .lib by LibraryCompiler Synopsys

Issues and future work

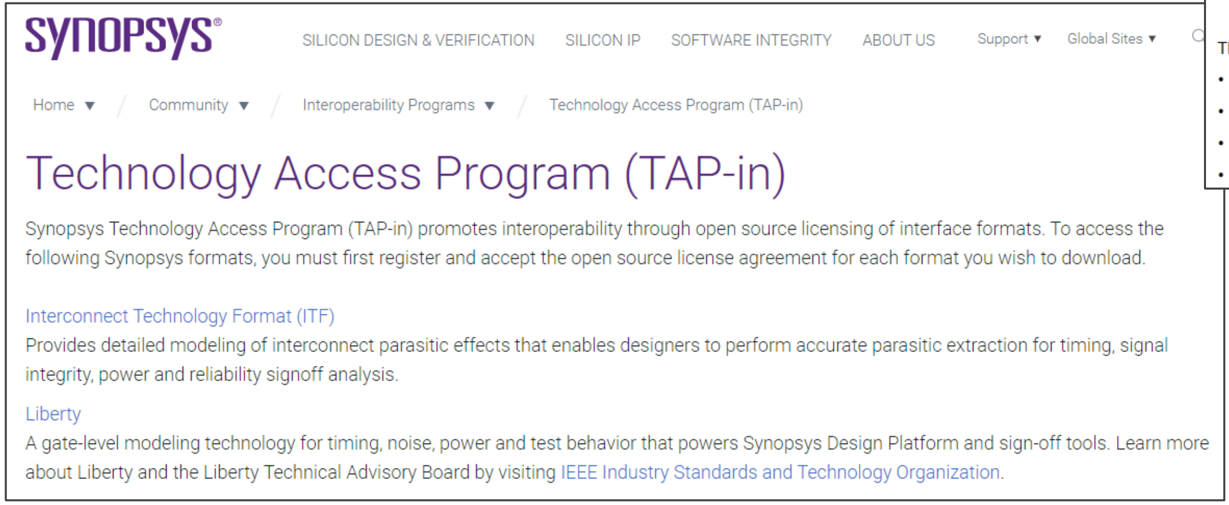
- Support more logic cells
  - Combinational cells: multi-output cell support
  - Sequential cells: latch, scan support
- Performance improvement
  - Use parallelism
  - Algorithm improvement for accuracy, speedup
- Code: https://github.com/snishizawa/libretto
- Contact: nishizawa@aoni.waseda.jp
  - Acknowledge: VDEC, Univ. of Tokyo, Nihon Synopsys G.K. Fund. from Logic Research and Fukuoka Univ.

Appendix: right of Liberty format

- Liberty format is open
  - Can access liberty user guide


Liberty User Guides and Reference Manual Suite Version 2020.09

The Liberty User Guides and Reference Manual Suite includes the following documents:
- Liberty Release Notes
- Liberty User Guide Volume 1
- Liberty User Guide Volume 2
- Liberty Reference Manual


SYNOPSYS® SILICON DESIGN & VERIFICATION SILICON IP SOFTWARE INTEGRITY ABOUT US Support ▾ Global Sites ▾

Home ▾ / Community ▾ / Interoperability Programs ▾ / Technology Access Program (TAP-in)

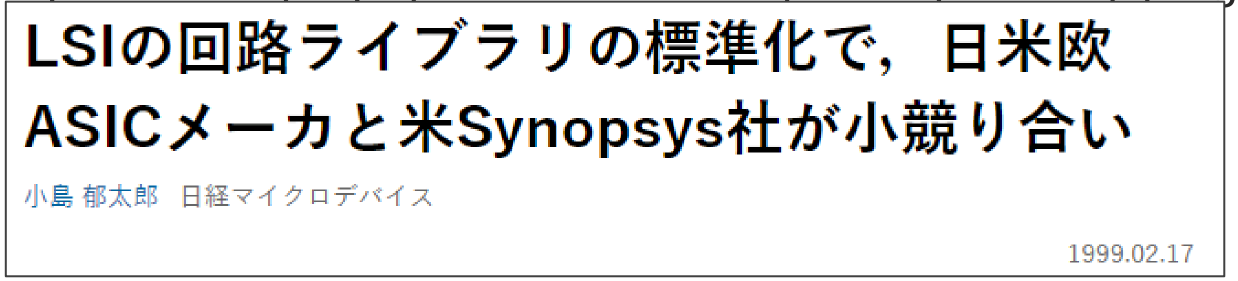
Technology Access Program (TAP-in)

Synopsys Technology Access Program (TAP-in) promotes interoperability through open source licensing of interface formats. To access the following Synopsys formats, you must first register and accept the open source license agreement for each format you wish to download.

Interconnect Technology Format (ITF)
Provides detailed modeling of interconnect parasitic effects that enables designers to perform accurate parasitic extraction for timing, signal integrity, power and reliability signoff analysis.

Liberty
A gate-level modeling technology for timing, noise, power and test behavior that powers Synopsys Design Platform and sign-off tools. Learn more about Liberty and the Liberty Technical Advisory Board by visiting IEEE Industry Standards and Technology Organization.

https://www.synopsys.com/community/interoperability-programs/tap-in.html


LSIの回路ライブラリの標準化で，日米欧ASICメーカと米Synopsys社が小競り合い

小島 郁太郎　日経マイクロデバイス

1999.02.17

Nikkei XTECH, 99/02/17, https://xtech.nikkei.com/dm/article/NEWS/20070402/129966/

What is characterizer ?

- Extract delay and power of combinational cell and sequential cells
  - Generate SPICE file
  - Simulate w/ SPICE
  - Store the results as .lib
- Commercial tools are available

SYNOPSYS®

SiliconSmart

A Smarter Way to Get PrimeTime Signoff-Quality Timing Models

cādence®

Liberate Trio Characterization Suite

Viola

SILVACO

Q1: Why we need characterizer

- Fab should provide this information, why?

- Ans.1: In some cases, special function cell will drastically improve Power Performance Area(PPA)
  - XOR help synthesis of Multiplier, reduce area 60% [a1]
- Ans.2: At the development of the fabrication process, cell library may not ready, or need modification

[a1] S. Nishizawa, et al., "Analysis and Comparison of XOR Cell Structures for Low Voltage Circuit Design," in *ISQED*, Mar. 2013, pp. 703–708.

Q2: What is the advantage to commercial tool?

- Ans.1: Nothing…
- Ans.2: Price ?
  - One fabless company using this characterizer
- Ans.3: Education ?
  - Can check generated netlist
- Ans.4: For open-EDAs, this characterizer will help designers to use their own standard cells