オープンソース集積回路設計に向けたオープンソースキャラクタライザ

西澤 真一[1,a)]　名倉 徹[2,b)]

概要：本稿では集積回路設計で利用するオープンなセルライブラリキャラクタライザについて報告する．近年オープンソース集積回路設計が活性化している．これらはオープンソース EDA に支えられているが，我々の調査した限りにおいて，オープンソースのキャラクタライザとして有力なものが提案されていない．そこで我々はオープンソースのキャラクタライザを開発する事によってセルライブラリのタイミング特性および電力特性の自動抽出を実現することで，設計者が設計したセルライブラリを利用したデジタル回路設計を可能にする．

Open-source library characterizer for open-source VLSI design

Shinichi Nishizawa[1,a)]　Tohru Nakura[2,b)]

Abstract: We propose an open source-library characterizer for open-source silicon design. Recently, open-source silicon design has been attracts hobby designer, academies and industry. Open-source silicon design has been supported by many open-source EDA tools. However, in our knowledge, it lacks open-source library characterizer to extract timing information and power information of cell library. We designed a open source cell library characterizer to extract both timing information and power information of his/her own library cells and enable to use them in digital circuit design.

1. 序論

集積回路の製造プロセスの微細化によって，集積回路はその高速化，低消費エネルギー化，高集積化のすべてを達成してきた．このような最先端の集積回路設計は高度な製造プロセスと発達した商用 EDA ツールによって支えられてきた．一方でこのような最先端の集積回路設計環境は機密情報とされ，一部の限られた設計者しか技術の詳細が明かされていない．

一方で，集積回路設計をオープンソース化する設計プロジェクトが近年注目を浴びている [1], [2]．多くのオープンソース EDA を利用したオープンな集積回路設計が実現されている [3]．一方で，私たちが調べた限りにおいて，オープンソースの有力なセルライブラリキャラクタライザは提案されていない．キャラクタライザはセルのタイミング情報および電力情報の抽出に利用され，大規模回路の高精度な遅延見積もりおよび電力見積もりに必要である．

本論文ではオープンソースなセルライブラリキャラクタライザの開発内容について報告する．提案するキャラクタライザはオープンソースな SPICE シミュレータである ngspice[4] を採用し，SPICE ファイル生成，シミュレーション実行，実行結果からタイミング情報および電力情報を抽出し，業界標準の Synopsys Liberty [5] 形式としてデータを出力する[*1]．提案するキャラクタライザは商用キャラクタライザに対して優位性を主張するものではないが，オープンとする事でオープンソース集積回路設計の発展を後押しする事を期待している．

本論文は以下のような構成である．2 章では関連研究について述べる．3 章ではキャラクタライザの全体像について述べる．4 章ではキャラクタライザが実施するタイミング情報および電力情報の抽出について述べる．5 章では実験結果について述べる．6 章にて結論を述べる．

2. 関連文献

非商用のキャラクタライザはすでにいくつか提案がなされているため紹介する．

pharsoc[8] は *The Art of Standard Cell Library Design* にて提供されているスタンダードセルライブラリおよびキャラクタライザである．**pharsoc** は組み合わせセルおよび順序セルの両方をキャラクタライズする事ができ，タイミング情報を Liberty 形式で出力する事が可能である．課題は，プログラムがバイナリデータで配布されている事と，論理ごとに異なるバイナリが必要であると思われる事である．対象とする順序セルはセットおよびリセット

[1] 早稲田大学 大学院情報生産システム研究科
Graduate School of Information, Production and System, Waseda University
[2] 福岡大学 工学部
Faculty of Engineering, Fukuoka University
[a)] nishizawa@aoni.waseda.jp
[b)] nakura@fukuoka-u.ac.jp

[*1] Liberty は, Synopsys[6]　および IEEE-ISTO における Liberty Technical Advisory Board[7] によって公開されている．

を持たない最もシンプルなフリップフロップのみを対象としているように見受けられる．シミュレーションで用いる WinSpice3 は商用 SPICE シミュレータである．

AutoLibGen[9] はオープンソースのキャラクタライザで，シミュレータとして ngspice を利用している．Composite Curren Source(CCS) モデルでタイミング情報を記述するため Non-Linear Delay Model (NLDM) よりもタイミング解析の精度を向上できる．課題は順序セルに対応していない事である．またオープンソースである事を主張しているがソースの公開場所は明示されていない．

LiChEn[10] はオープンソースのキャラクタライザであり，非同期回路向けスタンダードセルを対象としているが，同期設計向けの組み合わせセルもキャラクタライズする事が可能である．プログラムコードは C/C++で記述されたものが公開されている．課題はシミュレータが Cadence Specter であり商用である事と，順序セルのキャラクタライズを対象としていない事である．

ASCLIC[11][12] はライブラリセルのキャラクタライズを無償で行う事を目的として開発されたキャラクタライザである．**ASCLIC** は NLDM によるタイミングモデルの抽出だけでなく，Non-Linear Power Model (NLPM) における電力モデルの抽出も可能である．マルチコア環境を適切に利用できると述べられている．課題は順序セルに対応していない事である．

本研究ではオープンセルキャラクタライザについて開発を行った．本キャラクタライザは Python3 で記述されており，フリーの SPICE シミュレータである ngspice を利用する．本キャラクタライザは組み合わせセルおよび順序セルの両方を対象としている．対象となる論理関数の真理値表を追加する事で，自由にキャラクタライズする論理関数を追加する事が可能である．順序セルは最も基本的なポジティブエッジフリップフロップだけでなく，ネガティブエッジクロック，およびセットリセットの動作も対象としている．タイミング情報および電力情報を NLDM および NLPM として抽出し，産業界で広く使われている Liberty 形式で出力する事が可能である．

3. キャラクタライズの全体像

キャラクタライザへの入力として (1) シミュレーションするネットリスト，トランジスタモデル，(2) キャラクタライズ条件などを含むコマンドファイルが必要である．本キャラクタライザはこれらのファイルを読み取り，ngspice によるシミュレーションの実施，シミュレーション結果の解析，そして Liberty 形式を出力する．またゲートレベルシミュレーションを行うための Verilog-HDL シミュレーションモデルを生成する．

図 1 にキャラクタライズの内部フローを示す．最初のステージではライブラリでの共通設定を読み取る．

- ライブラリ名，
- .lib ファイル名，
- セル名のプレフィックス，サフィクス (必要があれば)，
- 電圧，電流，容量，抵抗，電力の単位，
- VDD，VSS，p-well, n-well の名前．

二番目のステージではライブラリで共通となるキャラクタライズ条件を定義する．

- PVT (process, voltage, temperature) 条件，
- キャラクタライズでの論理閾値電圧条件，
- シミュレータバイナリの場所．

三番目のステージでは，対象のセルの論理によって分岐する．

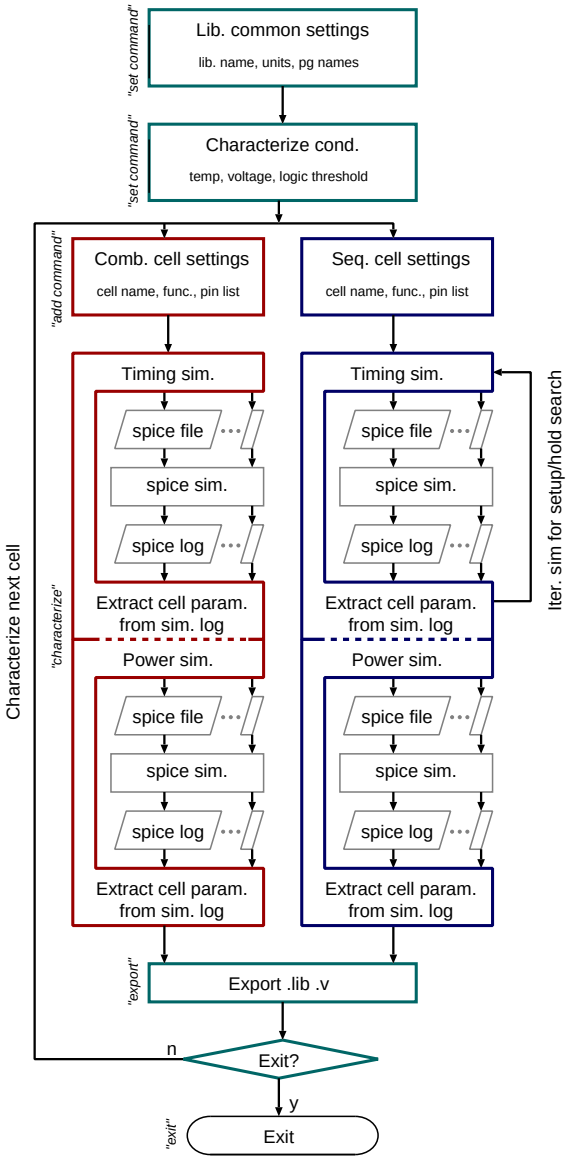


図 1 キャラクタライザの内部フロー．

個々のセルごとに異なるキャラクタライズ条件を設定する．

- SPICE のサブサーキット名，
- 論理関数，
- 入出力ピン名，
- 入力負荷，出力容量，
- シミュレーション時の各種時間刻み幅，
- Verilog-HDL として出力する論理式．

順序セルの場合はさらに追加で設定が必要となる．

- クロックピン名，
- セット・リセットピン名 (必要であれば)，
- セル内部の記憶素子名．

すべての設定を終えると，キャラクタライザは SPICE ファイルを生成し，シミュレーションの実行，実行結果の解析を行う．シミュレーションは 2 段階に分けて行われる．最初に遅延シミュレーションを行い，伝搬遅延，遷移遅延，波形遷移の開始終了時刻を抽出する．次に，電力シミュレーションを行い，動的電力とリーク電力を抽出する．組み合わせセルの場合はすべての取りうる入力パターンに対し入力遷移スリューおよび出力負荷容量を変化させながら遅延および電力情報を抽出する．順序セルの場合はセットアップ時間およびホールド時間を探査が必要となる．

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)
```

4. キャラクタライズの方法

提案キャラクタライザが実施するタイミング情報および電力情報の抽出方法について述べる．本キャラクタライザは基本的には Liberty Users Guide[5] に示された方法に則ってキャラクタライズを行う．

4.1 論理関数の定義

シミュレーションを正常に実施するためには，キャラクタライザは対象のセルの論理関数を事前に知り，その論理関数を満たす入力波形の生成と，出力波形の評価を行う必要がある．提案キャラクタライザは入力値と出力期待値の関係を真理値表の形式で内部に保持し，論理関数に応じて真理値表に沿ったシミュレーションファイルを生成する．

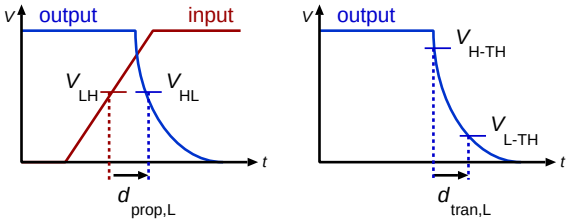
リスト 1 に，NAND2 を対象とした真理値表の内部表現を示す．NAND2 は 2 入力であるから 4 行の表現となる．この表において “0” および “1” は固定値で，“01” および “10” は VSS から VDD へ，もしくは VDD から VSS へ遷移する入出力となる．例えば 1 行目において入力 **in0** は “01” であり，VSS から VDD へ遷移する．入力 **in1** は “1” であるから VDD で固定である．出力 **out** は “10” であるから，VDD から VSS へ遷移する事が期待されている．

リスト 2 に，ポジティブエッジクロックを持つ，非同期セットリセット付き Flip-Flop を対象とした真理値表の内部表現を示す．本キャラクタライザでは本 Flip-Flop の事を “DFF_PCPU_NRNS” と定義している．本 Flip-Flop の場合，真理値表のうち 2 行はデータ入力に対応し，次の 2 行はセットおよびリセット動作に対応している．クロック信号は “0101” となっており，1 度のシミュレーション内で 2 回クロックの遷移を行う；1 度目のクロックで初期値を設定し，2 度目のクロックで遅延シミュレーションおよび電力シミュレーションを行う．

本キャラクタライザは事前に論理関数と対応する真理値表を登録する必要があるため，未定義の論理関数が入力された場合はキャラクタライズを止めワーニングを出力する仕様となっている．登録されている論理関数を表 1 に示す．

表 1 登録されている論理関数.

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



(a) Propagation delay　(b) Transient delay

図 2 伝搬遅延と遷移遅延の定義.

4.2 タイミング情報抽出

4.2.1 タイミング情報抽出における共通設定

タイミング情報抽出においては，伝搬遅延と遷移遅延をシミュレーションによって抽出する．伝搬遅延 ($d_{\mathrm{prop,L(H)}}$) は入力が電源電圧の半分 ($V_{\mathrm{LH(HL)}}$) から出力が電源電圧の半分 ($V_{\mathrm{HL(LH)}}$) まで遷移する時間であり，遷移遅延 ($d_{\mathrm{tran,L(H)}}$) は出力が論理 High(Low)($V_{\mathrm{H(L)-TH}}$) から論理 Low(High)($V_{\mathrm{L(H)-TH}}$) まで遷移する時間であり，これらは図 2 に示されたとおりである[*2]．遅延シミュレーションでは，同時に入力の遷移開始時刻 (t_{start}) と出力の遷移終了時刻 (t_{end}) を計算し，次の電力シミュレーションでの積分に利用する．図 3(a) に遅延シミュレーションにおける “.measure” 文で評価されるパラメータをまとめている．

SPICE シミュレーションにおいて，シミュレーションの時間刻み幅とシミュレーション終了時刻を適切に設定する事はシミュレーションの精度と速度に大きく影響する．特に ngspice は時間刻み幅の自動設定機能が弱い課題がある．本キャラクタライザにおいては，時間刻み幅とシミュレーション終了時刻は設計者が外部から設定する事が可能であり，もしくは自動設定を指示する事で時間刻み幅とシミュレーション終了時刻をを入力遷移時間のインデックスから自動的に設定する事も可能である．回路シミュレーションにおいては SPICE シミュレータはバッチモードで動作する事を想定しており，すべての評価項目は “.measure” 文によって評価された後自動的に終了する．

SPICE への入力ファイルは実行ディレクトリ内に生成される．設計者は必要に応じて生成された SPICE ファイルを確認する事が可能である．もしくは SPICE ファイル中の “.control” 文を有効にする事で，ngspice において波形を確認する事も可能である．

[*2] これらの閾値は外部からコマンドで変更可能である．

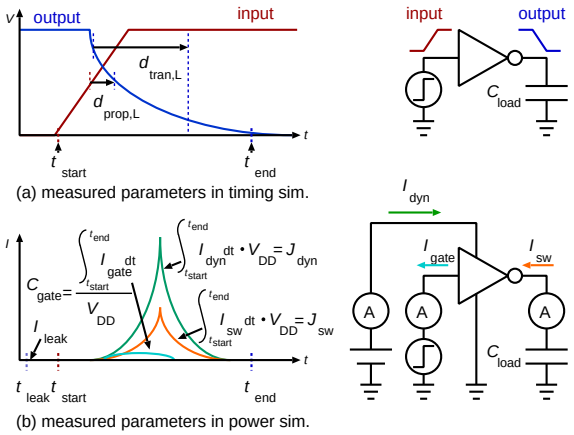


図 3 “.measure” による遅延とエネルギーの評価方法.

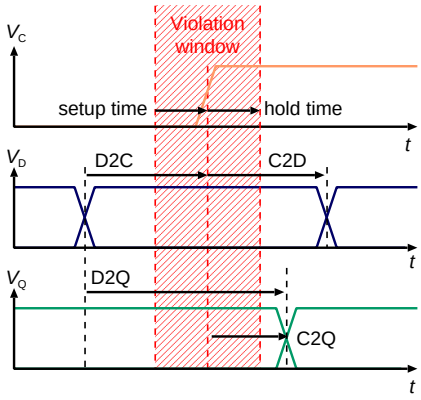


図 4 順序セルにおける遅延の定義.

4.2.2 順序セルにおける Setup/Hold 探査

順序セルはその遅延性能を表す様々な指標があり，それらを図 4 にまとめている．例としてポジティブエッジフリップフロップについて取り上げる，データ入力 (*D*)，クロック入力 (*C*)，データ出力 (*Q*) を持つ物とする．*C*-to-*Q* (C2Q) 遅延はクロック入力から出力までの遅延である．*D*-to-*C* (D2C) 遅延はデータ入力からクロックの立ち上がり前まで時間差である．フリップフロップの正常動作にはクロック入力に対しデータ入力が事前に安定している必要があり，この時間差をセットアップ時間と呼ぶ．*C*-to-*D* (C2D) 遅延はクロックの立ち上がり後データ入力が変化を開始する時間差である．フリップフロップの正常動作にはクロック入力後データ入力が安定している必要があり，この時間差をセットアップ時間と呼ぶ．*D*-to-*Q* (D2Q) 遅延はデータ入力から出力までの遅延であり，D2C 遅延と C2Q 遅延の和となる．

セットアップ時間とホールド時間共に D2C 遅延および C2D 遅延の最小値から求まるため，複数回のシミュレーションを行い最小値を探査する必要がある．本論文では最小 D2C 遅延となる D2C 遅延でもってセットアップ時間として定義する [14]．図 5 に，セットアップ時間とホールド時間の探査法について示す．D2C 遅延を徐々に厳しくしながら D2Q 遅延を調べ，それが最小となる D2C 遅延をつまりセットアップ時間とする．求まったセットアップ時間の元で C2D 遅延を徐々に厳しくし繰り返し，正常動作する最小の C2D 遅延をホールド時間として定義する．

SPICE シミュレーションは計算時間がかかる欠点がある一方で，キャラクタライズ条件が複数ある事から並列実行の実現は時間短縮に有効である．現在のキャラクタライザはすべての処理を逐次実行しているが，並列化の有効活用は今後の課題である．

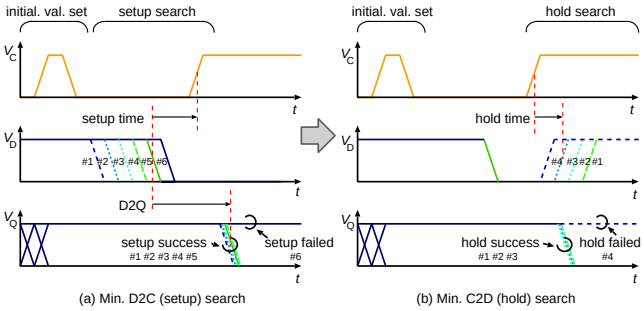


図 5 Setup/Hold 探査.

表 2 キャラクタライズ条件

キャラクタライザ	提案ツール		SiliconSmart
シミュレータ	ngspice	hspice	hspice
オプション	default	default	default
# of threads	1	1	32
プロセス	Commercial 180-nm w/ small modification	Commercial 180-nm	
入力波形	ランプ波形		
プロセス条件	Typical		
電圧	Nominal (1.8 V)		
温度	25°C		
入力スロープ	0.01 ns, 0.1 ns, 1.0 ns		
出力負荷容量	0.1 pF, 0.7 pF, 4.9 pF		
論理ハイ/ロウ電圧	1.44 V, 0.36 V (80%,20%)		
論理閾値	0.9 V, 0.9 V (50%)		
組み合わせセル DUT	Inverter		
順序セル DUT	Flip-Flop w/ pos. edge clock		

4.3 電力情報抽出

電力情報抽出ではセルのトグルあたりのエネルギー[*3] とリーク電力を抽出する．電源電圧源と出力負荷に電流 (I_{dyn}，I_{sw}) を時刻 t_{start} から t_{end} の間積分し，エネルギー (E_{dyn}, E_{sw}) に変換する．消費エネルギーは E_{dyn} から E_{sw} を差し引き計算する．リーク電力については，シミュレーション開始直後のリーク電流 I_{leak} すべてのシミュレーションパターンについて計算し，それらの平均値からセルのリーク電力を計算する．[*4] ゲート容量はゲートへの入力電流 (I_{gate}) の積分値を電源電圧で割る事で計算している．これらの定義は組み合わせセルおよび順序セルの両方で同じである．図 3(b) にシミュレーションにて利用されている “.measure” パラメータを示す．

5. 実験結果

提案キャラクタライザは Python3 言語によって設計され，シミュレータとしては ngspice を採用している．設計実験では商用 180-nm におけるセルのネットリストを利用した．ファウンドリが提供するトランジスタモデルは Synopsys hspice を想定しており，ngspice で利用するためにトランジスタモデルに変更を加えている．Synopsys SiliconSmart の結果を真値とし，我々の提案するキャラクタライザの性能を評価する．なお SiliconSmart はシミュレータとして hspice を利用するため，シミュレータの違いを比較するため我々のキャラクタライザは ngspice だけでなく

[*3] Liberty フォーマットにおける “power table” は電力ではなくエネルギーをルックアップテーブルとして保持している．これは Synopsys Liberty User Guide[5] の定義である．

[*4] Liberty では入力電圧依存性を考慮したリーク電力のモデルを生成する事も可能であるが，すべての可能な入力パターンすべてに対してシミュレーションを行う必要があるため，今回はシミュレーションを行った入力パターンの平均値を採用している．

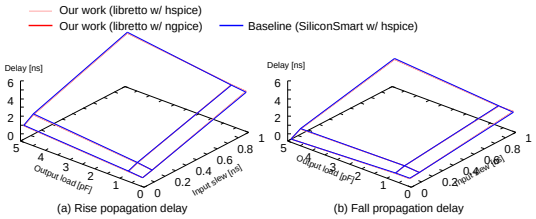


図 6 Inverter の伝搬遅延.

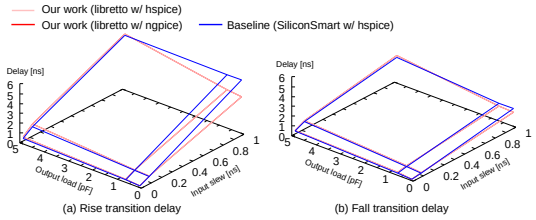


図 7 Inverter の遷移遅延.

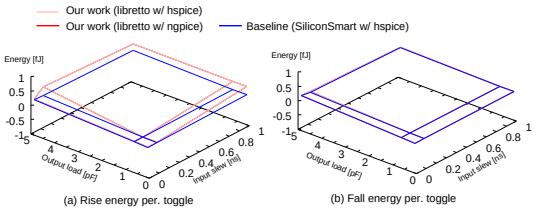


図 8 Inverter の内部消費エネルギー.

表 3 Inverter の入力容量とリーク電力.

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

hspice を利用する事も可能である．表 2 に各キャラクタライザの基本設定を示す．Inverter を組み合わせセルの代表とし，立ち上がりエッジフリップフロップを順序セルの代表として，その遅延およびエネルギーの比較を行う．

5.1 組み合わせセルの特性抽出

Inverter を対象に，その遅延，消費エネルギー，リーク電力および入力容量を評価した．伝搬遅延，遷移遅延，トグルあたりの消費エネルギーを図 6,7,8 に示す．我々のキャラクタライザは hspice(ピンク) および ngspice(赤) の両方の結果が存在するが，両者は同じ結果となっている．一方で SiliconSmart とは異なる結果である事から，大きな違いはキャラクタライザ内部のアルゴリズムにある事がわかる．個々の項目に注目すると，伝搬遅延はおおよそ一致しているが，遷移遅延は入力スロープが大きい領域において大きな誤差が現れている．消費エネルギーについては出力立ち上がりの場合に悪化しており，提案キャラクタライザは悲観的な見積もりをしている．伝搬遅延および遷移遅延の最悪誤差は，どちらも立ち下がりの時に観測されそれぞれ 0.50%, 1.44% であった．消費エネルギーについては出力立ち上がり時に最大 418%の見積もり誤差となった．表 3 にリーク電力と入力容量の比較結果を乗せる．リーク電力はよく合致しており，入力容量は 10%の誤差であった．

5.2 順序セルの特性抽出

順序セルのキャラクタライズの例として，フリップフロップの C2Q 伝搬遅延，C2Q 遷移遅延，セットアップ時間，ホールド時間，消費エネルギーについて図 10,11,12,13 に示す．伝搬遅延および遷移遅延は出力立ち下がりでかつ入力スリューと出力容量が大きい条件において大きな差が発生している．伝搬遅延の最大誤差は共に立ち下がりの時においてそれぞれ 1125%および 2407% であった．

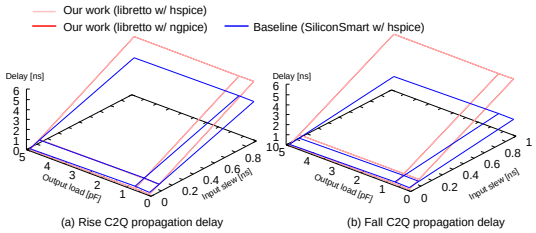


図 9 フリップフロップの C2Q 伝搬遅延.

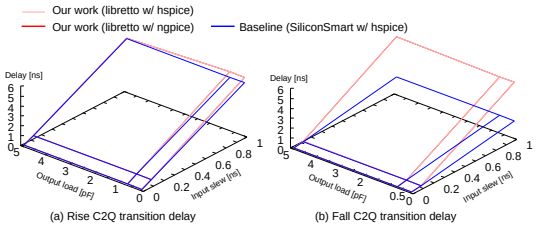


図 10 フリップフロップの C2Q 遷移遅延.

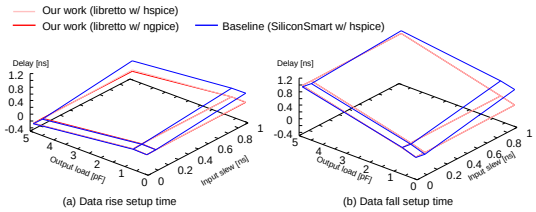


図 11 フリップフロップのセットアップ時間.

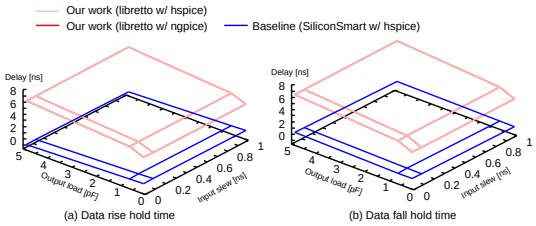


図 12 フリップフロップのホールド時間.

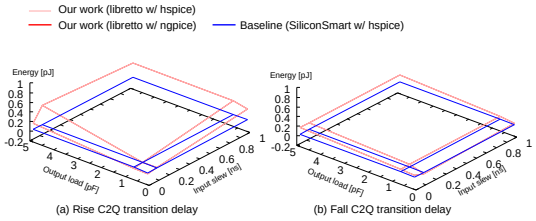


図 13 フリップフロップの内部消費エネルギー.

図 11，12 より提案キャラクタライザは相対的に小さいセットアップ時間と大きいホールド時間を報告している事がわかる．この理由としてセットアップ時間とホールド時間の相互依存性が考えられる：セットアップ時間 (もしくはホールド時間) を厳しくすると，より長いホールド時間 (もしくはセットアップ時間) が必要になるということである [15], [16], [17] ．提案キャラクタライザでは，最初にセットアップ時間の最小を探査するため結果として必要なホールド時間が長くなったと考えられる．商用のキャラクタライザ，例えば SiliconSmart はセットアップ時間とホールド時間の探査を同時に行うことで，セットアップ時間とホールド時間の相互依存性に起因する悲観性を取り除いている．図 13 に消費エネルギーの比較結果を示す．提案キャラクタライザは悲観的な消費エネルギーの見積もり結果を示しており，最大誤差は立ち上がり立ち下がりでそれぞれ 889%，1915%であった．ただし

表 4 実験環境.

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

表 5 キャラクタライズの実行時間比較.

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×

セットアップ時間とホールド時間の制約が異なる事も誤差の要因であると考えられる.

5.3 実行時間評価

本キャラクタライザは CentOS 7 の稼働する AMD Ryzen マシン上で実行された，実行環境の詳細を表 4 に示す．また実行時間を表 5 に示す．2 つのセルのキャラクタライズにおいて提案キャラクタライザと ngspice を組み合わせると 2.5 時間の実行時間が必要であった．シミュレータを hspice に変更することで実行時間を 2.26 倍高速化できた．SiliconSmart では hspice を利用し 32-thread で実行した結果 215 倍の速度差となった．現在のキャラクタライザは逐次処理となっている．簡単な勾配法によるセットアップ時間，ホールド時間の探査となっており，シミュレーションの回数も SiliconSmart に比べて非常に多い問題がある．現在の実装では遅延と消費エネルギーを別のシミュレーションにて行っている，これは ngspice が".measure"文の入れ子構造をサポートしておらず，消費エネルギーの積分開始および終了時刻を得るために事前の遅延シミュレーションを必要としているためである．また ngspice そのものも商用シミュレータに比べると速度が劣る問題もある．将来的には並列性を利用したキャラクタライズの高速化について取り組む予定である.

6. 結論

本論文ではオープンソース EDA としてキャラクタライザの開発について報告した．本キャラクタライザは組み合わせセルおよび順序セルの両方をキャラクタライズする事ができる．タイミングモデルおよび電力モデルの両方を抽出する事が可能であり，本結果を利用する事で現実的なセル遅延を考慮した STA，論理合成，配置配線を実現する．フリーでオープンなキャラクタライザがオープンソース VLSI 設計の助けになれば幸いである．プログラムコードは Github にて公開している.

https://github.com/snishizawa/libretto

課題としては性能と精度があげられる．シミュレーションの並列実行によって実行時間の短縮が可能である．入力波形を単純なランプ波形から現実的な波形ドライバモデルへ変更する事，またセットアップ時間とホールド時間の相互依存性を考慮したキャラクタライズを実現することで精度向上が可能である.

謝辞 本研究は株式会社ロジックリサーチ (190402，210104) および福岡大学 (197105，217301) による支援によっておこなわれた．設計実験は東京大学大規模集積システム設計教育研究センターを通しシノプシス株式会社の協力で行われた.

参考文献

[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, et al.,"Real Silicon Using Open-Source EDA," IEEE Design and Test, vol.38, no.2, pp.38–44, 2021.
[4] 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, 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," 2008 International Conference on Electronic Design, 2008.
[10] M.T. Moreira, et al., "LiChEn: Automated electrical characterization of asynchronous standard cell libraries,"Euromicro Conference on Digital System Design, pp.933–940, 2013.
[11] M.S.I. Bin Hussin, et al., "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, pp.93–97, 2018.
[12] C.H. Oliveira, et al., "ASCEnD-FreePDK45: An open source standard cell library for asynchronous design," 2016 IEEE International Conference on Electronics, Circuits and Systems, pp.652–655, 2017.
[13] A. Beg, et al., "A collaborative platform for facilitating standard cell characterization," Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design, pp.202–206, 2013.
[14] N.H.E. Weste and D.M. Harris, CMOS VLSI Design, 4 ed., Addison Wesley, 2010.
[15] E. Salman, et al., "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, et al., "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, et al., "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.