IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

DOI:10.1587/transfun.2022VLP0007

Publicized:2022/09/13

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY

The Institute of Electronics, Information and Communication Engineers

Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER Special Section on VLSI Design and CAD Algorithms

libretto: An Open Cell Timing Characterizer for Open Source VLSI Design

Shinichi NISHIZAWA[†a)] ***and*** **Toru NAKURA**[††b)]**,** ***Members***

SUMMARY We propose an open source cell library characterizer. Recently, free and open-sourced silicon design communities are attracted by hobby designers, academies and industries. These open-sourced silicon designs are supported by free and open sourced EDAs, however, in our knowledge, tool-chain lacks cell library characterizer to use original standard cells into digital circuit design. This paper proposes an open source cell library characterizer which can generate timing models and power models of standard cell library.

key words: *Open source VLSI design, Timing library, Characterizer*

1. Introduction

The progressive scaling of CMOS transistor fabrication technology achieves faster device speed, lower energy consumption and higher area density, and it results in the continuous improvement of VLSI circuit performance. The state-of-the-art circuit design in the advanced technology is strongly supported by both fabrication companies and commercial EDA vendors. However, this state-of-the-art design environment is very difficult to access since these technologies are protected strictly to prevent technology leakage.

On the other hands, several open-sourced VLSI design projects have been proposed [1], [2]. Many open EDAs are widely proposed and these EDAs enable these open-source VLSI design projects. Many open-sourced CAD tools are proposed to support system design to logic design, logic synthesis, place-and-route, test insertion, verification, and summarized as sevelral RTL-to-GDS flows [3]. However, to our knowledge, tool-chain lacks library characterizer to extract timing and power of standard cells to enable precise timing and power estimation using Static Timing Analysis (STA). Without this timing information, each design cannot ensure its feasibility of operation speed and operation correctness. Usually, designer expects foundly to provide both physical library and timing library for digital circuit design. However, the cell library provided by foundary is not one-size-fits-all: in some cases, library may need to be characterized at different operation conditions, or need to add specific functions to achive better power performance area of target circuit. To make designers to add own combinational or sequential cells into his/her circuit, it is very useful to use the characterizer to automatically extract the timing and power information.

This paper proposes an open and free timing characterizer, named **libretto**, to support accurate timing analysis in digital circuit design flow. **libretto** uses one of the major free spice simulator ngspice [4] for timing and power simulation. **libretto** automatically generates several spice files for ngspice, run ngspice, accumulate simulation results and export timing and power information as Synopsys Liberty format[5], which is industry-wide standard open format of timing and power description*. Note that **libretto** does not have any advantage to the commercial characterizer, but it enables to use characterizer for free, and accelerate open-sourced VLSI designs.

The rest of this paper is organized as follows. Section 2 describes related works on this field. Section 3 describes the overview of our characterizing flow. Section 4 describes the details of timing and power characterization. Section 5 describes the experimental results. Section 6 concludes this paper.

2. Related Works

Several works are available for non-commercial timing and power characterizing of standard cell library.

pharsoc[8] is a characterizer and standard cell libraries provided by *The Art of Standard Cell Library Design*. **pharsoc** can characterize both of combinational and sequential cells, and generate timing information as Liberty format. Disadvantage is that only the binary of the program is distributed. It looks that the individually different binaries are used for different types of logic, and only one type of Flip-Flop without set/reset is supported for characterization. Also, **pharsoc** uses Winspice3 as simulation engine but Winspice3 is a commercial simulator.

AutoLibGen[9] is an open sourced characterizer, which uses ngspice as simulation engine and extract information as Liberty format. Advantage is that it can express timing information as Composite Current Source (CCS) model to enable accurate timing estimation compared to conventional Non-Linear Delay Model (NLDM). Disadvantage is that it has no support for sequential cells and gate with multiple outputs. Also, the location of program source

[†]The author is with the Graduate School of Information, Production and Systems, Waseda University.

[††]The author is with the Department of Electronics Engineering and Computer Science, Fukuoka University.

a) E-mail: nishizawa@aoni.waseda.jp

b) E-mail: nakura@fukuoka-u.ac.jp

*Liberty is an open format, sponsored by Synopsys[6] and Liberty Technical Advisory Board of IEEE-ISTO[7].

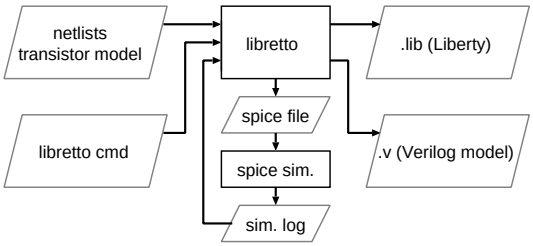


Fig. 1 Flow overview of **libretto**.

is not described in the paper.

LiChEn[10] is the another open sourced characterizer. **LiChEn** has been proposed to characterize asynchronous standard cells but it also characterizes basic combinational cells for VLSI design. Program code written in C/C++ is opened by author so anyone can use and extend this work. Disadvantage is that it uses Cadence Specter as a simulation engine, and lacks the ability of the sequential cell characterization.

ASCLIC[11][12] is a characterizer, which is developed to use characterizer in free. **ASCLIC** target to generate NLDM model and Non-Linear Power Model (NPTM) to enable both timing simulation and power simulation, and authors claim **ASCLIC** support multi-core system efficiently. Disadvantage is that it does not support sequential cell characterization.

SpiceGen[13] generate circuit netlist and spice simulation file to evaluate logic cell performance via web interface. **SpiceGen** supports to sweep the parameters of the cell to tune or optimize the performance. Disadvantage is that **SpiceGen** is not characterizer, and it looks **SpiceGen** does not support sequential cell simulation.

This work, named **libretto**, is a free and open-sourced characterizer. **libretto** is implemented using Python3, and ngspice is selected for simulation engine, assumes to running on Linux system. It supports to characterize both combinational and sequential cells. Advantage is the wide support of the functions for characterization, especially for sequential cells with positive edge or negative edge clock, set and reset support. **libretto** extract timing and power information as NLDM and NLPM, respectively, and exports in Liberty format which is industry-wide standard format.

3. Overview of characterization

Figure 1 shows the overview of characterization in **libretto**. Designer need to prepare following files,

- netlist of target cells, and transistor model,
- command file which contains characterize settings for library and individual logic cells.

libretto reads command file and automatically generates spice files to run spice, read spice outputs, and write out as Liberty format. **libretto** also outputs Verilog-HDL model for gate level simulation.

Figure 2 shows the internal flow of **libretto**. In first

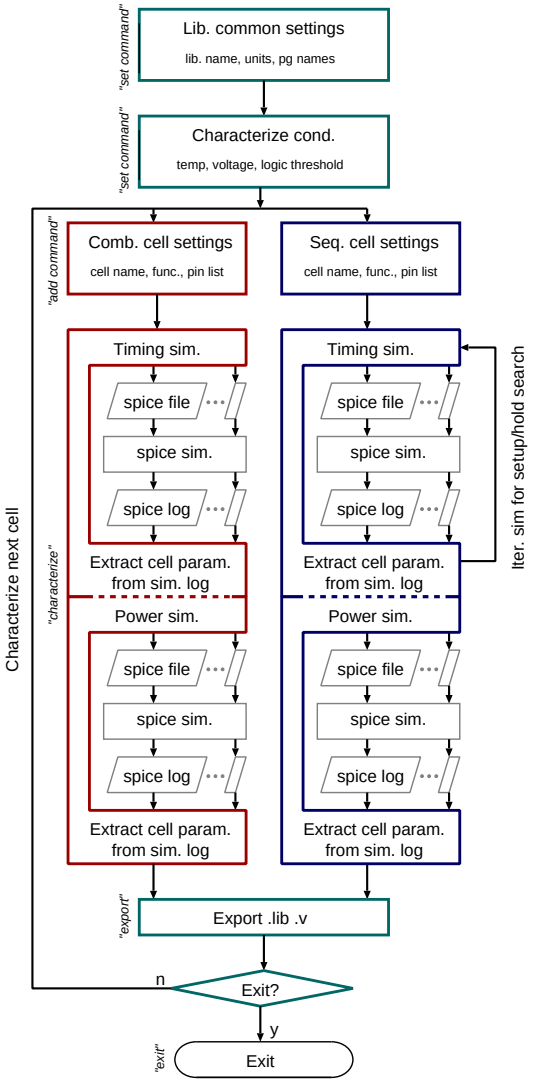


Fig. 2 Internal flow of **libretto**.

stage, library common settings are defined. These common settings are common parameters for whole library, such as

- library name,
- .lib file name,
- cell name prefix, suffix (if needed),
- units of voltage, current, capacitance, resistance, power,
- power, ground, p-well, n-well name.

In second stage, characterized conditions are defined. These are also common settings for whole library, such as

- PVT (process, voltage, temperature) conditions,
- logic threshold for delay definition,
- simulator binary location.

Third stage branches depend on the target cell function (combinational or sequential). In this stage, individual settings for each logic cell are defined. Each cell requires to set its

Listing 1

```
if(targetCell.logic == 'NAND2'):
  ## [in0, in1, out0]
  expectationList2 = [['01','1','10'],\
                      ['10','1','01'],\
                      ['1','01','10'],\
                      ['1','10','01']]
  return runCombIn2Out1(targetLib, targetCell,
      expectationList2,"neg")
```

Listing 2

```
if(targetCell.logic == 'DFF_PCPU_NRNS'):
  ##      [ D, C, S, R, Q]
  expectationList2 = [['01','0101', '1', '1', '01'], \
                      ['10','0101', '1', '1', '10'], \
                      [ '0', '0101','10', '1', '01'], \
                      [ '1', '0101', '1','10', '10']]
  ## run spice deck for flop
  return runFlop(targetLib, targetCell, expectationList2)
```

- spice subcircuit name,
- logic function type,
- input, output pin name,
- input slopes, output loadings,
- simulation time step,
- function written in Verilog-HDL.

Additionally, sequential cell requires following information to characterize, such that

- clock pin name,
- set/reset pin name (if required),
- name of storage elements.

In fourth stage, spice files of target cell are generated and invoke spice simulation. This simulation stage composed of two stage. Firstly, timing simulation stage is performed to extract propagation delay, transition delay, start point of input waveform and end point of output waveform. Secondly, power simulation stage is performed to extract leakage power and dynamic power. Combinational cell needs this simulation for all of the input patterns with different input slew and output loading. Sequential cell needs much larger simulation runs to find the setup time and hold time (detail is described in the next section).

4. Detail of characterization

This section describes the detail of the timing characterization and power characterization. **libretto** uses straight forward implementation of cell characterization, based on Liberty Users Guide[5].

4.1 Logic type definition for simulation

For the successful simulation, characterizer should know the input values and expected output value. **libretto** needs this information as truth table internally, and call the target truth table depends on its function.

Table 1 Set of cell functions.

Logic 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
Adder	Half Adder, Full Adder
Flip-Flop	pos. clk, pos. unate
Flip-Flop	pos. clk, neg. unate
Flip-Flop	neg. clk, pos. unate
Flip-Flop	neg. clk, neg. unate
Flip-Flop	pos. clk, pos. unate, neg. async. reest
Flip-Flop	pos. clk, pos. unate, neg. async. set and reest

Listings 1 shows the internal description of NAND2 characterization. NAND2 has two inputs thus rows should be four (2^2). In this table, "0" and "1" are the stable values and "01" and "10" are the non-stable values. For example, in first column, inputs **in0** and **in1** are "01" and "1", respectively. This means **in0** is driven from VSS to VDD voltage, and **in1** is connected to VDD. Output **out0** is "10" and it means **out0** is expected to swing from VDD to VSS voltage.

Listings 2 show the example code of Flip-Flop, which has positive edge clock, asynchronous negative set and reset, with positive unate output. We define this function as "DFF_PCPU_NRNS" in **libretto**. In this case, first two rows correspond to the rise and fall edge of data input, and next two rows correspond to the fall edge of set and reset cases. Clock value of "0101" means two clock rising edges are connected into Flip-Flop; first edge is to fix initial value, and next edge is for delay measurement.

libretto need this information to generate correct spice simulation file. If the target function is not registered, **libretto** outputs warning so designer need to define the function and truth table. Table 1 shows the supported functions in current version.

4.2 Timing characterization

4.2.1 Common setting for timing simulation

In timing characterization, propagation delay and transition delay are extracted from simulation. Propagation delay ($d_{\mathrm{prop,L(H)}}$) is the time from half-VDD of input ($V_{\mathrm{LH(HL)}}$) to output ($V_{\mathrm{HL(LH)}}$), and transition delay ($d_{\mathrm{tran,L(H)}}$) is the time from output logic-high(low) threshold ($V_{\mathrm{H(L)-TH}}$) to output logic-low(high) threshold ($V_{\mathrm{L(H)-TH}}$), and these definitions are illustrated as figure 3 [†]. **libretto** also measure the start time of input swing (t_{start}) and end time of output swing

[†]User of **libretto** can redefine these thresholds by command.

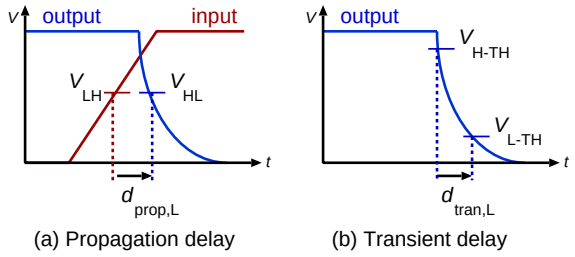


Fig. 3 Definition of propagation delay and transition delay.

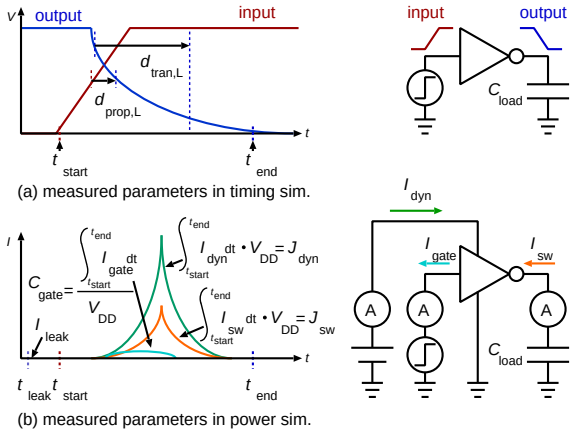


Fig. 4 ".measure" parameters in timing and energy evaluation.

(t_{end}) for dynamic power calculation. Figure 4(a) shows the ".measure" parameters in the timing simulation.

For accurate simulation, simulation time step and end time are very important, since ngspice has less ability to automatically adjust simulation time step and end time. **libretto** supports individual setting of simulation time step and end time by manual, or automatically set the time step and end time as 1/10× of minimum input slope, and 10× of maximum input slope, respectively.

In circuit simulation, ngspice runs in batch mode and it automatically finishes if circuit operate correctly and all the ".measure" statement finishes.

Spice simulation file is generated at simulator working directory. Designer can see and use this file to check the circuit operation. Also, designer can check the waveform enabling the ".control" section of the file. Figure 5 show the waveform of input and output of Inverter, generated by the ".control" section of simulation file.

4.2.2 Setup/Hold search for sequential cells

Sequential cell has several metrics of delay, and these are illustrated in figure 6. Let us consider a Flip-Flop with positive edge clock, which has data input (D), clock input (C), and data output (Q). C-to-Q (C2Q) is the delay from clock edge to output change. The data input must be stable around some time window of clock edge. D-to-C (D2C) is the delay between data edge before the clock and clock edge, and it is called setup time which is the minimum required time

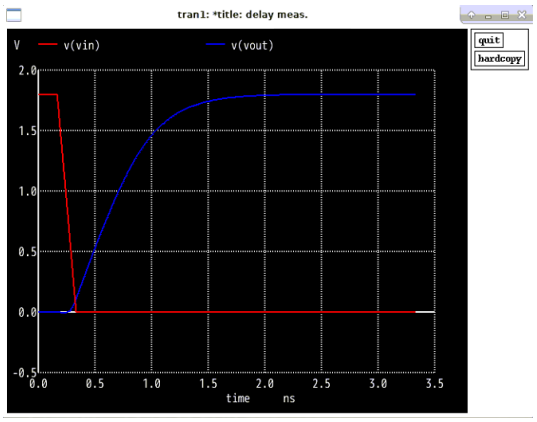


Fig. 5 Simulation waveform of Inverter using ngspice.

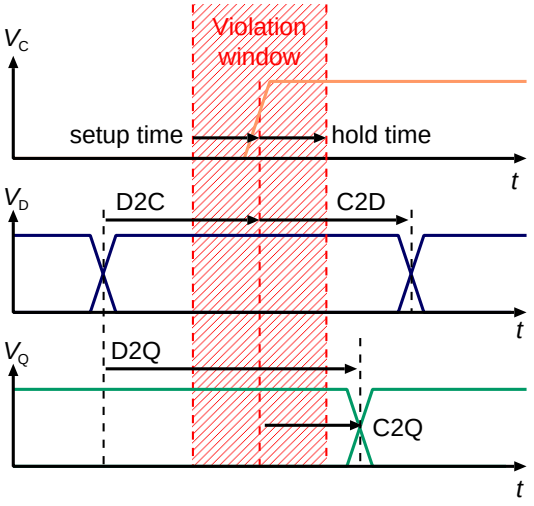
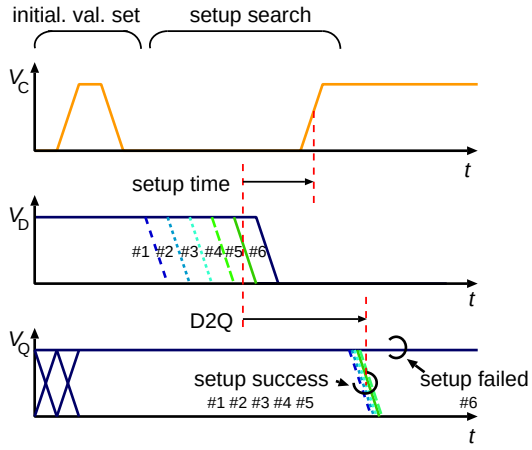


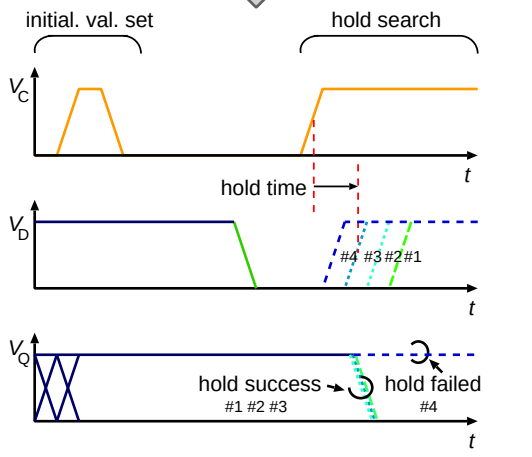
Fig. 6 Definition of delay in sequential cell.

that data input must be stable before the rising edge of clock input. C-to-D (C2D) is the delay between clock edge and data edge after the clock, and it is called hold time which is the minimum required time that data input must be stable after the rising edge of clock input. D-to-Q (D2Q) is the minimum delay from data edge to the output change, and it is the sum of D2C and C2Q delay.

Since the definition of setup and hold time is the minimum time of sequential cell timing, we need to search this timing with multiple simulation runs. There are two definitions for setup time: one is D2C which increase 3% to 5% larger C2Q from minimum C2Q, and another is the D2C which achieves minimum D2Q. In this paper, we use minimum D2Q to search setup time [14]. Figure 7 show the setup, D2Q, and hold search in simulation. Firstly, **libretto** generates multiple spice files from larger D2C to smaller D2C and find the minimum D2Q and define its D2C as setup. After the setup time is fixed, secondly, **libretto** generates multiple spice files from larger C2D to smaller C2D, and find the minimum C2D with correct Flip-Flop operation as hold time.



(a) Min. D2C (setup) search



(b) Min. C2D (hold) search

Fig. 7 Setup and hold search.

In the sequential cell simulation, it will not operate correctly with inadequate D2C or C2D condition. In this case, ".measure" statement of spice fails and large simulation time is needed. To prevent this problem, **libretto** firstly generate spice file which try to measure 10% voltage swing of output and check it will operate or not. If 10% voltage swing is observed in logic output, **libretto** secondly generate spice file with full simulation.

It is obvious that this simulation requires large simulation time, and parallel execution will help to speed up the characterization. However, this version of **libretto** uses sequential run of spice simulation and parallel execution is our future work.

4.3 Power characterization

In power characterization, we need to measure energy consumption per operation [†] and leakage power. Current flow from power source (I_{dyn}) and output loading (I_{sw}) are integrated from t_{start} to t_{end}, then converted as energy (J_{dyn}, J_{sw}). Energy consumption is calculated subtracting J_{sw} from J_{dyn}. For leakage power, **libretto** measure the average of current consumption (I_{leak}) of each input patterns at the start time t_{leak} of the simulation then calculate cell leakage power [††] . Gate capacitance is calculated from the charge of gate (integral of gate current I_{gate}) and divided it by supply voltage. This definition is common for both combinational cells and sequential cells. Figure 4(b) shows the ".measure" parameters for power simulation.

[†]Note that, in Liberty format, "power table" contains the information of energy consumptions (not power itself) as look-up table. This definition is comes from Synopsys Liberty User Guide[5].

[††]Liberty format supports leakage power model considering the input port voltage dependence, however **libretto** do not support this feature because it requires to simulate all of the possible input patterns.

Table 2 Characterization setting.

Characterizer	**libretto**		SiliconSmart
Simulator	ngspice	hspice	hspice
Simulator option	default	default	default
# of threads	1	1	32
Process tech.	Commercial 180-nm w/ small modification	Commercial 180-nm	
Input waveform	ramp		
Process cond.	Typical		
Voltage	Nominal (1.8 V)		
Temperature	25°C		
Input slope	0.01 ns, 0.1 ns, 1.0 ns		
Output load	0.1 pF, 0.7 pF, 4.9 pF		
Logic high/low voltage	1.44 V, 0.36 V (80%,20%)		
High-to-low low-to-high volt.	0.9 V, 0.9 V (50%)		
DUT for comb. cell	Inverter		
DUT for seq. cell	Flip-Flop w/ pos. edge clock		

5. Experimental Results

We implement **libretto** using Python3, and ngspice is used as simulation engine. We prepare several netlists of the cell from PDK which targets 180-nm process as an evaluation. Foundry provided spice model only supports Synopsys hspice, so small modification is added to use the spice model for ngspice. We use Synopsys SiliconSmart as a baseline of the characterizer and evaluate the performance of **libretto**. Also, we add support of hspice for **libretto** to check the algorithm difference in characterizer. Table 2 shows the common settings for both characterizers. Inverter is selected as a representative of combinational cell, and positive edge Flip-Flop is selected as a representative of sequential cell. Results does not added any timing and power mergins from simulation result, explicitly.

5.1 Combinational logic characterization

Inverter cell is characterized to compare its delay, energy consumption, leakage power and capacitance. Figure 8,9,10 show the comparison result of propagation delay, transition

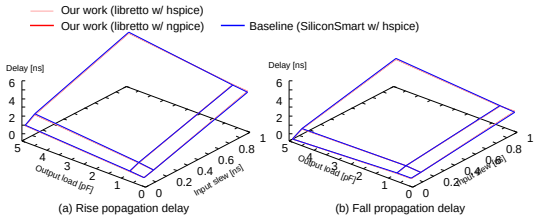


Fig. 8 Propagation delay of Inverter.

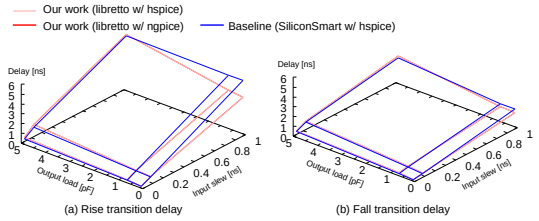


Fig. 9 Transition delay of Inverter.

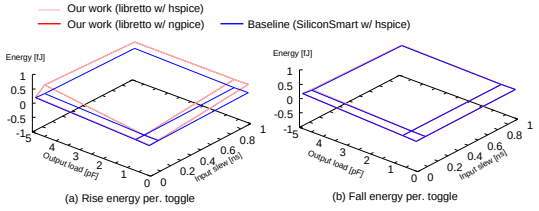


Fig. 10 Internal energy of Inverter.

Table 3 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

delay, and energy consumption. Characterization results in **libretto** with two different simulator ngspice (in red color) and hspice (in pink color) show almost same result, and two results are overlapped in these figures. Main difference in characterized result comes from characterizion algorithms in two characterizer. Three results of propagation delay match well. Transition delay has mismatch at the higher input slope. Energy consumption has large mismatch at the input rise case, it will result the power estimation in pessimistic. Maximum errors in propagation delay and transition delay are 0.50% in rise, 1.44% in fall. Internal energy shows 418% pesimistic estimation in rise condition. Table 3 shows the comparison result of leakage power and input capacitance. Leakage power looks matched well. Input capacitances has almost 10% mismatch.

Flip-Flop is characterized to compare its C2Q propagation delay, transition delay, setup, hold and energy consumption. Comparison results are plotted at figure 11, 12,13,14,15. Propagation delay and transition delay at the fall input pattern have some mismatch at large input slope and large output load conditions. Maximum errors in propagation delay are 1125% and 2407% at rise and fall conditions, respectively [†] .

[†]Maximum errors are observed at the small output load condition thus ratio value becomes large.

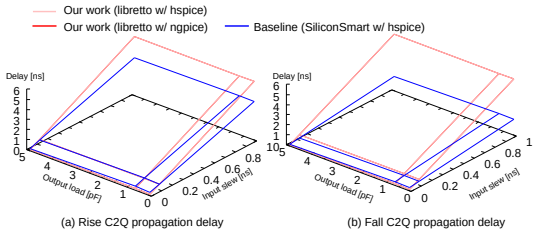


Fig. 11 C2Q propagation delay of Flip-Flop.

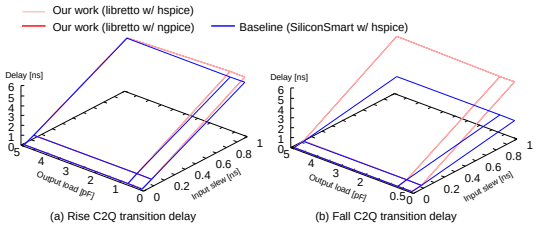


Fig. 12 C2Q transition delay of Flip-Flop.

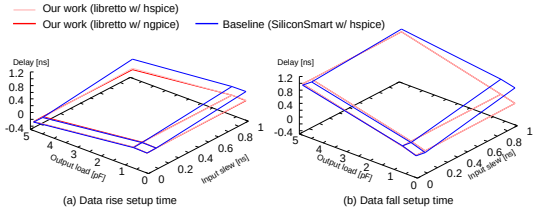


Fig. 13 Setup time of Flip-Flop.

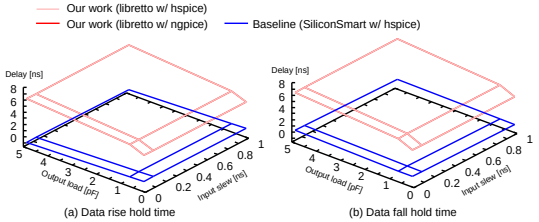


Fig. 14 Hold time of Flip-Flop.

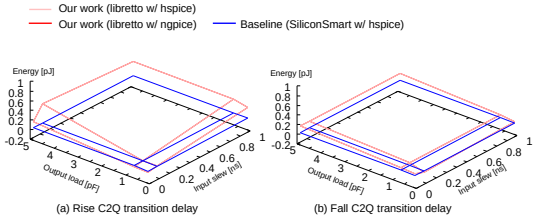


Fig. 15 Energy consumption of Flip-Flop.

From figure 13 and 14, **libretto** reports smaller setup time but larger hold time. One main reason is inter-dependence of the setup and hold time: tighter setup(hold) constraint requires longer hold(setup) time for correct operation of Flip-Flop[15]–[17]. **libretto** first tries to search minimum setup time of Flip-Flop thus it results larger hold time. The characterizing flow used in **libretto** is straightforward and reduce the accuracy of STA, thus several literature discuss to handle this inter-dependence of setup and hold time [18]–[20]. SiliconSmart can handle this issue to search setup and hold in the same time to find much more

Table 4 Environmane for characterization.

OS	CentOS 7.8
CPU	AMD Ryzen Threadripper 2990wx 3 GHz 32-core
MEM	DDR4-2400 96GB ECC
SSD	3TB

Table 5 Required time for characterization.

Characterizer	Simulator	# thread	CPU time	CPU time (norm.)
libretto	ngspice	1	147 min.	1 (baseline)
libretto	hspice	1	65 min.	2.26×
SiliconSmart	hspice	32	41 sec.	215×

balanced constraint. Figure 15 shows the energy consumption of Flip-Flop, and **libretto** reports pesimistic result of energy consumption. Maximum errors in energy consumption of Flip-Flop are 889% and 1915% at rise and fall conditions, respectively. Different setup constraints may affect this difference, since tighter setup constrain affect robust operation of and energy consumption of Flip-Flop.

5.2 Runtime comparison

We use Linux machine with AMD Ryzen CPU for characterization, which is summerized in table 4. Table 5 shows the CPU time for characterization. To characterize two cells, **libretto** with ngspice in single thread simulation requires 2.5 hour in total. **libretto** with hspice in single thread simulationachieves 2.26× faster characterization. SiliconSmart with hspice in 32-thread simulation achieves 215× faster characterizatin, compared to **libretto** with ngspice. This is because the sequential operation of Flip-Flop setup and hold search, since this search requires multiple spice simulation to find the minimum. Current version of **libretto** does not support multi-thread simulation, thus all of the simulation are done in sequential. Simple gradient method is used to search minimum in **libretto**, and this algorithm requires larger simulation run-time, compared to the modern commercial characterizer. **libretto** uses individual simulation for delay and power, since ngspice does not support nested structure of ".measure" statement. These inefficiency reads large difference in runtime comparison. Also ngspice itself is not so fast compare to the commercial spice simulator. Utilize the parallelism is our future work.

6. Conclusion

In this paper, we propose a free and open-sourced characterizer. It supports the characterization of both combinational cells and sequential cells. It supports both timing model and power model to enable timing estimation and power estimation in STA. Free and open-sourced characterizer will help open-sourced VLSI design project to add designers own cell into VLSI design. Program code and environment is uploaded in authors GitHub page.
https://github.com/snishizawa/libretto

Our future work is performance improvement and accuracy improvement in characterization. Characterization time can be reduced drastically when parallelism is enabled in spice simulation. Input waveform model should be improved from current simple ramp waveform to realistic driver model. Interdependency of setup time and hold time should be consider in the characterization to improve the accuracy of STA.

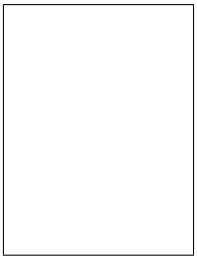
Acknowledgment

This work has been supported by Logic Research (190402, 210104), and funding from Fukuoka University (197105, 217301). This work is also partly supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc..

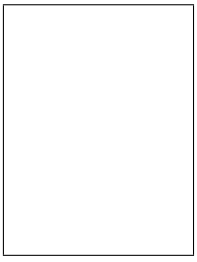
References

[1] "MakeLSI Project."
[2] R.T. Edwards, "Google/SkyWater and the Promise of the Open PDK," Workshop on Open-Source EDA Technology, 2020.
[3] R.T. Edwards, M. Shalan, and M. Kassem, "Real Silicon Using Open-Source EDA," IEEE Design and Test, vol.38, no.2, pp.38–44, 2021.
[4] Ngspice, ngspice - open source spice simulator.
[5] Synopsys, Liberty User Guide Volume 1.
[6] Synopsys, "Technology Access Program (TAP-in)."
[7] "Liberty Technical Advisory Board."
[8] G. Bronstein, I. N.; Semendjajew, K. A.; Musiol, "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," 2008 International Conference on Electronic Design, ICED 2008, 2008.
[10] M.T. Moreira, C.H.M. Oliveira, N.L.V. Calazans, and L.C. Ost, "LiChEn: Automated electrical characterization of asynchronous standard cell libraries," Proceedings - 16th Euromicro Conference on Digital System Design, DSD 2013, pp.933–940, 2013.
[11] M.S.I. Bin Hussin, Y.W. Lim, N.A. Kamsani, S.J. Hashim, and F.Z. Rokhani, "Development of automated standard cell library characterization (ASCLIC) for nanometer system-on-chip design," IEEE Student Conference on Research and Development: Inspiring Technology for Humanity, SCOReD 2017 - Proceedings, vol.2018-Janua, pp.93–97, 2018.
[12] C.H. Oliveira, M.T. Moreira, R.A. Guazzelli, and N.L. Calazans, "ASCEnD-FreePDK45: An open source standard cell library for asynchronous design," 2016 IEEE International Conference on Electronics, Circuits and Systems, ICECS 2016, pp.652–655, 2017.
[13] A. Beg, A. Elchouemi, and R. Beg, "A collaborative platform for facilitating standard cell characterization," Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2013, pp.202–206, 2013.
[14] N.H.E. Weste and D.M. Harris, CMOS VLSI Design, 4 ed., Addison Wesley, 2010.
[15] E. Salman, A. Dasdan, F. Taraporevala, K. Kucukcakar, and E.G. Friedman, "Pessimism reduction in static timing analysis using interdependent setup and hold times," International Symposium on Quality Electronic Design, pp.159–164, 2006.
[16] S. Srivastava and J. Roychowdhury, "Independent and interdependent latch setup/hold time characterization via Newton-Raphson solution and Euler curve tracking of state-transition equations," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.27, no.5, pp.817–830, 2008.

[17] A.B. Kanng and H. Lee, “Timing margin recovery with flexible flip-flop timing model,” International Symposium on Quality Electronic Design, pp.496–503, 2014.
[18] H. Seo, J. Heo, and T. Kim, “Clock Skew Optimization for Maximizing Time Margin by Utilizing Flexible Flip-Flop Timing,” International Symposium on Quality Electronic Design, pp.35–39, IEEE, 2015.
[19] E. Salman and E.G. Friedman, “Utilizing Interdependent Timing Constraints to Enhance Robustness in Synchronous Circuits,” Microelectronics Journal, vol.43, no.2, pp.119–127, 2012.
[20] E. Salman, A. Dasdan, F. Taraporevala, K. Küçükçakar, and E.G. Friedman, “Exploiting Setup-Hold-Time Interdependence in Static Timing Analysis,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.26, no.6, pp.1114–1125, 2007.



Shinichi Nishizawa received the B.E. degree from Ritsumeikan University, Japan, in 2009, and Ph.D degree from Kyoto University, Japan, in 2015. He is a Assistant Professor in Waseda University.



Toru Nakura (S ’02–M ’07) was born in Fukuoka, Japan, in 1972. He received the B.S. and M.S. degrees in electronic engineering and the Ph.D. degree from the University of Tokyo, Tokyo, Japan, in 1995, 1997, and 2005, respectively. He was a Circuit Designer of high-speed communication using SOI devices in Mitsubishi Electric, Hyogo, Japan, and NEC, Kanagawa, Japan. He was also an EDA Tool Developer with Avant! Corporation, OR, US. He was an Associate Professor with VLSI Design and Education Center and Electrical Engineering and Information Systems, University of Tokyo. He is currently a Full Professor with the Department of Electronics Engineering and Computer Science, Fukuoka University, Fukuoka, Japan. His current research interest includes signal integrity, reliability, power supply, digitally assist analog circuits, and fully automated analog circuit synthesis.

7. Appendix

Listings3 is an example command for **libretto** to characterize NAND2 gate and positive edge Flip-Flop with asynchronous negative set and negative reset. First block defines the common settings for the library (we call “set command”). Second block defines the common characterize conditions for the library (we also call “set command”). Third block defines the individual characterization condition for each cell (we call“add command”). “characterize” command start the characterization and “export” command exports data as Liberty format.

Listing 3

```
# common settings for library
set_lib_name OSU035
set_dotlib_name OSU035.lib
set_verilog_name OSU035.v
set_cell_name_suffix OSU035_
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_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 3.5
set_vss_voltage 0
set_pwell_voltage 0
set_nwell_voltage 3.5
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_energy_meas_low_threshold 0.01
set_energy_meas_high_threshold 0.99
set_energy_meas_time_extent 4
set_operating_conditions PVT_3P5V_25C

# initialize workspace
initialize

## add circuit
add_cell -n NAND2_1X -l NAND2 -i A B -o YB -f YB=A*B
add_slope {1 4 16 64}
add_load {1 4 16 64}
add_netlist NETLIST/AND2_1X.spi
add_model NETLIST/model.sp
add_simulation_timestep auto
characterize
export

## DFF, positive clock positive unate, async neg-reset, async neg -set
add_flop -n DFF_ARAS_1X -l DFF_PCPU_NRNS -i DATA -c CLK -s NSET -r NRST -o Q -q IQ IQN -f Q=IQ QN=IQN
add_slope {1 4 16 64}
add_load {1 4 16 64}
add_clock_slope auto
add_netlist NETLIST/DFF_ARAS_1X.spi
add_model NETLIST/model.sp
add_simulation_timestep auto
add_simulation_setup_auto
add_simulation_hold_auto
characterize
export

exit
```