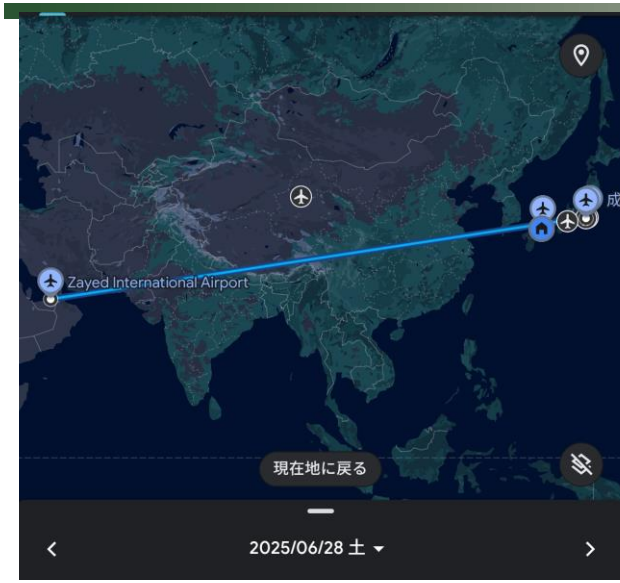
Free Silicon Conf. 2025 (FSiC)
訪問報告

広島大学 西澤真一
nishizawa@hiroshima-u.ac.jp

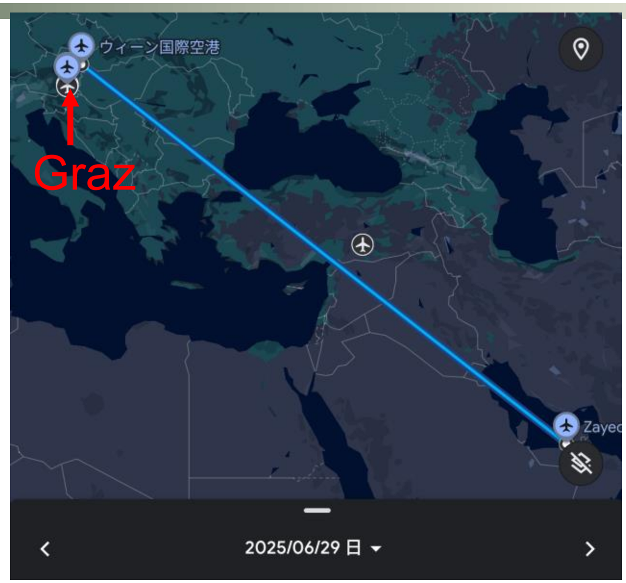
アウトライン

1. 6/30 に Graz University of Technology (TUGraz) へ訪問
   - Microwave Theoryの講義に関連して(?) 私の分野 (先端物理設計技術) に関する特別講義を開催
   - Title：Design Technology Co-Optimization for Continuous Innovation in Semiconductor Design
     - 訳：半導体設計の継続的な発展を支える設計・製造技術協調設計
2. 7/2～7/4 に Free Silicon Conf. (Frankfurt (Oder)) へ参加
   - オープンソース集積回路設計に関する会議
   - Title: libretto: An open-source library characterizer for open-source VLSI design

経路

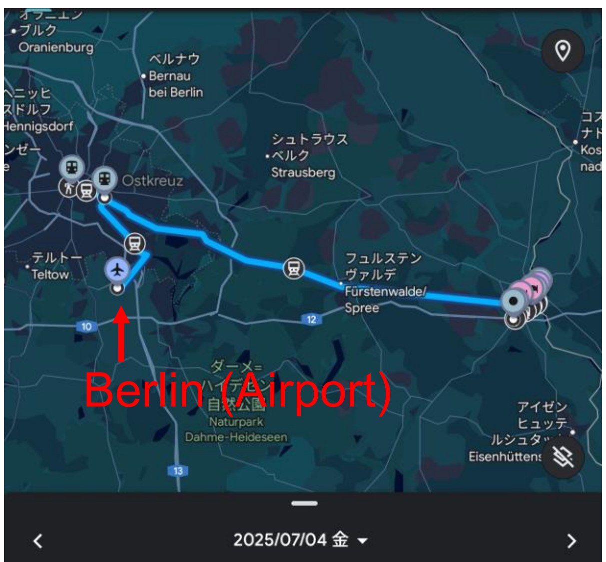


Zayed International Airport
現在地に戻る
2025/06/28 土

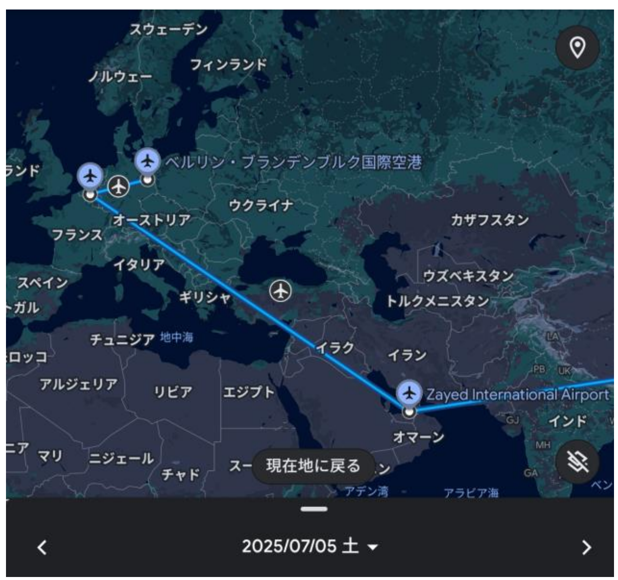


ウィーン国際空港
Graz
2025/06/29 日

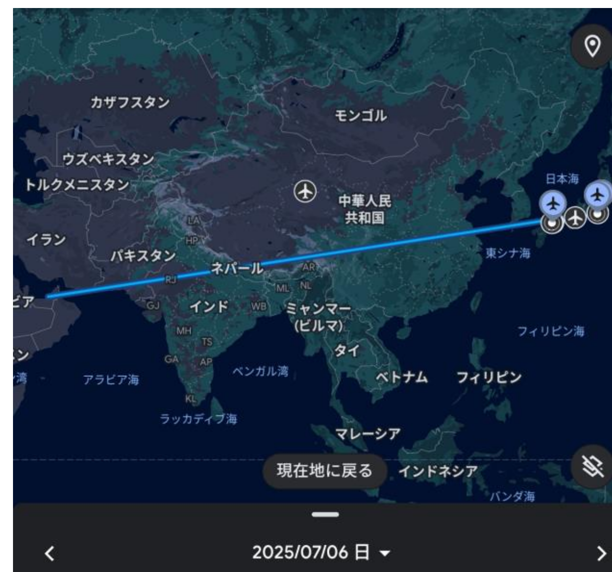
HOTEL POLONIA Frankfurt/Oder
Frankfurt
(Oder)
ウィーン国際空港
ibis Styles Graz Messe
2025/07/01 火



Oranienburg
ベルナウ
Bernau
bei Berlin
シュトラウス
ベルク
Strausberg
Ostkreuz
テルトー
Teltow
フュルステン
ヴァルデ
Fürstenwalde/
Spree
Berlin (Airport)
Naturpark
Dahme-Heideseen
2025/07/04 金



スウェーデン
フィンランド
ノルウェー
ベルリン・ブランデンブルク国際空港
ウクライナ
オーストリア
フランス
カザフスタン
イタリア
スペイン
ギリシャ
ウズベキスタン
トルクメニスタン
チュニジア
地中海
イラク
イラン
アルジェリア
リビア
エジプト
Zayed International Airport
インド
オマーン
マリ
ニジェール
チャド
現在地に戻る
2025/07/05 土



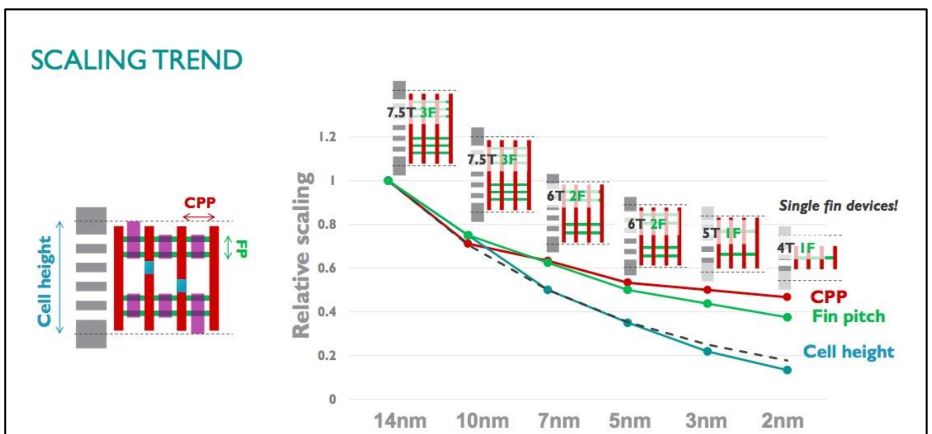
カザフスタン
モンゴル
ウズベキスタン
トルクメニスタン
中華人民
共和国
日本海
イラン
パキスタン
ネパール
東シナ海
インド
ミャンマー
(ビルマ)
フィリピン海
タイ
アラビア海
ベンガル湾
ベトナム
フィリピン
ラッカディブ海
マレーシア
現在地に戻る
インドネシア
2025/07/06 日

1. 製造技術の微細化とその限界

- 従来の集積回路の進歩は製造技術の微細化によるもの大
  - ムーア則，デナード則
  - 加工寸法1/kスケール：速度と消費電力 1/k，トランジスタ密度 k^2
- 単純な微細化は180nmプロセスあたりで速度低下
  - 180nm：位相シフトマスク，銅配線，STI (Shallow-Trench Iso.)
  - 90nm：歪みシリコン
  - 45nm：High-K/MG,
  - 28nm：最後のシングルパターニング
  - 22nm：FinFET，ダブル&トリプルパターニング，カラーリング
    - 製造技術の進歩だけではトランジスタ密度向上が困難に

1.設計･製造技術協調設計 (DTCO)

- 製造だけが頑張るのでなく設計と協調しトランジスタ密度を向上
  - 単位面積 (セル) 内のトランジスタ並列数を下げ，単位高さを下げ密度を上げ高性能なセルを実現
  - One-size-NOT-fits-all: 製造は数多くのライブラリを提供．設計は回路にとってベストなものを選ぶ
- 報告者のDTCOに関するライブラリ設計技術について紹介



Dan Micuta on *Nanosheets, CFETS: A Perspective on Logic Scaling and Beyond, in* imec technology forum (ITF) 2018

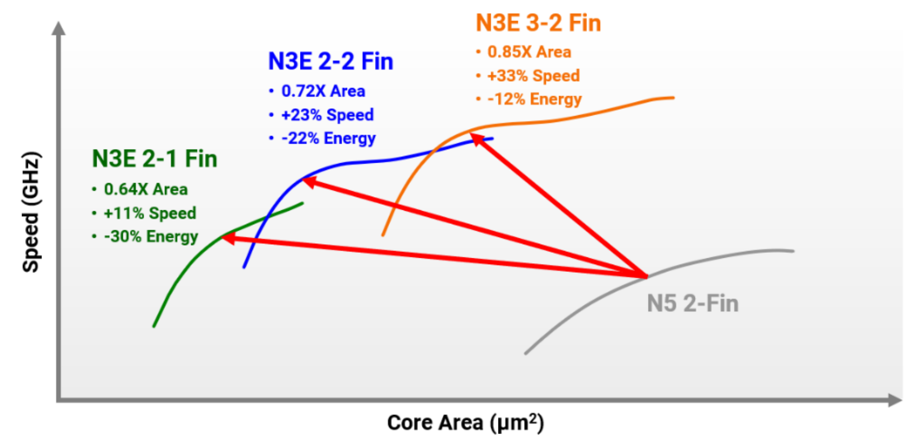


Diagram 1: N3 with FINFLEX delivers maximum flexibility and gives chip designers the ideal characteristics for each of the key functional blocks on the same die, with the same design toolset.

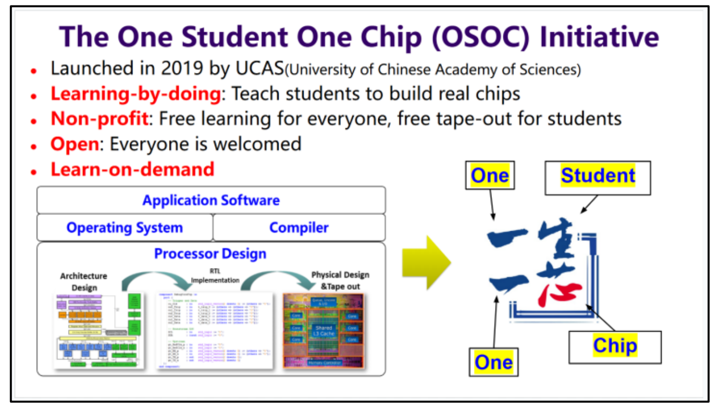
“TSMC FINFLEX™ – Ultimate Performance, Power Efficiency, Density and Flexibility”, TSMC Blog

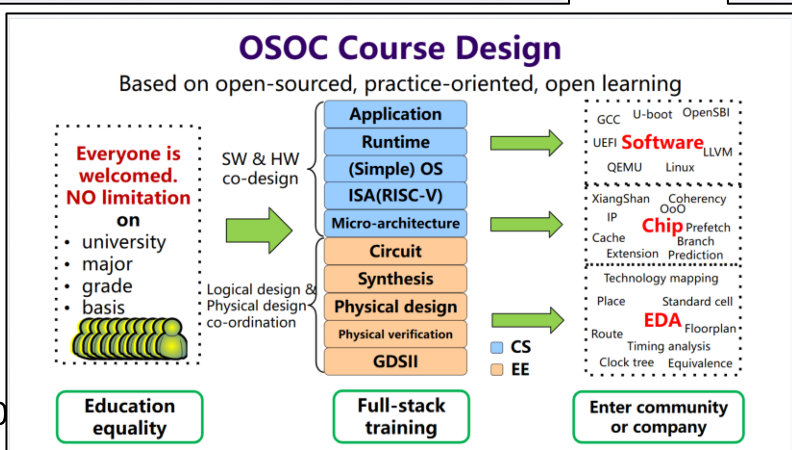
2. Free Silicon Conf. (FSiC)

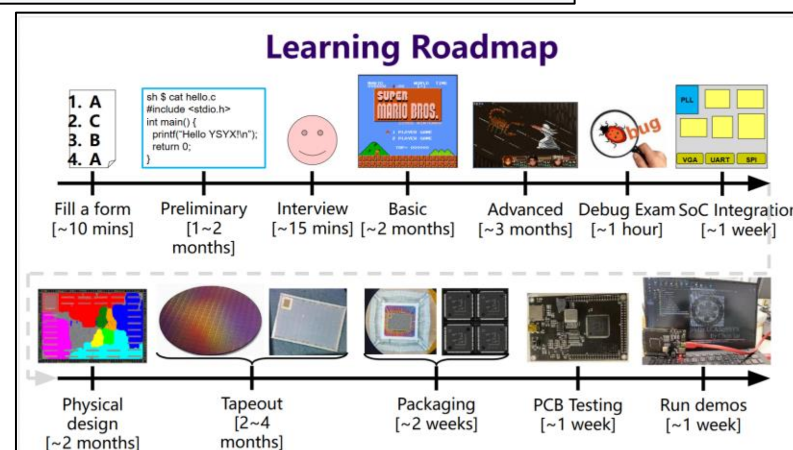
- オープンソース集積回路設計 (Open-Source Silicon) の会議
  - Leibniz-Institut für innovative Mikroelektronik: IHP開催
  - Google+Skywater+eFablessに代わりOSSを牽引していく
  - 設計，設計環境，EDA，ファンディング，教育，標準化を議論
- 私は開発しているEDAに関して発表を行った
  - キャラクタライザ：セルの電力遅延を抽出するツール
    - ほぼ同時期にOSSキャラクタライザは3つ提案された
      - lctime：Thomas Kramer さん (FSiCの運営) 開発，広く使われている
      - libretto: 西澤の提案
      - CharLib: オクラホマ州立大開発. (librettoのコピーに別名つけて公開)
        - コードはあとで書き直したらしいが…
        - librettoのアイディアをさも自分たちのもののように発表するのは…

2. 一生一芯プロジェクト

- 学部各個人でRISC-Vプロセッサを作るプロジェクト
  - SW/HWコデザイン+チップの物理実装まで
  - チップは製造し，PCB実装後発送，オンラインで発表
  - 教材，TA，全てオンライン，12000アカウント@2025







- FSiCでの発表資料に続く

libretto: An open-source library characterizer for open-source VLSI design

Shinichi Nishizawa, Hiroshima Univ., nishizawa@hiroshima-u.ac.jp

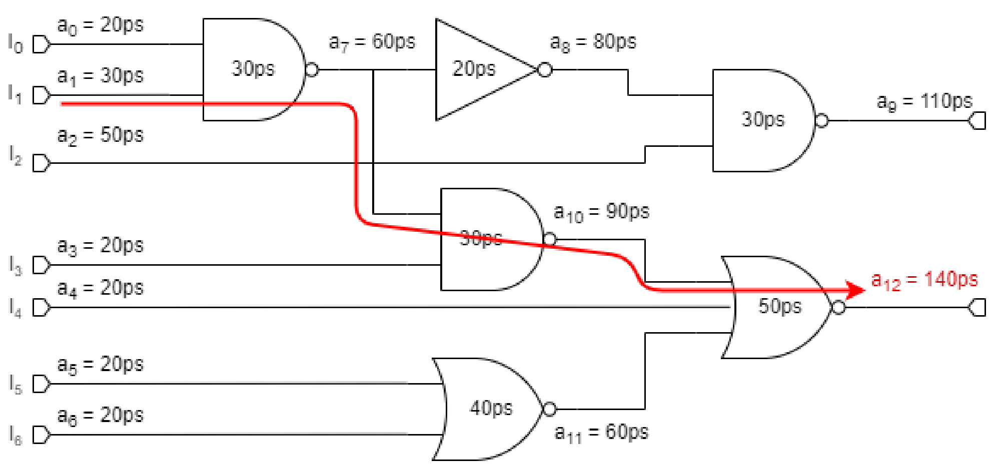
広島大学

ISHI-Kai (Japan)

Open Source EDA supporters (Japan)

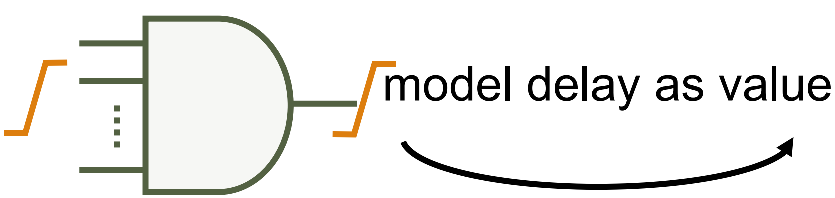
Digital circuit design and STA

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



Characterizer

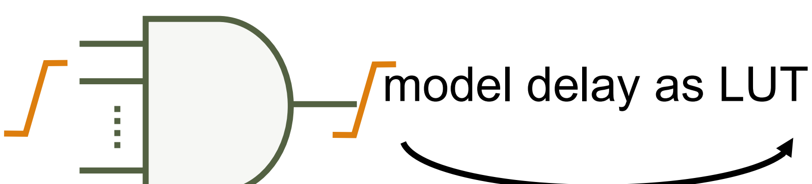
- Simulate and extract delay and power of std. cell
  - Prop delay: t_{pd} = xx ps. Trans delay: t_{td} = xx ps
    - Arrival time: $a_i = \max_{j\in \mathrm{fanin}(i)}\{a_j\} + t_{\mathrm{pd},i}$

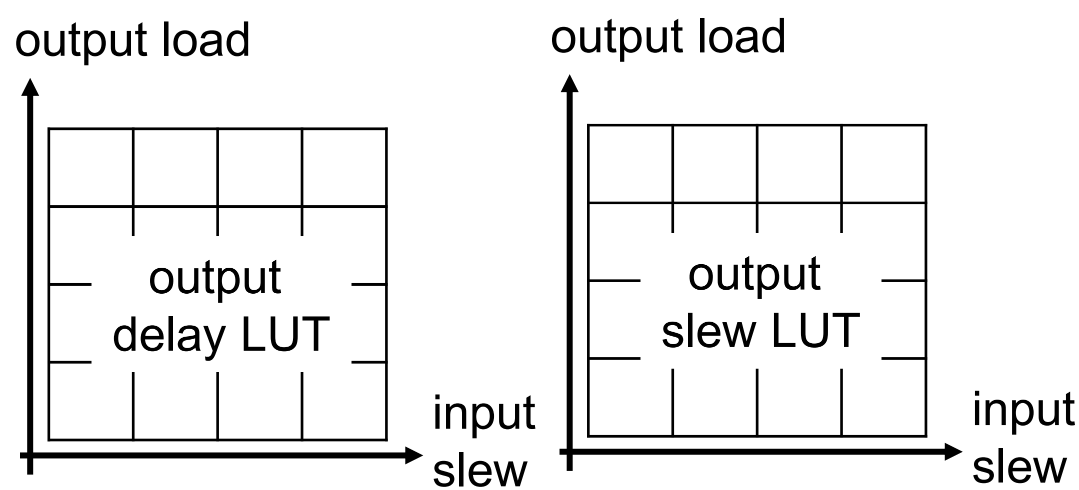


Prop delay: t_{pd} = xx ps.
Trans delay: t_{td} = xx ps

Characterizer

- Simulate and extract delay and power of std. cell
  - Prop delay: t_{pd} = xx ps. Trans delay: t_{td} = xx ps
    - Arrival time: $a_i = \max_{j \in \mathrm{fanin}(i)}\{a_j\} + t_{\mathrm{pd},i}$
- Delay and energy are the function of input slope, output load
    - Arrival time: $a_i = \max_{j \in \mathrm{fanin}(i)}\{a_j\} + t_{\mathrm{pd},i}(t_{\mathrm{pd,PI}_j}, C_{\mathrm{load},i})$

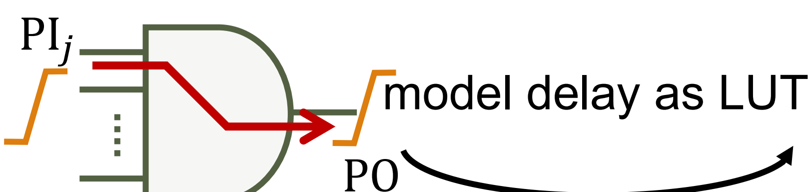


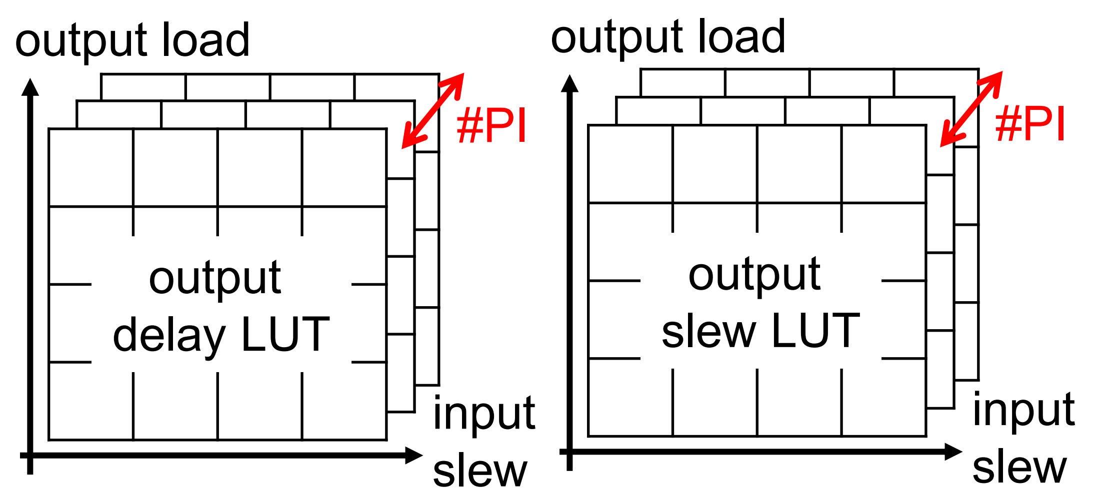


Characterizer

- Simulate and extract delay and power of std. cell
  - Prop delay: t_{pd} = xx ps. Trans delay: t_{td} = xx ps
    - ❌ Arrival time: $a_i = \max_{j \in \mathrm{fanin}(i)}\{a_j\} + t_{\mathrm{pd},i}$
- Delay and energy are the function of input slope, output load
    - ❌ Arrival time: $a_i = \max_{j \in \mathrm{fanin}(i)}\{a_j\} + t_{\mathrm{pd},i}(t_{\mathrm{pd,PI}_j}, C_{\mathrm{load},i})$
- Each primary input (PI) has different delay and energy
    - Arrival time: $a_i = \max_{j \in \mathrm{fanin}(i)}\{a_j\} + \underline{t_{\mathrm{pd},i,\mathrm{PI}_j}(t_{\mathrm{pd,PI}_j}, C_{\mathrm{load},i})}$

Characterizer automates simulation & lib. creation





Related works

- Open-source characterizers w/ free SPICE are available
  - LibreCell (lctime) [1]：Widely used, well known
    - Uses “template .lib” to specify char. condition
    - Pro. tool (need to read/write .lib by engineer)
  - CharLib [2]：
    - Uses YAML to specify characterize condition
    - Needs PySpice backend
    - Supports multithread
  - And many open characterizers that uses commercial SPICE

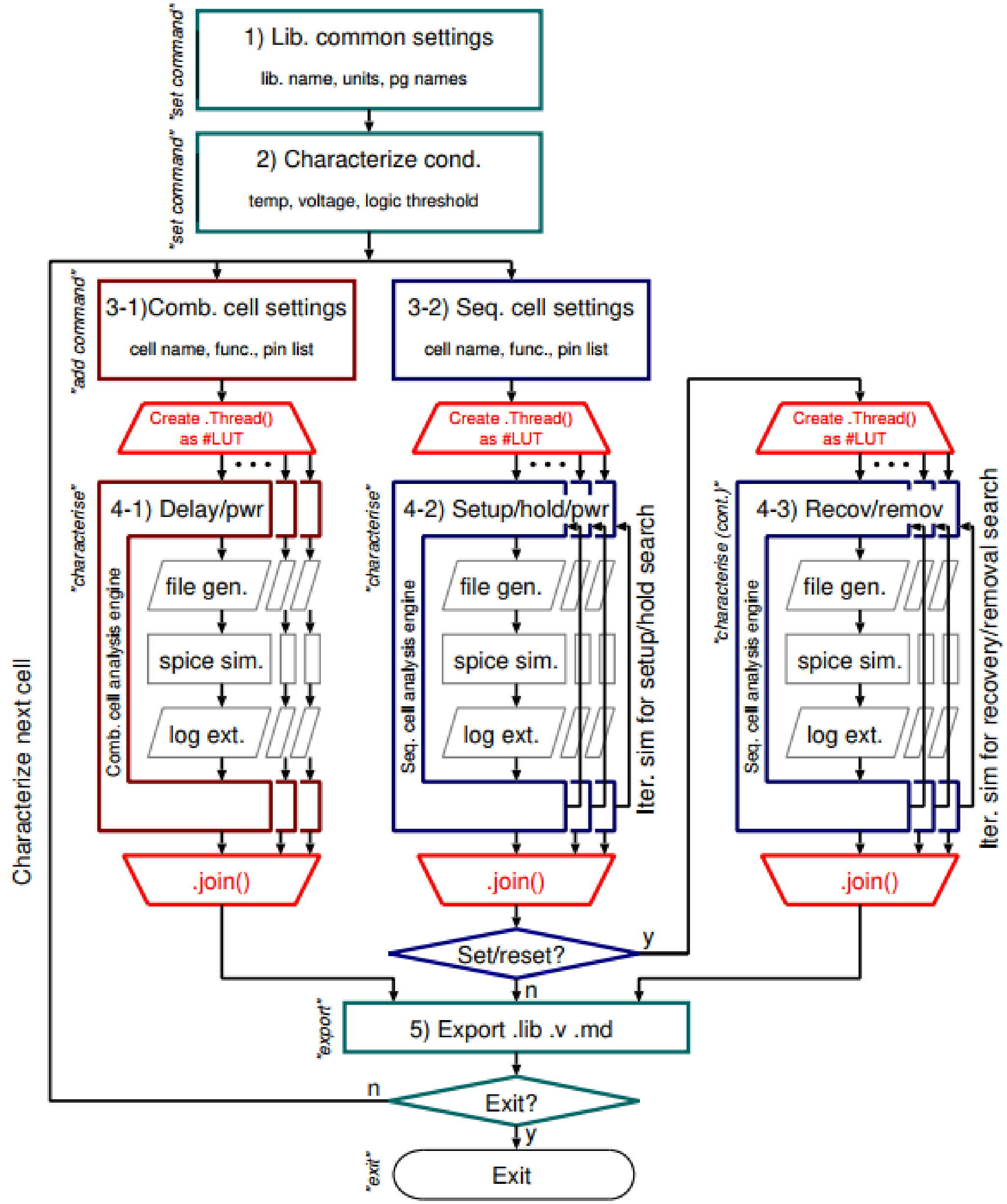
[1] T. Kramer, “lctime.” https://codeberg.org/librecell/lctime
[2] J. E. S. Jr., “CharLib.” https://github.com/stineje/CharLib.

Proposed characterizer (libretto)

- Open-source characterizer
  - Language: Python3
  - Simulator: ngspice, hspice (for comparison)
- Advantage
  - Characterize both combinational and sequential cells
    - Support Flip-Flops w/ pos/neg clock and async. set/reset
  - Easy to add functions
    - Two analysis engines: for combinational and sequential
      - Do not prepare engines for each logic function
      - Use truth table to handle different logic functions
  - Nothing new. No advantage over commercial tools
    - But open and free

Operation flow

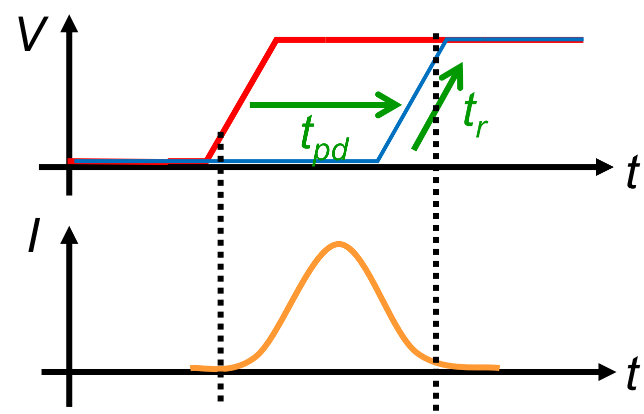
- Setup library and cells
- Branch to comb. or seq.
  - Generates test bench based on the target logic function
  - Launch simulator
    - Multiple slew/load cond. in multi-thread
- Seq. cell need iterations
  - Find min. delay by setup/hold search



Operation flow.

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 : integrate current at the beginning of sim*[2]
- Simulation times: time_step, end_time
  - Set by designer, or use “auto”



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

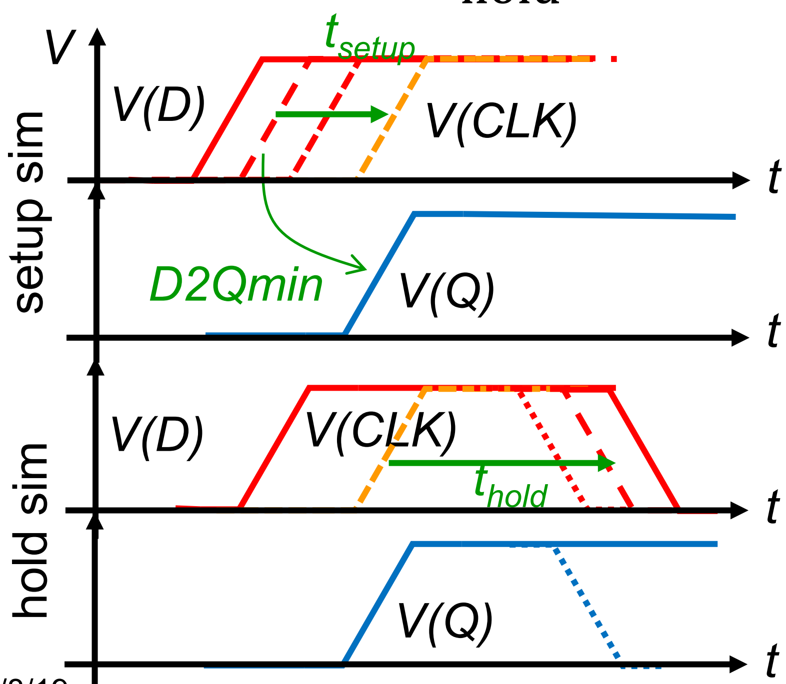
* Parameters can be changed by designer.

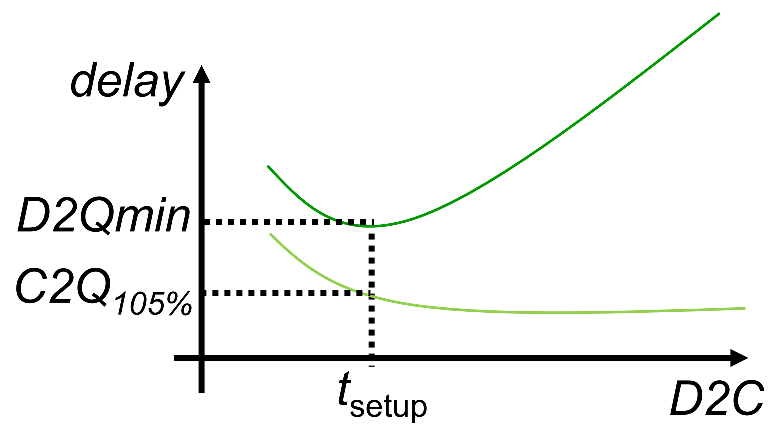
*[1]: This should be converted to energy

*[2]: Does not 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

libretto use def. of 2.

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 conditions
  - Add cell
  - Input slope (in array)
  - Output load (in array)
  - Netlist
  - Timestep*. sim. end*
- Flip-Flop needs
  - 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

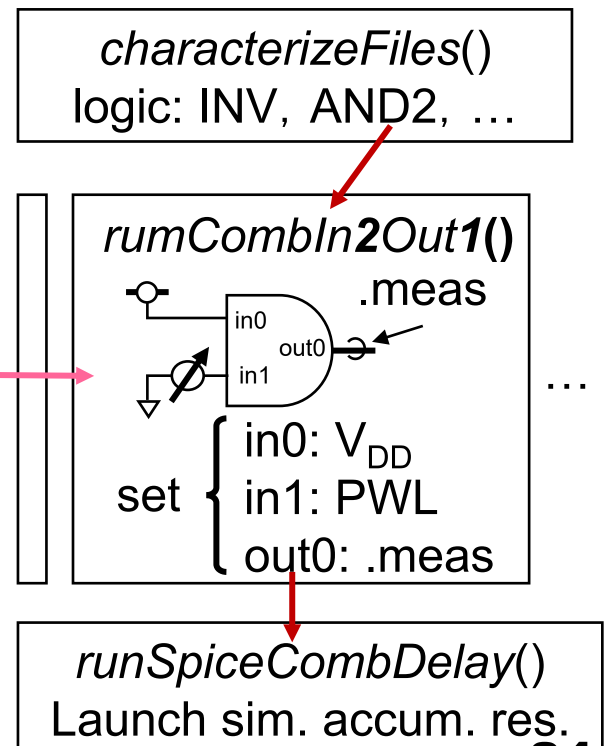
Command example: logic function, in/out pins, storage, logic expression, slew/cap index, simulation time step

Netlist gen. and simulation (comb. cell)

- *characterizeFiles*() : def. of logic func., its truth table
- *runCombInnOutm*() : 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 a function for netlist gen., spice run, analysis (*runSpiceCombDelay*())

- Pin is analyzed, connected proper voltage sources and spice measure statements

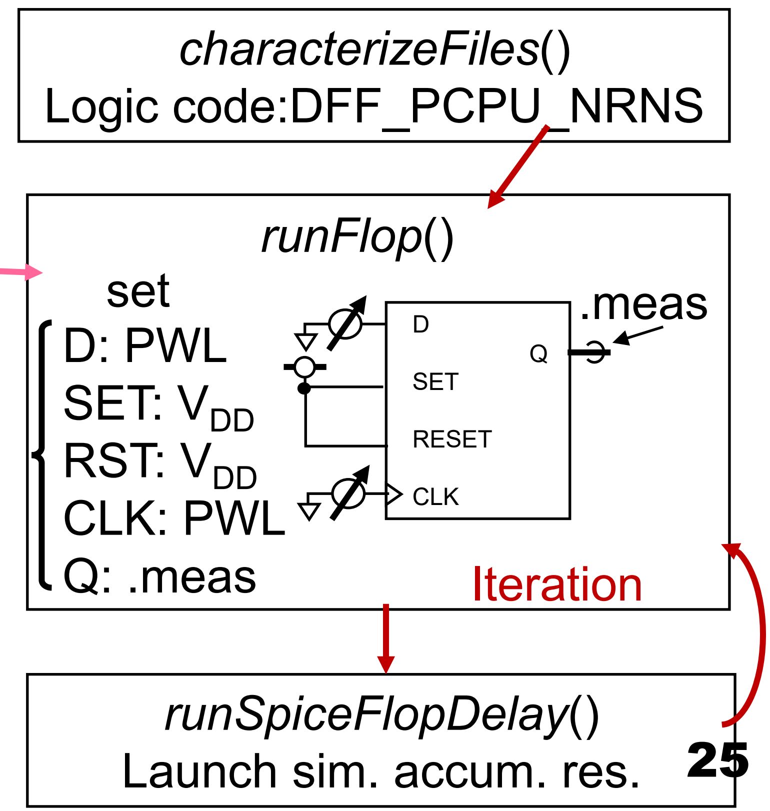
Netlist gen. and simulation (seq. cell)

- *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



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 polarity

Evaluation setup

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

- Use commercial 180-nm for evaluation

	Proposed		PrimeLib	
Simulator	ngspice		hspice	
#parallel	1	49 (#index)	32	64
Condition	Typical (TT, 1.8V, 25℃)			
Slew (ns)	0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4			
Capacitance (pF)	0.01, 0.02, 0.04, 0.08, 0.16, 0.32, 0.64			
Device Under Test	INVx1, NAND2x1, NAND3x1, NAND4x1 NOR2x1, NOR3x1, NOR4x1, DFF (posedge)			
Runtime	57.9 h (1x)	4.00 h (**14.5x**)	92.0 s (2265x)	106 s (1966x)

- Large runtime: need more parallelism, poor search algorithm.
- Separated timing and power sim. is also issue (ngspice do not support nesting of .meas)

Result

- No difference in simulator (ngspice vs HSPICE)
- Two characterizers show different result (PrimeLib vs libreto)
  - Delay of comb. cells might be acceptable (Max error in prop.: 0.5%, trans: 24%)
  - Intl. energy has large error: (Max 418%)
  - Seq. has problem (C2Q delay : 1125%, 2407% pessimistic)
    - Setup/hold interdependence (?)

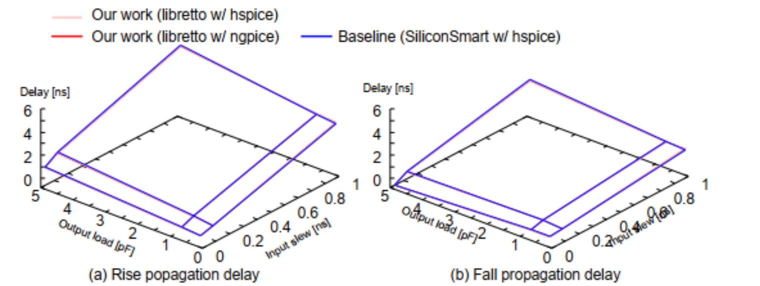


Fig. 6: Propagation delay of Inverter.

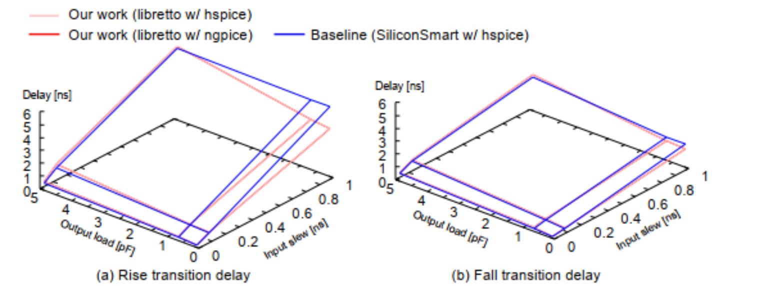


Fig. 7: Transition delay of Inverter.

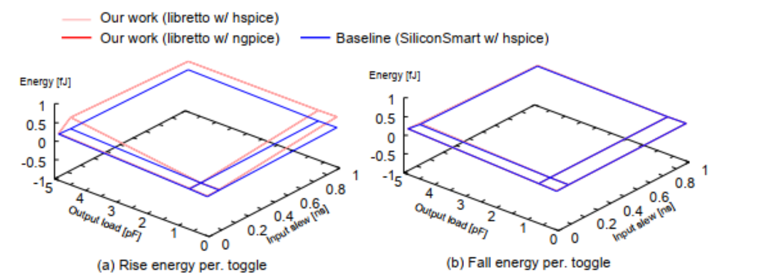


Fig. 8: Internal energy of Inverter.

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

Conclusion

- Open characterizer w/ open simulator
  - Generates timing/power library as .lib
  - Used for timing analysis (simulation, STA)
- Supports both combinational and sequential cells
- Evaluate delay, energy, and performance
  - Slow processing speed (1/215),
  - Combinational cell: delay and energy acceptable (?)
  - Sequential cell: large gap
- Checked by LibraryCompiler Synopsys
- Users: 1 in Japan, 1 in Itally
  - Let me know if you use libretto (motivate me!)

- 以上