

修士論文

コンパクトな高速低消費電力 ライブラリーの研究

2000 年 2 月 4 日提出

指導教官
桜井貴康 教授

東京大学大学院工学系研究科
電子工学専攻
学籍番号 8 6 5 0 7

ゲン ドウック ミン
Nguyen Minh Duc

目次

第1章	研究背景と目的	1
1.1.	自動論理合成による LSI の設計	1
1.2.	コンパクトなライブラリーの必要性	2
1.3.	DEPLETION 型 CMOS 混在のライブラリー	3
1.4.	本論文の構成	5
第2章	コンパクトなライブラリー	6
2.1.	従来のライブラリーの構成	6
2.2.	コンパクトなライブラリーの生成アルゴリズム	8
2.3.	ライブラリーの性能評価	11
2.4.	セル数とライブラリーの性能の関係	11
2.5.	最適なコンパクトライブラリー	15
2.6.	実際の設計への応用	17
2.7.	配線遅延による影響の検証	18
第3章	低電源電圧 DEPLETION 型 CMOS の最低動作閾値電圧	23
3.1.	DEPLETION 型 CMOS 混在のライブラリー	23
3.2.	最低動作閾値電圧の定義	23
3.3.	最低動作閾値電圧の理論計算	25
3.4.	SPICE シミュレーションによる検証	27
第4章	結論	29
1.	ベンチマーク回路のフォーマット	30
2.	変換プログラム	31

第 1 章 研究背景と目的

1.1. 自動論理合成による LSI の設計

プロセス技術の向上によって 1 つのチップに入るトランジスタ数が年々増加している。これに伴って集積回路がますます複雑化していくため、設計期間が長くなるばかりか設計ミスも大発生しやすくなる。いかにして設計を自動化し、人手による作業を減らすかが、全体のプロジェクトを成功へ導くかぎを握ると言っても過言ではない。

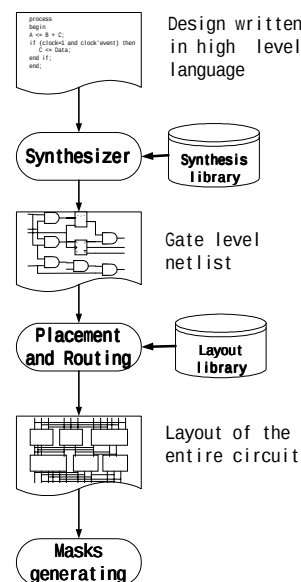


Fig. 1 Cell-based design methodology

スタンダードセル手法は ASIC などの自動設計に最もよく使われている方法である[1]。この方法は設計しようとする回路の基本部品となるスタンダードセルをあらかじめ用意しておく。各セルのレイアウトや遅延などに関する情報はライブラリー(セルライブラリーという)の形で設計者に提供される。設計者はハードウェア記述言語で回路の機能を記述し、機能シミュレータを用いて回路の動作を検証した後に、論理合成ツールを使ってゲートレベルのネットリストに変換する。配置配線ツールはセル間の接続情報を持ったネットリストと、個々のセルのレイアウト情報を含んだレイアウトライブラリーに基づいて、総面積と配線長が最も小さくなるようにセルを適切な位置に配置し、回路のレイアウトを自動的に生成する (Fig. 1)。

このように、人為的なミスを犯しやすく、手間も大変かった物理レベルでの設計はすべて計算機によって自動的に行われるため、設計期間が大幅に短縮され、設計者が本質的な機能設計

¹ これは無制約の場合である。Timing-driven layout を行う時は、面積と配線が最適化されるのではなく、上位の論理合成ツールなどから渡されるタイミング制約が満たされるようにレイアウトが生成される。

に専念できるようになる。また、セルのレイアウトを行い、遅延や面積などを抽出しライブラリーを作成する作業は一世代のテクノロジーに対して1回だけ行えば、何回でも様々なデザインに活用できるので NR (Non-Recurring Expense) コストを削減できるというメリットがある。

1.2. コンパクトなライブラリーの必要性

一般的なセルライブラリーは数百から千近くの種類のセルより構成されている。インバータ、AND、OR、NAND、NOR、XOR など基本的な論理機能を持ったゲートから、マルチプレックスー、AND-OR-INVERTER や OR-AND-INVERTER などの複合ゲート、全加算器、デコーダ、エンコーダ、パリティチェックなど機能が比較的複雑なものまである。論理合成ツールがすべてのパスの遅延や総面積などを計算して回路を最適化できるように、各々のセルに対して多くのデータをライブラリーに登録する必要がある。使用する計算モデルとセルにもよるが、入力端子の容量、出力端子の駆動力、セルの面積、セルの機能を表す入出力間の論理式（順序セルの場合は状態テーブル）の他に、遅延を計算するのに必要なフィッティングパラメータなどがある。Table 1 に代表的なパラメータを示す。

Table 1 Typical parameters used in a standard cell library

Functionality	$O = f(I_1, I_2, I_3, \dots, I_n)$	
Cell area	$S [\mu\text{m}^2]$	
Input capacitance	$C_1, C_2, \dots, C_n[\text{fF}]$	
Driving ability	Falling resistance $R_f[\text{k}\Omega]$	
	Rising resistance $R_r[\text{k}\Omega]$	
Delay (H-L & L-H) parameters for every input-output pair	Intrinsic delay	$D_i[\text{ps}]$
	Slope delay	D_s
	Wire delay	$D_w[\text{ps/m}]$

このように、各々のセルに対して十種類以上のパラメータをレイアウトから抽出する必要がある。セルが数百個以上あるため、ライブラリーを新しくするのは非常に時間とコストがかかる。数十人が共同で作成しても数ヶ月かかるのが普通である[2]。今までは半導体メーカーがライブラリーを作成し、それを多数のユーザーに提供し、ライブラリーを何回も再利用することによってそのコストを緩和させてきた。また、プロセスが2年に一回のペースで更新するならばライブラリーの作成時間もそれほど問題にはならない。

一方、プロセス技術は大変めざましい発展を示している。テクノロジーの世代交代は年々早まっている。以前なら3年ないし2年に一度だったが、最近では1年に一度、または6ヶ月に一度のペースで進んでいる[3]。微細化のみならず、LSIの低消費電力を実現するためにMTCMOS、VTCMOS、DTMOSやpartially depleted SOIなどのような全く新しいテクノロジーも数多く開発されている[8]。新しいテクノロジーに移行する際にライブラリーを作り直さなければならない。ライブラリーを作るのに数ヶ月もかかるため、プロセスが安定してから実際の設計環境が整うまで大きな遅れがある。ライブラリーがないために、設計を最新のプロセスでインプリメントできないケースもある。場合によってはライブラリーの作成作業がテクノロジーの世代交代よりも遅いと、ライブラリーができた時にすでに次の世代に入るという可能性も出てくる。

ライブラリーの生成時間とコストを減らすために、様々なライブラリー自動生成システムが提案されている[5][6][7]。このようなシステムではセルの種類を予め定めておき、全セルのテンプレートを用意する。セルの機能やレイアウトの相対的な座標情報を表すパラメータをすべてテンプレートに持たせる。こうすることによって、テンプレートのパラメータを変更（スケーリン

グ)するだけでいつでも新しいレイアウトを生成できる (Fig. 2)。RC など寄生パラメータのレイアウトからの抽出も自動化できるので人手による作業がほとんどなく、生成時間とコストを大幅に削減できる。

このようなライブラリー自動生成システムはセルのレイアウトスタイルに変更がないことを前提としているため、設計ルールだけが変った時には非常に有効であるが、MTCMOS や VTCMOS などのように、今までと全く違ったレイアウトを必要とするテクノロジーには適用できない。このようなテクノロジーではテンプレートのパラメータを変更するだけでは新しいレイアウトを生成できない。

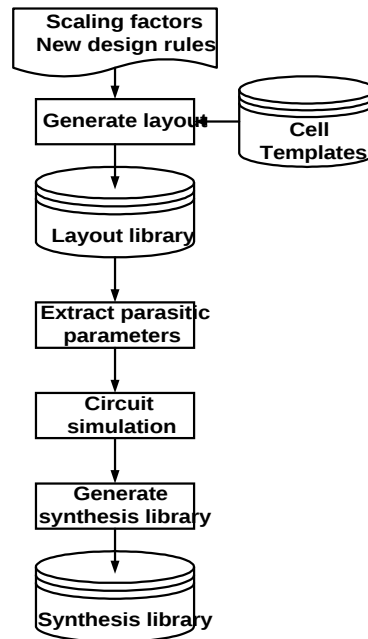


Fig. 2 Library generating system

ライブラリーの生成時間とコストを削減するための本質的な解決策はライブラリーの縮小化にある。セル数が少なく、かつ数百セルから構成される従来の大きなライブラリーと同等なパフォーマンスを持つコンパクトなライブラリーは今後の LSI の設計に不可欠であろう。本研究の第一の目的はこのようなライブラリーを求めることである。

1.3. Depletion 型 CMOS 混在のライブラリー

集積回路の消費電力の爆発的な増大に伴い、消費電力はライブラリーを設計する際に考慮しなければならないもう一つの重要な評価基準になってきた。

CMOS 回路の消費電力 P はのように表すことができる。

$$P = \alpha f C_L V_{dd}^2 + I_{leak} V_{dd} \quad (1)$$

ここで f , a , V_{dd} , C_L , I_{leak} はそれぞれ回路の動作周波数、一周期内の ON/OFF する確率、電源電圧、負荷容量および漏れ電流を表している。実際、漏れ電流 I_{leak} は極めて小さいので上式の第 2 項はほとんど無視できる。このように、消費電力は電源電圧の二乗に比例しているため、消費電力を削減する最も有効な方法は電源電圧 V_{dd} を下げることである (Fig. 3)。ここ十年間、消費電力を減らすために電源電圧は 5[V] から、3.3[V] そして 1.8[V] に下げられてきた。

一方、遅延時間は次のような式で表せる。

$$D = k \frac{Q}{I} = k \frac{V_{dd} C_L}{(V_{dd} - V_{th})^\alpha} \quad (2)$$

ここで、 k , Q , V_{th} はそれぞれ比例定数, ON/OFF する際に出力ノードが放電（充電）する電荷量および閾値電圧である。電源電圧 V_{dd} が下がるとトランジスタの駆動力が弱くなり、遅延時間が増える (Fig. 3) という問題が発生する。遅延時間が増えないようにするためには閾値電圧も電源電圧と同時に下げなければならない。電源電圧の低下に伴って閾値電圧もますます低くなっているのが現状である。将来、消費電力よりも遅延時間の方がはるかに重要なクリティカルパスなどでは閾値電圧が 0 [V] 以下の depletion 型の CMOS が使われるようになる可能性もある。

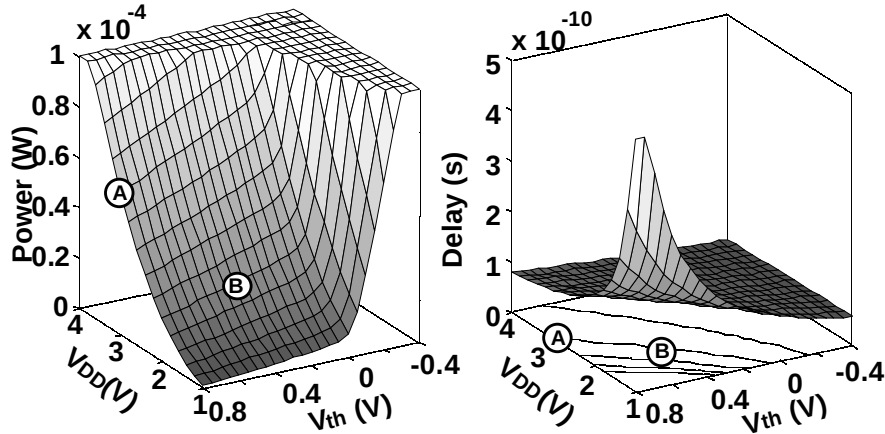


Fig. 3 Relationship between power dissipation, delay, Vdd and Vth

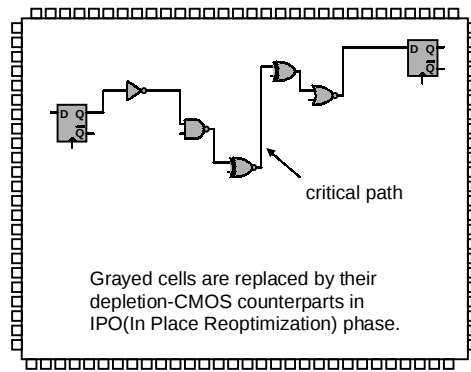


Fig. 4 Achieving higher performance by swapping cells in the critical path

高速かつ低消費電力という二つの相反な要求を満たすためにライブラリーの中に depletion 型 CMOS を使ったセルを含める必要がある。初期段階の論理合成フェーズでは通常の高閾値電圧の enhancement-CMOS セルを用いて回路を構成する。詳細な配置配線が完了し、クリティカルパスが明確になった再最適化の段階に、enhancement 型 CMOS セルを depletion 型 CMOS セルに置換えることによって高速化を図る。Depletion 型 CMOS 回路は従来の enhancement 型 CMOS に比べ、消費電力が大きい、回路の大部分は enhancement 型 CMOS で構成され、クリティカルパス上にあるごく一部だけに depletion 型 CMOS が使われる (Fig. 4) ので、全体の総消費電力はほとんど増加しない。一方、クリティカルパスの遅延は全体のパフォーマンスに直結するので、depletion 型 CMOS セルを導入することによって高速化できる。

Depletion 型 CMOS の場合は p-MOS と n-MOS が常時同時に導通しており、スイッチング特性が急峻ではないので安定性が悪い。Enhancement 型 CMOS はデジタル信号を修復する能力を持ち、入力信号が多少劣化してもゲートを通過して出て行く時は必ずフル振幅に復元される。Depletion 型 CMOS にはこのような修復能力がなく、出力信号の振幅は入力信号の振幅より小さい。閾値電圧が低くなるにつれてスイッチング特性が劣化する。閾値電圧がある一定値以下になると出力がとうとう $V_{dd}/2$ に収束し、回路はその論理機能を失ってしまう。

回路の動作を保証する意味でこの閾値電圧の下限を知ることが大変重要である。閾値電圧の下限を求め、depletion 型 CMOS セル混在ライブラリーの設計ガイドラインを設定することが本研究の第 2 の目的である。

1.4. 本論文の構成

本論文は大きく二部に分けることができる。

第 2 章でコンパクトなライブラリーについて述べる。ここでは主に論理レベルでのライブラリーの縮小化を説明する。本研究が提案したライブラリー生成 (縮小化) アルゴリズムを説明した後、得られたライブラリーと実際の設計への応用例を示す。コンパクトライブラリーを生成する時、配線の遅延と面積は仮配線モデルの形でしか反映されないが、大規模なサンプル回路に対して完全な配置配線を行うことによって配線の影響を検証する。

第 3 章では、depletion 型 CMOS 混在ライブラリーの 1 つの重要な設計ガイドラインとなる閾値電圧の下限について述べる。depletion 型 CMOS セルを導入することによって、トランジスタレベルでの高速化を行う場合、どこまで閾値電圧を下げられるかについて検討する。

最後の第 4 章で、本研究で得られた結果をまとめる。

第2章 コンパクトなライブラリー

2.1. 従来のライブラリーの構成

スタンダードセルライブラリーを構成するセルの選択に関しては確立した方法論は存在していない[4]。設計者の経験に基づき、必要に応じてセルを追加していくのが一般的なやり方である。パフォーマンスを少しでも向上させようとする狙いから、新しいテクノロジーに移行するためにライブラリーを作り直す必要が出てくる時に、新しいセルが追加されることが多い。その結果、ライブラリーはますます膨大化していく傾向にある。現在の一般的なセルライブラリーは数百から千以上のセルからなっている。

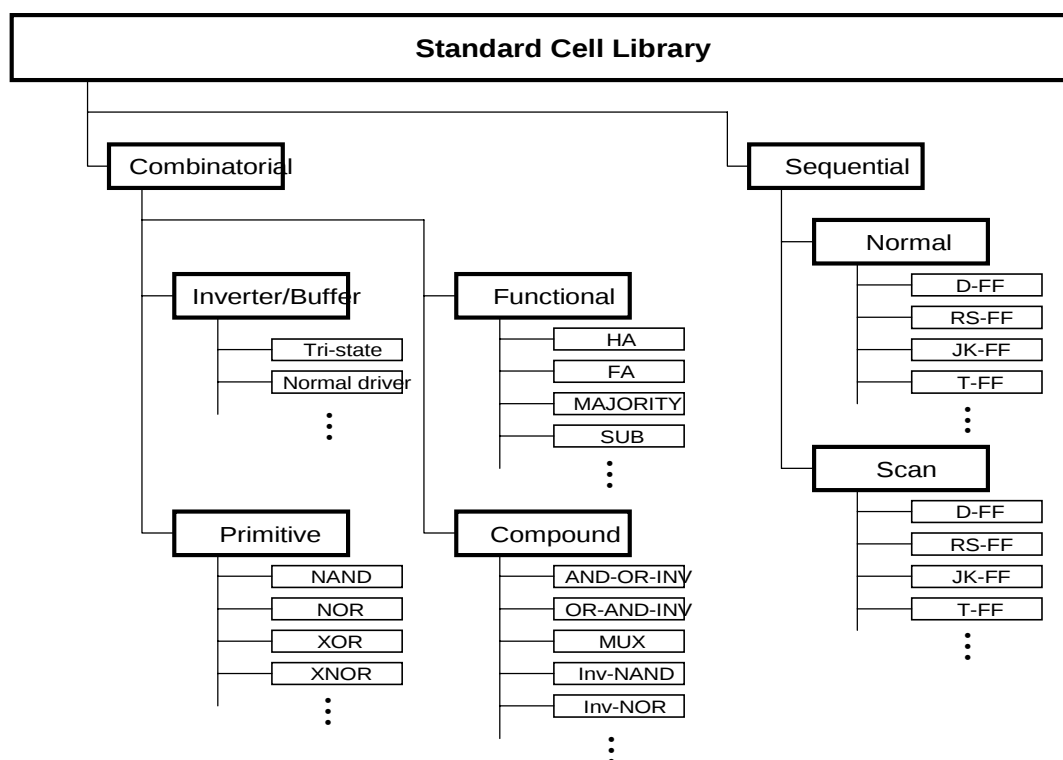


Fig. 5 Common structure of a standard cell library

Fig. 5 にセルライブラリーの一般的な構成を示す。このように同じライブラリーの中に様々な機能を持ったセルがある。またそれぞれの機能に対して、駆動力だけが異なるセルが数種類用意されるのが普通である。特にインバータとバッファは post layout optimization を行う

時に、負荷の重い配線に挿入されることが多いため、駆動力のバリエーションが一番大きい。ライブラリーにもよるが 10 種類以上のインバータ（バッファ）セルが登録されることが多い。

ライブラリーのセルは基本的に組み合わせセルと順序セルという 2 種類に分類できる。組み合わせセルは更にバッファ・インバータ類やプリミティブロジックゲート、複合ゲートなどに分けることができる。演算機能を簡単に実現できるように、半加算器や全加算器など比較的複雑な機能を持った回路が 1 つの組み合わせセルとしてライブラリーに登録されることもある。順序セルには D-FF や RS-FF などのフリップフロップがある。一部のライブラリーは D-FF のデータ入力ピンに一種の組み合わせ回路を挿入したような複雑な順序セルを持っている。またテスト容易化設計を実現するために、ほとんどのライブラリーでは、全順序セルは対応するスキャンセルとペアで作られる。

ライブラリーのセル数は一千近くにも及ぶが、すべてのセルが同じ頻度で使用されるわけではない。インバータ、バッファ、2 入力 NAND や 2 入力 NOR のように、機能が単純で、面積の小さいセルは比較的に多く使われるのに対して、入力数の大きいマルチプレクサ - や半加算器、全加算器など、機能が複雑で面積の大きいセルはめったに使われない。Fig. 6 に、Fig. 5 にあるようなセルをすべて持っているライブラリーを使って、ISCAS89 ベンチマーク[10]を論理合成した場合の各セルの平均使用頻度を示す。

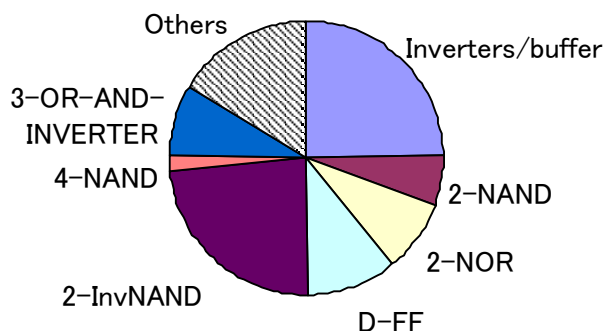


Fig. 6 Average percentage of number of cells (ISCAS89 benchmark)

このようにインバータ/バッファ類と 2-InvNAND (2 入力の内の一つが反転された 2 入力 NAND) が全体の 50% 近くを占めている。その他に、2 入力 NAND や 2 入力 NOR などプリミティブなセルが頻繁に使われることも上図からうかがえる。また、図に名前が示されている使用頻度の比較的に高いセル以外のセルは全部あわせても使用頻度の合計が高々 20% しかない。ライブラリーは非常に多くのセルを持っているが、実際に使われるのはその中の小さな一部だけである。ライブラリーが主に設計者の経験に基づいて作られるため、バージョンアップする過程に冗長な（ほとんど使用されない）セルが入り混じっていると言える。

一方、セル数を増やすば増やすほどライブラリーの性能がよくなるとは限らない。自動論理合成ツールが理想的な決定性アルゴリズムを使用し、必ず真の最適解を見けられるならば、セル数が多くなれば、選択肢も増え、よりよいパフォーマンスを達成できる。しかし、現状の論理合成ツールはいずれもヒューリスティックな非決定性アルゴリズムを使用し、真の最適解ではなく、それに近い近似最適解しか導出することができない。セル数が多すぎると探索空間が爆発的に増え、ローカルな最適点に陥りやすくなる (Fig. 7)。場合によっては、最適解を見つけられず、論理合成される回路のパフォーマンスが改善されるところか、悪くなってしまうケースもある。

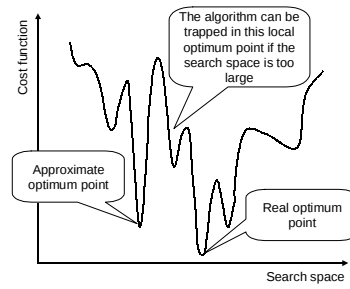


Fig. 7 Searching optimum point in a large search space

2.2. コンパクトなライブラリーの生成アルゴリズム

使用頻度の小さいセルをライブラリーから取り除けば、元のライブラリーと同等な性能を持ったコンパクトライブラリーが得られる。ここで重要なのはどのセルを取り除くかである。あるセルがライブラリーから外されると、論理合成ツールがそのセルを残りのセルの組み合わせで置換える。置換えられる部分が大きければ大きいほど回路の構造に加えられる変更が大きいため、パフォーマンスも大きく変わる。ライブラリーの冗長性や論理合成ツールのアルゴリズムの不完全性などから、セル数が少なくなるとパフォーマンスがよくなる可能性もあるが、多くの場合は多少悪くなると思われる。したがって、自動生成される回路のパフォーマンスを劣化させないためには、回路の構造をあまり変えないような取り除き方をする必要がある。

セルがライブラリーに含まれるか否かによって、回路のパフォーマンスが大きく左右される意味で、セルが回路の構造に与える影響の度合いをそのセルの重要度と見なせる。重要度の小さいセルをライブラリーから取り除くことで、セル数が少なく、しかも同等なパフォーマンスを持ったライブラリーを作ることができる。

セルが多く使われれば、明らかに回路に及ぼす影響が大きい。またセルが大きければ大きいほどその機能が複雑なので、ライブラリーから取り除いた時、より多くのセルを組み合わせで置換えなければならない。すなわち、そのセルが回路の構造（パフォーマンス）に与える影響が大きい。このように、セルの重要度（＝回路の構造に与える影響度）は、セルの使用回数とセルの面積の両方に依存する。よって、セルの使用回数にセル面積を掛けた数値＝セルの占める平均面積割合をセルの重要度として定義することができる。本研究ではこの平均値を計算するために ISCA89 というベンチマークを用いた[10]。このベンチマークは、31 個の順序回路から構成されている。これらの回路は規模が数百ゲートから 1 万ゲート以上までのものであり、機能の偏りが比較的少ないことから、論理合成アルゴリズムや配置配線アルゴリズムなどの評価によく用いられる。Table 2 に ISCAS89 ベンチマークの各回路の規模、入力端子数などを示す。

¹ 本論文で言うパフォーマンスとは遅延、面積および消費電力のいずれかである。自動生成される回路の遅延、面積および消費電力が小さければ、ライブラリーのパフォーマンスがよいということになる。

Table 2 ISCAS89 benchmark circuit characteristics [10]

Circuit name	# of inputs	# of outputs	# of D flip-flops	# of combinatorial gates
s27	4	1	3	10
s208	11	2	8	96
s298	3	6	14	119
s344	9	11	15	160
s349	9	11	15	161
s382	3	6	21	158
s386	7	7	6	159
s400	3	6	21	162
s420	19	2	16	196
s444	3	6	21	181
s510	9	7	6	211
s526n	3	6	21	194
s526	3	6	21	193
s641	35	24	19	379
s713	35	23	19	393
s820	18	19	5	289
s832	18	19	5	287
s838	35	2	32	390
s953	16	23	29	395
s1196	14	14	18	529
s1238	14	14	18	508
s1423	17	5	74	657
s1488	8	19	6	653
s1494	8	19	6	647
s5378	35	49	179	2779
s9234	19	22	228	5597
s13207	31	121	669	7951
s15850	14	87	597	9772
s35932	35	320	1728	16065
s38417	28	106	1636	22179
s38584	12	278	1452	19253

セルの重要度 $\bar{a}$ は次式のように定義される。

$$\bar{a} = \frac{1}{N_B} \sum_{i=1}^{N_B} \frac{f_i \cdot a}{A_i} \quad (3)$$

ここで、 N_B はベンチマーク回路の数 (ISCAS89 の場合は 31)、 A_i は i 番目のベンチマーク回路の総面積、 f_i はこのベンチマーク回路におけるセルの使用回数、 a はセルの面積である。

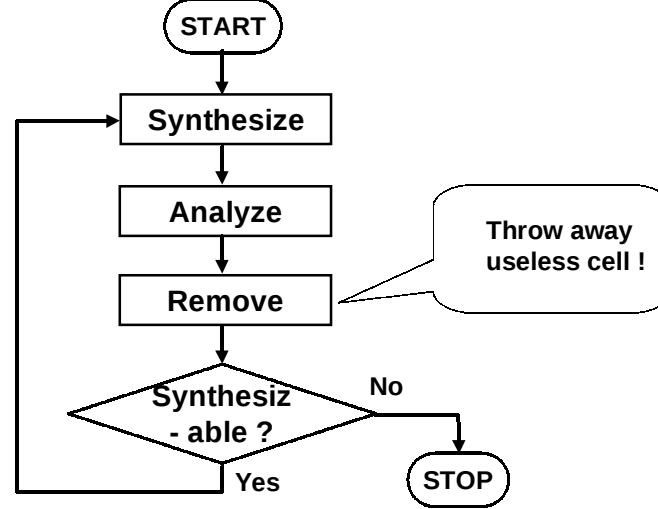


Fig. 8 Flow of the proposed library generating algorithm

Fig. 8 はコンパクトライブラリーの生成アルゴリズムのフロー図である。既存のライブラリーから比較的に重要でないセルを順番に取り除き、セル数の小さい新しいライブラリーを生成するこのアルゴリズムは次のようなステップに分けることができる。

1. 全ベンチマーク回路 (N_B 個) を論理合成する。
2. 得られたゲートレベルのネットリストを解析して、上式の定義に基づきライブラリー中にある全セルの重要度 $\bar{a}$ を計算する。
3. 最も重要度の小さい s 個のセルをライブラリーから取り除く。これでセル数が s 個少ないライブラリーが得られる。 s を小さく取った方が、セル数に対するライブラリーの性能の変化を細かく見ることができるが、計算時間は長くなる。実際ライブラリーの性能は、セル数を減らしていく場合、セル数が大きい (数百個以上) 時はほとんど変わらないが、セル数が少なくなるとつれてその変化が著しくなるので、セル数にあわせて s の値を動的に変えることによって高速化を図る。
4. 得られたライブラリーで論理合成できるかどうかチェックする。論理合成可能ならばステップ 1 に戻る。そうでなければ終了する。このステップが必要なのは、論理合成するためにはライブラリーがある最小限のセルセットを含んでいなければならないからである。論理合成ツールにもよるが、一般的にインバーター、D-FF および 2 入力 NAND があれば、論理合成は可能である。ステップ 1、2 および 3 が繰り返されるにつれ、ライブラリーのセル数が減っていき、最小限のセルセット中のどれかがライブラリーから削除されると計算が完了する。

上記のライブラリー生成アルゴリズムが終了した時、一連のライブラリーが得られる。

$$L > L_1 > L_2 > \dots > L_m \quad (4)$$

ここで、 L は最初のライブラリー、 L_1 は L から s セルを取り除くことによって得られるライブラリーである。 L_2 は L_1 から s セルを取り除いた後に得られるライブラリーである。 m はループ(ステップ 1 - 2 - 3 - 4)が繰り返される回数である。この中で、 L は一番セル数が大きく、 L_m は一番セル数が少ない。

各々のライブラリーのセル数と、それを使って論理合成する時の回路の性能との関係を調べれば、セル数が少なく、かつ性能のよいライブラリーを見つけることができる。

なお、上記のアルゴリズムは Perl で実装された。プログラムの詳細な説明は付録 B に掲載してある。また、ISCAS89 のベンチマーク回路は独自の形式で記述されているため、論理合成ツールが理解できる形式に変換しなければならない。フォーマットの変換の詳細は付録 A に載せた。

2.3. ライブラリーの性能評価

ライブラリーの性能(パフォーマンス)は、それを使って自動論理合成された回路の平均遅延、面積および消費電力の 3 つの指標で評価することができる。本研究では、ベンチマーク回路の平均相対パフォーマンスをライブラリーの性能と定義する。

前節の m 個のライブラリー中の i 番目のライブラリー L_i の性能 $\overline{p_i}$ は

$$\overline{p_i} = \frac{1}{N_B} \sum_{j=1}^{N_B} \frac{p_{ij}}{p_j} \quad (5)$$

で与えられる。

ここで、 N_B は前節と同じく、ベンチマーク回路数を表す。 p_j は、元のライブラリー L を使った時の j 番目のベンチマーク回路のパフォーマンス(遅延、面積または消費電力)で、 p_{ij} は i 番目のライブラリーを使った時の同じベンチマーク回路のパフォーマンスである。

2.4. セル数とライブラリーの性能の関係

本節では、2.2 述べたアルゴリズムでスタンダードセルライブラリーを縮小化する場合、平均遅延、面積および消費電力がどのように変化するかについて述べる。

3 つの異なるライブラリーをスターティングライブラリーとしてを使用した。この 3 つのライブラリーはいずれも実際の設計に使用されているもので、異なる設計ルールを持ち、異なるセルから構成されている。Table 3 にこれらのライブラリーの設計ルールとセル数を示す。

Table 3 Libraries used in this research

Library	Feature size [μm]	Number of cells
A	1.5	~ 400
B	0.6	~ 400
C	0.25	~ 400

セル数と平均相対遅延、面積および消費電力との関係を表したグラフを Fig. 9、Fig. 10(13 ページ)と Fig. 11(14 ページ)に示す。各々の図には 3 つのライブラリーに対応する 3 つの折線がある。図の横軸はライブラリーのセル数を表し、縦軸は 2.3 で定義された平均パフォーマンスである。図中の折り線グラフの右端はセル数が 150 前後の点にあるが、これはライブラリーの中に 400 セルがあっても、実際は約 150 セルしか使われていないことを意味する。

セル数を減らしライブラリーを縮小化しても遅延時間がほとんど変化しないことがFig. 9 から分かる。遅延時間は、セル数が 20 以下になって始めて徐々に増加し始め、セル数が 10 の時でも 10%前後しか増えない。セル数が 10 以下になると遅延時間の増加が大きくなる。

平均面積 (Fig. 10) は遅延に比べてはっきりした変化を呈しているが、セル数が 40 以上の時はセル数に関係なくほとんど一定である。面積はセル数が 20 以下になると大きく増える。

消費電力 (Fig. 11) は面積と同様な変化を示す。消費電力は、セル数が 40 以上の時はセル数とほとんど無関係であり、セル数が 20 以下になってはじめて著しく増え始める。

ライブラリーC の場合はセル数が小さくなると逆に面積と消費電力が減少する。その理由は、2.1 でも述べたように、論理合成ツールのアルゴリズムが真の最適解を見つけられないことにあると考えられる。セル数を減らすと探索空間が狭くなり、ローカルな極小点に陥る可能性が少なくなるため、よりよい結果が得られるのである。

論理合成時間はライブラリーの性能と無関係であるが、大規模な回路の合成時間が非常に長いので、設計時間を短縮する意味で重要な指標である。セル数を変えた時の平均論理合成時間 (2.3 の平均パフォーマンスと同様な方法で計算した) をFig. 12 (14ページ) に示す。遅延時間、面積および消費電力と違って、合成時間の変化は大変不規則的であり、セル数を減らすと減ったり増えたりする。しかし、全体的に見てセル数が小さくなれば (数十程度) 論理合成時間も短縮されることが分かる。

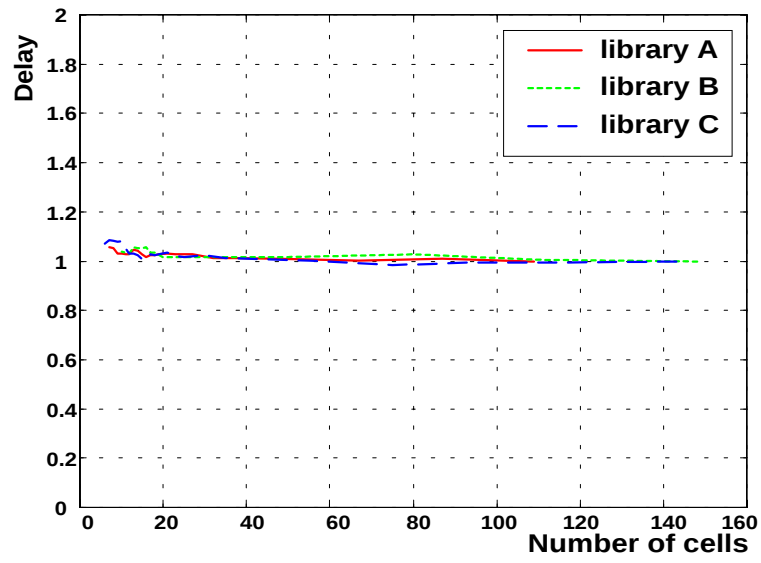


Fig. 9 Average relative delay vs. number of cells

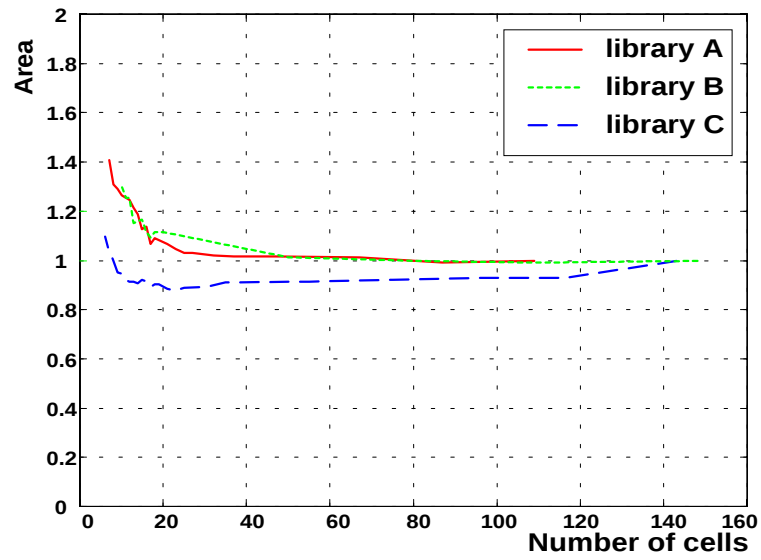


Fig. 10 Average relative area vs. number of cells

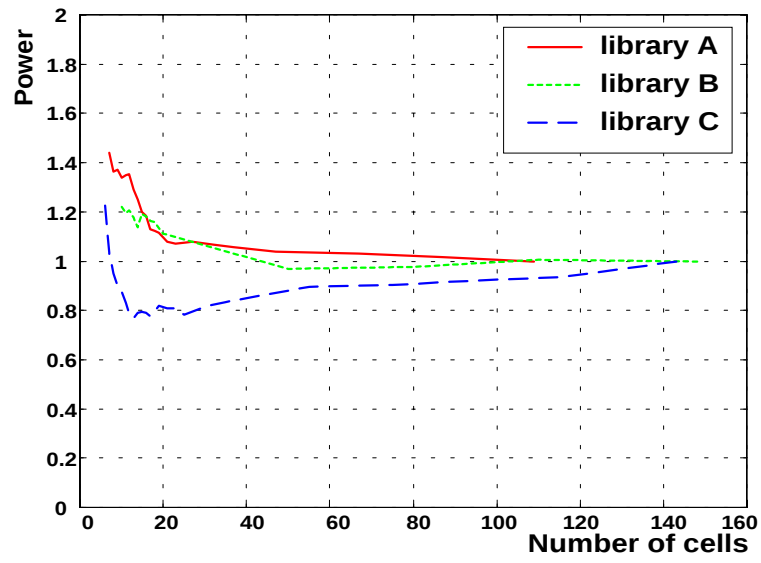


Fig. 11 Average relative power dissipation vs. number of cells

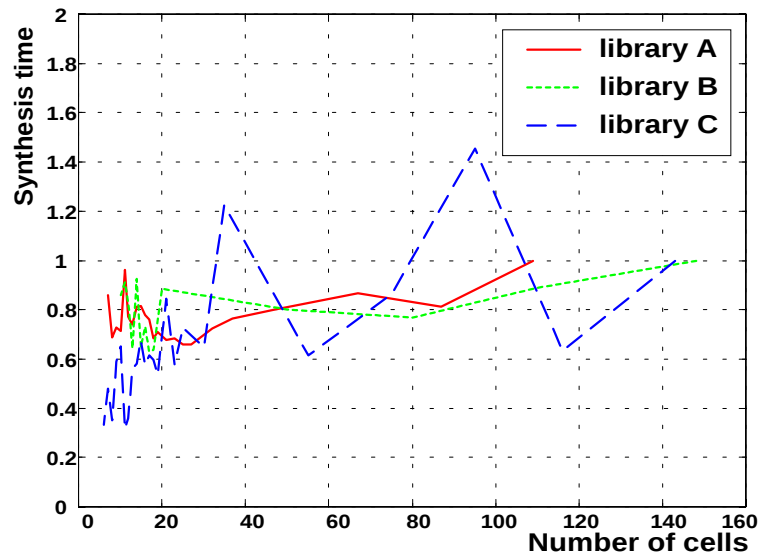


Fig. 12 Average relative synthesis time vs. number of cells

2.5. 最適なコンパクトライブラリー

前述したように遅延時間はセル数にほとんど依存せず、面積と消費電力はセル数が 20 以下になって始めて著しく増加するので、求める最適なコンパクトライブラリーのセル数は 20 以下であると考えられる。セル数が 20 のライブラリーは元の 400 セルのライブラリーと同等なパフォーマンス（遅延、面積および消費電力）を持つ。セル数が 10 以下になって遅延が初めて大きく増加するので、遅延だけが重要で、面積と消費電力はそれほど問題にならない、しかもライブラリーの生成時間を極力減らしたいような場合（新しいテクノロジーに素早く移行したい場合や、試作設計をする場合など）にはセル数が 10 のライブラリーが最も有用である。このような背景を考慮して、本研究ではセル数がそれぞれ 10 と 20 前後の 2 種類のライブラリーを提案する。

新しいライブラリーは、3 つの異なったライブラリー A, B, C を縮小化することによって得られるため、同じセル数でも中身が若干異なる。すなわち、セル数を n とすると、3 つのスターティングライブラリーに対して 3 つのコンパクトライブラリー $L_A(n)$, $L_B(n)$, $L_C(n)$ がある。 n が大きい（数百）時は、 $L_A(n)$, $L_B(n)$, $L_C(n)$ の共通な部分は少ない（極端に、 n が 400 の時は $L_A(n)$, $L_B(n)$, $L_C(n)$ はそれぞれ元のライブラリー A, B, C そのものであり、異なる部分が非常に大きい）。 n が小さくなると共通でない部分が徐々に少なくなる。 n が 20 前後の時は、 $L_A(n)$, $L_B(n)$, $L_C(n)$ 中のどれか 1 つのライブラリーの中であって、他のライブラリーにはないようなセルがごくわずかである。したがって、 $L_A(n)$, $L_B(n)$, $L_C(n)$ の共通な部分を求めるコンパクトライブラリーとすることができる。

提案の 2 つのコンパクトライブラリーはそれぞれ、上記の数値 n を 15 と 25 にした時のものである。この 2 つのライブラリーのはそれぞれ 11 と 20 セルを持っていることから、本研究では 11-CyHP と 20-CyHP (CyHP : Compact yet High Performance) ライブラリーと名づける。

$$11\text{-CyHP} \equiv L_A(15) \cap L_B(15) \cap L_C(15) \quad (6)$$

$$20\text{-CyHP} \equiv L_A(25) \cap L_B(25) \cap L_C(25) \quad (7)$$

11-CyHP と 20-CyHP を構成するセルを Table 4 と Table 5 に示す。見やすいようにセルをフリップフロップ、バッファ・インバーター類、基本ロジックゲート、複合ロジックゲートおよびマルチプレクサ - という 5 つのカテゴリーに分けている。表中の x_1 , x_2 などはセルの駆動力を表す。 x_1 がついているものは、最小ゲート幅を持ったトランジスタで構成される。 x_2 , x_4 , ... のついているセルは x_1 のついているセルの 2, 4, ... 倍の駆動力を持つ。セル名の前にある数値はそのセルの入力数を表す。例えば 3-AND-NOR は 3 入力セルであり、2-NAND は 2 入力のセルである。

11-CyHP は D フリップフロップと 2 入力 NAND, 2 入力 NOR, 2 入力 XNOR に加え、2-InvNAND と 2-InvNOR という 2 つの特種な複合ゲート、および出力が反転された 2 入力マルチプレクサ - 2-MUXInv を持っている。2-InvNAND (2-InvNOR) は片側の入力が反転された 2 入力 NAND (2 入力 NOR) である (Fig. 13)。

20-CyHP ライブラリーは 11-CyHP ライブラリーの全セルを持っているため、11-CyHP になセルのみが Table 5 に示される。20-CyHP は 11-CyHP に比べて、多くのフリップフロップとインバータを持っている。2 入力の NAND と NOR に加え、3 入力の NAND と NOR もある。複合ゲート 3-AND-NOR (3-OR-NAND) は 2 入力 AND (OR) ゲートの出力を 2 入力 NOR (NAND) の 1 つの入力に縦続接続したような 3 入力セルである (Fig. 13)。

両方のライブラリーにおいて、インバータは駆動力のバリエーションが最も大きく、使用頻度の大きいことを反映している。

Table 4 Contents of the 11-CyHP library

Category	Cell
Flip-flops	D-FF x1, D-FF x2
Inverters	INV x1, INV x2, INV x4
Primitive gates	2-NAND x2 2-NOR x2 2-XNOR x1
Compound gates	2-InvNAND x2 2-InvNOR x2
Multiplexers	2-MUXInv x1

Table 5 Contents of the 20-CyHP library

Category	Cell
Flip flops	D-FFN x1
Inverters	INV x8, INV x16
Primitive gates	2-NAND x1 2-NOR x1 3-NAND x1 3-NOR x1
Compound gates	3-AND-NOR x1 3-OR-NAND x1

(only cells that not in Table 4 are listed)

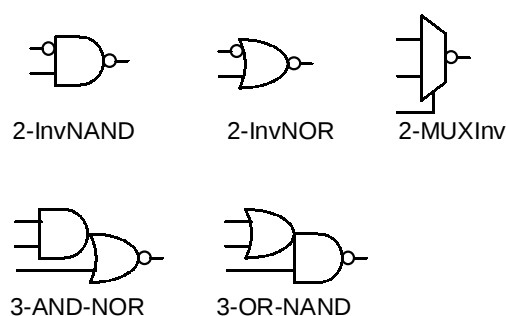


Fig. 13 Compound combinational cells in the proposed CyHP libraries

11-CyHP も 20-CyHP も トライステート・バッファ (high impedance 状態を有するバッファ) やリセット (セット) つきの フリップフロップ を持っていない。これは 使用した ISCAS89 ベンチマーク回路 がこのような機能を必要としなかったためである。このようなセルの使用は、RTL レベルでの記述 (または論理合成ツールの設定) によって避けることも可能であるが、設計スタイルや周辺回路とのインターフェイスなどによっては絶対的に必要になる場合がある。このような場合は、次のようにして CyHP ライブラリーを拡張できる。

- 各々のインバーターに対して、同じ駆動力を持ったトライステートバッファを追加する。
- 各々の D-FF に対して、同じ駆動力を持ったリセット (セット) つき D-FF を追加する。

必要に応じて a)と b)を独立に行うことができる。例えば、high impedance のバスインターフェイスを持った回路を設計する時は拡張 a)だけをすればよい。high impedance のデータバスとリセット入力を持った CPU コアを設計する時は同時に拡張 a)と b)をする。

11-CyHP と 20-CyHP を使った時の遅延時間、面積、消費電力および論理合成時間を Table 6 と Table 7 に示す。表中の数値は、31 個の ISCAS89 ベンチマーク回路の遅延、面積、消費電力および論理合成時間を、従来のライブラリー（約 400 セル）で論理合成した時に得られた値で割って規格化したものの平均値である（2.3 の（5）式）。

Table 6 と Table 7 は Fig. 9、Fig. 10、Fig. 11 および Fig. 12 のグラフと同様な結果を示しており、独立に縮小化される 3 つのライブラリーの共通な部分をコンパクトライブラリーと見なすことの妥当性を裏付ける。

11 セルからなる 11-CyHP を使う場合、遅延は平均 5 % 増える。面積と消費電力は平均してそれぞれ 3.5 %、5.8 % 増える。

20-CyHP を使うと、遅延の増加はほとんど無視できるほど小さくなり、わずか 2 % である。面積と消費電力の増加はそれぞれ 5 %、1.7 % に減り、11-CyHP に比べ大幅に改善される。

11-CyHP も 20-CyHP も平均論理合成時間は従来のライブラリーに比べ約 60 % 短縮される。

Table 6 Average relative delay, area, power and synthesis time of the 11-CyHP

Original library	Delay	Area	Power	Time
A	1.05	1.37	1.60	0.28
B	0.99	1.27	1.25	0.71
C	1.11	1.40	1.89	0.46
Average	1.05	1.35	1.58	0.48

(the numbers are calculated using equation (5) in section 2.3)

Table 7 Average relative delay, area, power and synthesis time of the 12-CyHP

Original library	Delay	Area	Power	Time
A	1.00	1.08	1.19	0.31
B	0.98	1.04	1.07	0.43
C	1.07	1.03	1.26	0.48
Average	1.02	1.05	1.17	0.41

(the numbers are calculated using equation (5) in section 2.3)

2.6. 実際の設計への応用

前節で得られた結果 (Table 6 と Table 7) はいずれも 31 個の ISCAS89 ベンチマーク回路 [10] の平均値である。ここでは、11-CyHP と 20-CyHP を実際の回路へ適用して、これらのコンパクトライブラリーの実用性を検証する。

検証に使われたサンプル回路は実際に製品化されたチップの一部で、主にランダムロジックから構成されている。この回路は RTL(Register Transfer Level)で記述されており、約 80K ゲート規模のものである。

サンプル回路は従来のライブラリー（400 セル）と本研究が提案しているコンパクトライブラリーの両方で論理合成された。得られた結果を Table 8 と Table 9 に示す。

Table 8 Relative performance of industry sample using 11-CyHP

Original library	Delay	Area	Power	Time
A	0.88	1.55	1.51	0.57
B	1.04	1.46	1.25	0.47
C	0.97	1.60	1.99	0.65
Average	0.96	1.54	1.58	0.56

Table 9 Relative performance of industry sample using 20-CyHP

Original library	Delay	Area	Power	Time
A	0.86	1.28	1.29	0.59
B	1.01	1.19	1.21	0.51
C	0.89	1.14	1.64	0.71
Average	0.92	1.20	1.38	0.60

11-CyHP は従来のライブラリーに比べ、遅延時間が逆に 4 % も小さくなる。面積と消費電力の増加は大きく、50 % 以上である。

20-CyHP を使う場合、遅延は更に小さくなり、従来のライブラリーに比べて平均 8 % よくなる。面積と消費電力の増加はそれぞれ 20 % と 38 % である。

また両方のコンパクトライブラリーは従来のライブラリーに比較して、論理合成時間が約 40 % 短縮されることも Table 8 と Table 9 から読み取れる。

2.7. 配線遅延による影響の検証

2.2 で述べたアルゴリズムは初期の論理合成段階で得られる情報を元にコンパクトライブラリーを生成するため、実際の配線による効果（遅延と面積の増加）は正しく反映されない。集積回路の微細化に伴い、配線による遅延はロジックゲートによる遅延に比べ相対的に増えている。最近のディープサブミクロンプロセスでは、特にファンアウトの大きい配線や、チップの隅から隅まで走る長いグローバル配線では、配線による遅延が支配的になっている。論理合成する時は、セルのがどの位置に配置され、セルを結ぶ配線がどのようなルートを通るかといった情報が全くないため、仮配線モデルで配線遅延や配線面積を見積もるしかない。

今回使用した論理合成ツール(Synopsys Design Compiler™)は、配線のファンアウト（同じ配線につながるピン数 - 1）と配線長との相関関係を利用した統計的なモデルを使用している[11]。このモデルは配線の抵抗と容量は配線長に比例しているとし、ファンアウトと平均配線長との関係を記述したテーブル、単位配線長当たりの抵抗および容量を内部に持っている。ファンアウトから配線長を見積もり、配線長から抵抗と容量を求めるのである。

以下に仮配線モデルの一例を示す。

```
wire_load("WL_EX"){
    resistance:    0.42          // wire resistance per unit length
    capacitance:   0.37          // wire capacitance per unit length
    area:          0.02          // wire area per unit length
    slope:         1.05          // used to estimate wire with large fanout
    fanout_length(1, 2.5)        // length of wire with fanout=1
    fanout_length(3, 4.5)        // length of wire with fanout=3
    fanout_length(4, 5.0)        // length of wire with fanout=5
```

}

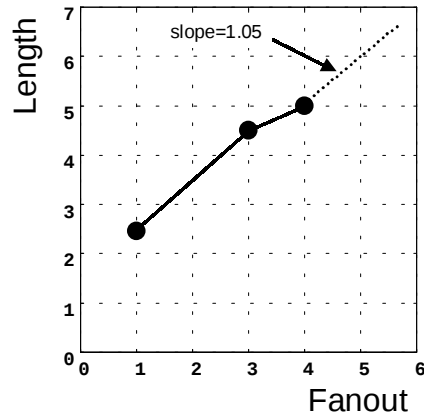


Fig. 14 Wire load model used by the logic synthesizer

WL_EX は配線モデル名を表している。WL_EX モデルではファンアウト 1,3 および 4 において配線長が定義されている。ファンアウト 2 における配線長はモデルの中に記述されていないので最も近いファンアウト (この場合は 1 と 3) から線形補完によって求められる。ファンアウト 5 以上のところでは slope ステートメントで定義されている傾き (この場合は 1.05) およびファンアウト 4 における配線長から線形外挿(extrapolation)によって計算される (Fig. 14 の点線部分)。

resistance、capacitance および area の後に記述されている数値はそれぞれ単位配線長当たりの寄生抵抗、寄生容量および面積である。ファンアウトから計算された配線長にこれらの比例定数を掛けることによって配線抵抗、容量および面積を求める。

実際、同じファンアウトの配線は必ずしも同じ長さを持っていない。また配線が複数のコンタクトを通り、複数のメタル層にまたがるため、容量と抵抗 (特に抵抗) は必ずしも配線長と比例関係にはないことから、このような仮配線モデルは正確に配線の抵抗や容量を表せない。更に、ファンアウトが 2 以上の場合は配線がツリー構造の形をするが、枝分かれの位置情報がまったくないため、配線全体の抵抗と容量を正確に見積もれたとしても個々の配線セグメントの抵抗と容量を計算できないので、個々の終点ノードまでの遅延を正確に求められない。例えば、Fig. 15 のようなツリー配線ではノード 1 がノード 2 に比べて始点ノード (駆動ノード) に近いので、ノード 1 までの遅延 D_1 はノード 2 までの遅延 D_2 より小さいが、仮配線モデルでは $D_1=D_2$ と見積もられる。

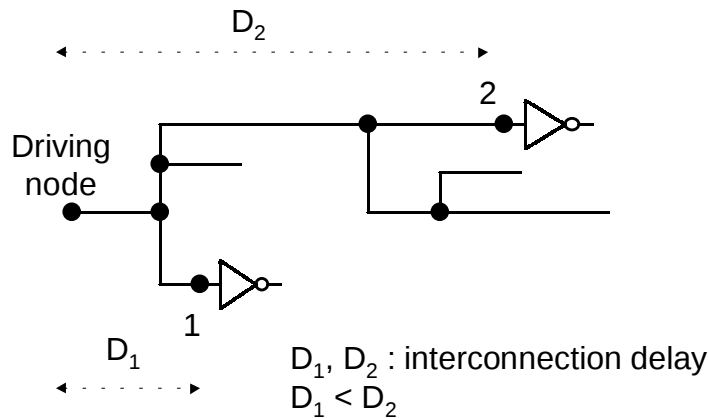


Fig. 15 Interconnection delay in case of tree wire

真の配線抵抗と配線容量は配置配線が完了するまで未知数であるため、このような仮配線モデルで見積もらなければならない。前述した理由により、論理合成を行う時に見積もられる配線情報と配置配線フェーズが完了した後に得られる実際の配線情報は一致しないことが多い。通常は配置配線後に得られる情報を論理合成ツールに戻して再最適化を行わなければならない (back-annotation)。場合によっては論理合成と配置配線 (または部分的な配置配線) を数回繰り返す必要がある。

前節で得られた結果はいずれも仮配線モデルで計算されたもので、実際のレイアウト後には変わる可能性がある。ここでは、2つのサンプルに対して、詳細な配置配線を行うことによってこれらの結果を検証する。

サンプル回路の1つは2.6にも出てきた80Kゲートの回路で、もう1つは組込み用の32ビットRISC CPUコアである。レイアウトは0.25[μm]、4メタル層のプロセスで行われた。20-CyHPライブラリーを使った時のレイアウト図をFig. 17とFig. 18示す。

レイアウト作業は完全に自動化できるわけではなく、人手による細かい設定が必要になるが、ここではできる限り実際の設計現場に近い条件で行った。Fig. 16にレイアウトフローが示されている。まず、RTLレベルのサンプル回路を論理合成してゲートレベルネットリストを生成する。同時に各パスのタイミング制約 (設定したクロックに対してどのぐらいの余裕があるのかといったような情報) も得られる。この制約情報をネットリストと一緒に配置配線ツールに与えて一回目のレイアウトを行う。配置配線ツールは渡された制約が満たされるようにレイアウトを生成する。配置配線後に得られる詳細な配線情報を論理合成ツールに戻してもう一回最適化を実行する。思いのほか配線遅延が大きくなったパスにバッファを挿入したり、セルをより駆動力の大きい (場合によっては小さい) ものに置換えたりすることによって部分的な最適化が行われる。再最適化された新しいネットリストを配置配線ツールに渡して2回目の配置配線を行い、最終的なレイアウトを生成する。2回目の配置配線では変更のあった部分について配置配線が行われ、全体的なレイアウトには大きな変更が加えられない。

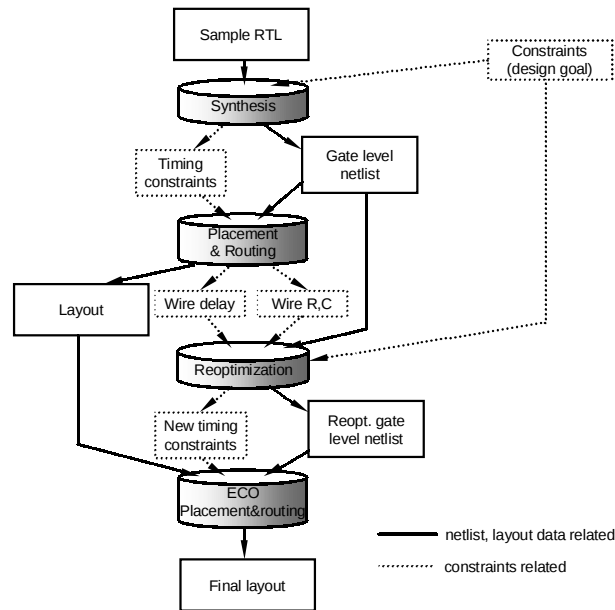


Fig. 16 Layout flow used to verify the proposed CyHP libraries

Table 10 Increases of delay, area and power of the 80K gate sample when using 11-CyHP

Performance	11-CyHP	20-CyHP
Delay	+4.3%	+0.8%
Area	+69.9%	+23.8%
Power	+77.2%	-0.7%

Table 11 Increases of delay, area and power of the 32 bit CPU core when using 11-CyHP

Performance	11-CyHP	20-CyHP
Delay	- 5.1%	- 2.6%
Area	+ 41.63%	+17.3%
Power	+139.6%	- 2.6%

従来のライブラリーとコンパクトライブラリーの両方を用いて、以上の手順でサンプル回路の論理合成と配置配線を行った。コンパクトライブラリーを用いた場合の、各々のサンプルの遅延、面積および消費電力の増加率をTable 10とTable 11に示す。

仮配線モデルで配線の影響を見積もる場合（Table 8とTable 9）に比べ、11-CyHPでは大きな面積と消費電力の増加が見られ、80K サンプルの場合は70%前後に達し、32ビットCPUコアの場合は100%を超えている。面積と消費電力がまったく問題にならないような場合に限って、11-CyHPが有用であると言える。

一方、20-CyHPの場合では遅延と消費電力の増加はほとんど無視でき、面積の増加は約20%である。実際のシステムLSIは、セルベースで設計されるランダムロジック以外にデータパスやメモリーなど大きなブロックがあり、ランダムロジック部は全体の20～30%しか占めないため、その面積が20%増えても総面積は数%しか増えない。したがって、20-CyHPを使った時の20%の面積増加は実用上ほとんど問題にならない。面積と違って、遅延がチップ全体の動

作周波数に直接影響を与えるので、コンパクトライブラリーを使用しても遅延が増えないことは非常に重要である。

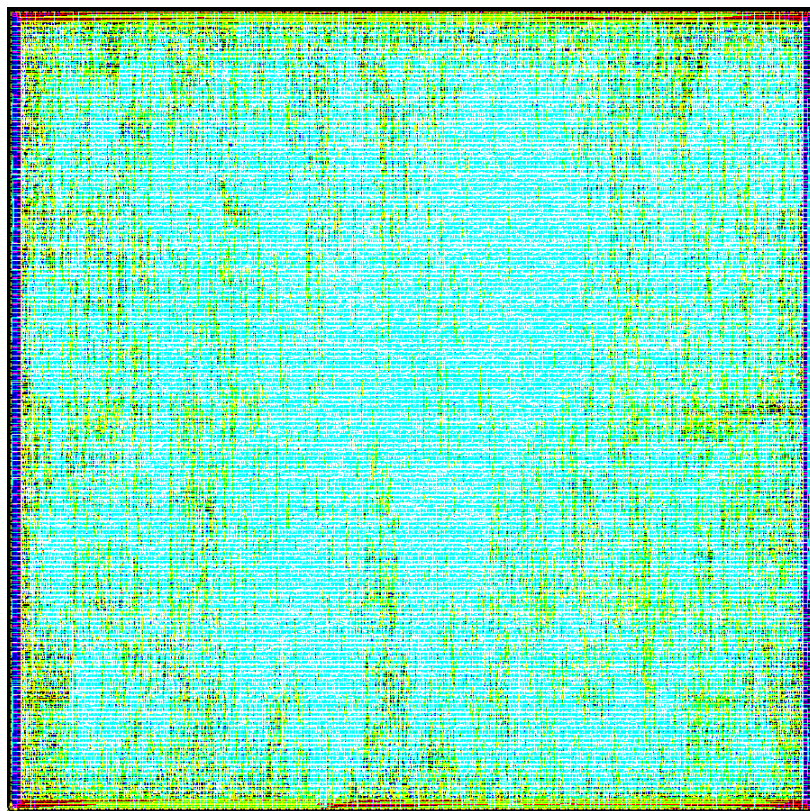


Fig. 17 Layout of the 80K gate sample (0.25[μ m], 4 metal layers, die size = 1.60[mm])

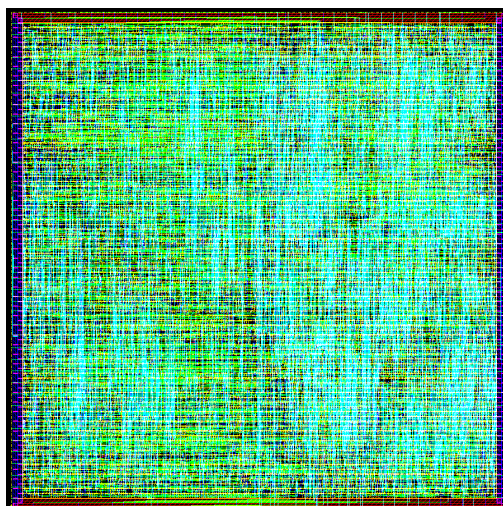


Fig. 18 Layout of the 32-bit RISC CPU core (0.25[μ m], 4 metal layers, die size = 0.91[mm])

第3章 低電源電圧 depletion 型 CMOS の最低動作閾値電圧

3.1. Depletion 型 CMOS 混在のライブラリー

エラー! 参照元が見つかりません。で述べたように、高速かつ低消費電力 LSI を実現するために電源電圧と閾値電圧の両方を同時に下げなければならない。閾値電圧を下げると消費電力が大きく増加するが、消費電力よりも遅延時間の方が重要なクリティカルパスなどで、動作速度を最大限に引き出すために従来の enhancement 型 CMOS の代わりに depletion 型 CMOS を使うことが考えられる。クリティカルパス以外の部分は消費電力の方が重要であるので、enhancement 型 CMOS でインプリメントする必要がある。したがって、セルライブラリーは、従来の enhancement 型 CMOS に加え、depletion 型 CMOS で構成されるセルも含んでいなければならない。Depletion 型 CMOS の場合、閾値電圧に下限が存在し、下限値よりも閾値電圧を低くすると回路が動作しなくなる。本節は、典型的な論理回路であるインバーターを例に取り、この閾値電圧の下限を求め、enhancement 型 CMOS と depletion 型 CMOS の両方が混在したライブラリーの設計ガイドラインを定める。

3.2. 最低動作閾値電圧の定義

Fig. 19 に CMOS インバータの伝達曲線が示されている。pMOS と nMOS が完全に対称であるとすれば、インバータが $V_{dd}/2$ のところで反転する (V_{dd} は電源電圧)。 $V_{dd}/2$ における伝達曲線の傾き A はインバータの小信号に対する増幅率を表し、インバータの動作を決定する重要なパラメータである。

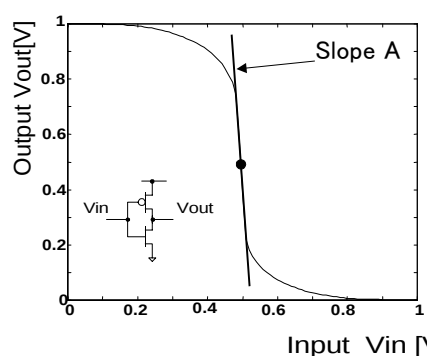


Fig. 19 Transfer curve of a symmetrical CMOS inverter

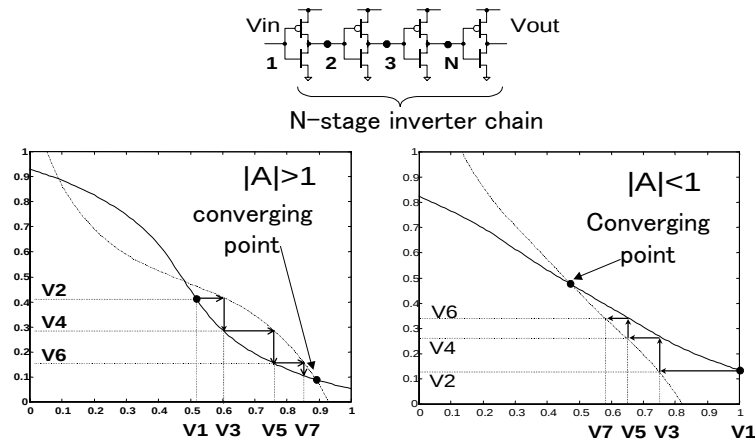


Fig. 20 Relationship between $|A|$ and the operation of an inverter.

$|A| > 1$ ¹の時は、インバータが増幅作用を持ち、出力信号の振幅が入力信号よりも大きくなる。Fig. 20 にあるような数段から構成されるインバータチェーンに方形波を入力として与えた場合、後段にいけばいくほど信号の振幅が大きくなっていく。すなわち、第 n 段に入る信号の振幅を V_n とすれば、

$$V_1 < V_2 < V_3 < \dots$$

が成り立つ。入力電圧が少々劣化しても数段のゲートを経た後に、振幅が V_{dd} に近い値まで修復される (Fig. 20の左側)。逆に、 $|A| < 1$ の時はインバータが信号を減衰させるため、

$$V_1 > V_2 > V_3 > \dots$$

となり、フル振幅を持った信号が入力されても最終的に出てくる信号はほとんど $V_{dd}/2$ に収束してしまう (Fig. 20の右側)。この場合はインバータがその論理機能である反転作用を完全に失っていると言える。

閾値電圧が正 (enhancement 型) の場合は CMOS 回路がトランジスタの飽和領域でスイッチングする (Fig. 21の左)。 $V_{dd}/2$ の近傍では入力 (nMOS のゲート) 電圧の微小な変化でも出力電圧 (nMOS のドレイン電圧) の大きな変化を引き起こすため、傾き $|A|$ は常に 1 以上であり、回路はどのように設計しても必ず動作する。これに対して、閾値電圧が負 (depletion 型) の場合はスイッチングがトランジスタの線形領域で行われるため、 $|A|$ は enhancement 型の場合に比べてはるかに小さくなる。更に閾値電圧を低くし、ある一定値以下にすると、やがて $|A| < 1$ になり、インバータが動作しなくなる。したがって、 $|A|$ がちょうど 1 となるような閾値電圧を depletion 型 CMOS が動作するための最低閾値電圧として定義することができる。

¹ インバータの場合は、出力電圧が入力電圧と逆の方向に変化するため、増幅率 A は常に負数である。ここで A の大きさを議論するので、絶対記号を使っている。

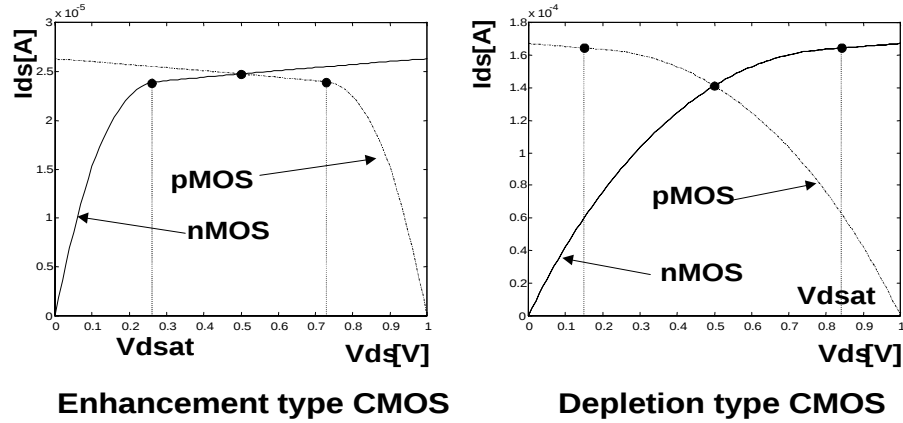


Fig. 21 Switching region(shaded area) of CMOS circuits in I_{ds} - V_{ds} plane.

3.3. 最低動作閾値電圧の理論計算

与えられた入力電圧 V_{IN} に対する出力電圧 V_{OUT} を求め、 V_{OUT} を V_{IN} で微分し、前節の定義 $|dV_{OUT}/dV_{IN}| = 1$ から最低動作閾値電圧 V_{THMIN} を求めることができる。

MOS トランジスタのドレイン電流 I_{DS} は α 乗則[9]を用いて次のように表せる。

$$V_{DSAT} = K_V (V_{GS} - V_T)^n \quad (8)$$

$$I_{DSAT} = K_I (V_{GS} - V_T)^\alpha \quad (V_{DS} \geq V_{DSAT}) \quad (9)$$

$$I_{DS} = I_{DSAT} (1 + \lambda V_{DS}) (2 - \frac{V_{DS}}{V_{DSAT}}) \frac{V_{DS}}{V_{DSAT}} \quad (V_{DS} < V_{DSAT}) \quad (10)$$

ここで、 V_{GS} 、 V_T 、 V_{DS} はそれぞれゲート・ソース間電圧、閾値電圧、ドレイン・ソース間電圧であり、 I_{DS} は線形領域におけるドレイン電流、 I_{DSAT} 、 V_{DSAT} は飽和ドレイン電流、飽和ドレイン電圧である。 λ はチャンネル長変調係数、 K_V 、 K_I および α 、 n はフィッティングパラメータである。Shockley の古典モデルが成り立つような長チャネルトランジスタでは、 $\alpha=2$ 、 $n=1$ 、 $K_V=1$ となる。ホットキャリアー効果の大きい短チャネルトランジスタでは、 α は 1 に近い (1 より大きい) 値を、 K_V は 0.4 ~ 0.7 前後の値を取る。に SPICE で計算した静特性を上記の α 乗則モデルにフィッティングした時に得られる α 、 n および K_V の値を示す。

Table 12 Actual values of α power law parameters

Parameter	Shockley model	Process A (1.5 [μ m])		Process B (0.6 [μ m])		Process C (0.25 [μ m])	
		nMOS	pMOS	nMOS	pMOS	nMOS	pMOS
α	2	1.72	1.98	1.57	2.00	1.54	1.62
n	1	1.29	0.44	0.71	1.00	0.36	0.78
K_V	1	0.43	0.43	0.64	0.66	0.36	0.71

前述したように、depletion 型 CMOS は線形領域で動作するためドレイン電流は式 (10) で計算しなければならない。計算を容易にするために次のような近似をする。 λ は飽和領域におけるドレイン電流の変化の度合いを示す係数であるが、線形領域では λ による影響は無視できると考えられるので、0 とする。飽和電圧の式に現れるパラメータ n は $\alpha/2$ とする。nMOS と pMOS が同じ全く対称的であり、

$$V_{TN} = -V_{TP} = V_T (< 0); \alpha_N = \alpha_P \quad (11)$$

$$K_{VN} = K_{VP} = K_V; K_{IN} = K_{IP} = K_I \quad (12)$$

が成り立っていると仮定するれば、インバータの出力電圧 V_{OUT} を入力電圧 V_{IN} 、電源電圧 V_{DD} およびパラメータ K_V 、 α の関数として表せる。方程式 $|dV_{OUT}/dV_{IN}| = 1$ を解けば、最低動作閾値電圧 V_{THMIN} を求めることができる。一般的に V_{THMIN} は V_{DD} 、 K_V と α の 3 変数関数として表せる。

$$V_{THMIN} = f(V_{DD}, K_V, \alpha) \quad (13)$$

任意の V_{DD} 、 K_V および α に対して、式 (12) の関数 f を解析的に求めることはできないが、 α が 2 の時、(12) は、

$$V_{THMIN} = -V_{DD} / 2 \quad (14)$$

となる。 $\alpha = 2$ 以外の場合については f を数値計算によって求めなければならない。

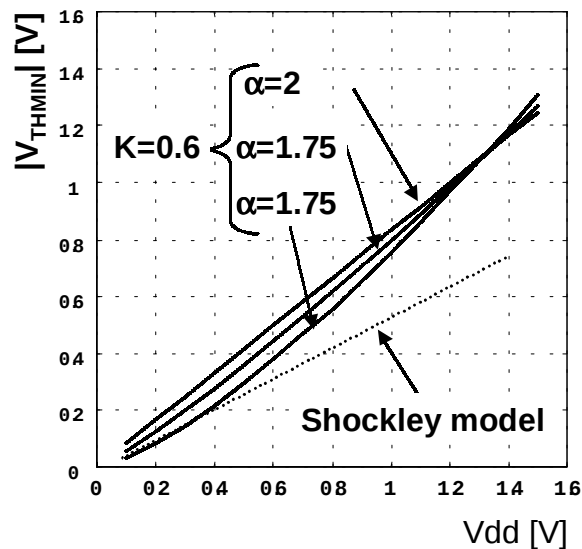


Fig. 22 $|V_{THMIN}|$ versus V_{DD} curve plotted with various values of α

Fig. 22 に電源電圧 V_{DD} を変えた時の $|V_{THMIN}|$ の変化を示す。ここで K_V は Table 12 中の平均値 0.6 に設定されている。比較しやすいように Shockley の古典モデルを使った場合の $|V_{THMIN}|$ も同じ図に示す (点線)。3 本の実線はそれぞれ $\alpha=2, 1.75, 1.5$ の場合に対応している。

Fig. 22 から $|V_{THMIN}|$ がほぼ電源電圧 V_{DD} に比例していることが分かる。計算した 3 つの α の値については比例定数は約 0.8 である。 V_{DD} が約 1.3[V] 以下の領域では、 α が小さくなると $|V_{THMIN}|$ も小さくなる。短チャネル効果が大きければ大きいほど、閾値電圧のとれる範囲が狭くなることを意味する。また、 α が 2 から 1.5 に変わった時、 $|V_{THMIN}|$ は電源電圧に関係なくほぼ 0.2[V] 小さくなる。電源電圧が比較的大きい (1[V] 以上) 時は、 $|V_{THMIN}|$ も大きく、 α の変化による影響はそれほど問題にはならない。しかし、極低電源電圧 (0.5[V] 以下) の領域では、短チ

ヤネル効果によって α が1に近い値を取るようにになると、 $|V_{THMIN}|$ の相対的な減少が大きく、閾値電圧のマージンがより厳しくなる。

α を変えた時の最低動作閾値電圧 $|V_{THMIN}|$ の変化をFig. 23 に示す。Fig. 22と同様 K_V は0.6 に設定されている。 V_{DD} が比較的大きい時 ($V_{DD}=1.25[V]$ の曲線) は、 $|V_{THMIN}|$ は α に関係なくほとんど一定であることが分かる。電源電圧 V_{DD} が小さくなるにつれて、 $|V_{THMIN}|$ の変化を表した曲線の傾きが次第に大きくなる。低電源電圧領域では、 $|V_{THMIN}|$ が α に強く依存することが分かる。

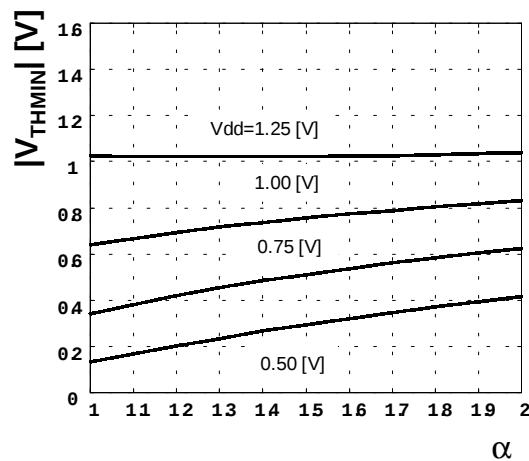


Fig. 23 $|V_{THMIN}|$ versus α plotted with various values of V_{DD}

以上の結果から2つの重要な結論を得る。

- 最低動作閾値電圧(の絶対値) $|V_{THMIN}|$ はほぼ電源電圧 V_{DD} に比例している。 $|V_{THMIN}|$ は α にも依存するが、一般的に $|V_{THMIN}| = 0.5 \sim 0.8 V_{DD}$ であり、電源電圧を下げると閾値電圧の取れる範囲がそれに比例して狭くなる。
- α が小さくなると $|V_{THMIN}|$ も小さくなる。微細化がますます進む今後、閾値電圧のマージンが更に厳しくなることを意味する。

3.4. SPICE シミュレーションによる検証

α 乗則を用いた前節の計算過程において、 n や K_V など一部のパラメータは典型的な値に設定された。また、 α 乗則自体にもある程度の誤差があるため、得られた結果をシミュレーションなどで検証する必要がある。もっとも望ましいのは、実際の測定による検証であるが、実際のデバイスで閾値電圧を単独で広い範囲に渡って変化させるのは極めて難しいので、ここでは SPICE シミュレーションによって検証を行った。

特定の閾値電圧において回路が動作するかどうか判定するために、20 段のインバータチェーンにフル振幅(電源電圧 V_{DD} と同じ振幅)の方形波を入力として与え、最終段の出力を観測する。方形波が出力されていれば回路が動作するとし、そうでなければ動作しないと判定する。

閾値電圧は SPICE のモデルファイル中の V_{FB} (モデルによっては V_{TH0})というパラメータを書き換えることによって擬似的に変化させる。実際、閾値電圧を変えるとトランジスタの他の特性も変わり、モデルの他のパラメータと独立に閾値電圧だけを変化させることはできないので、実際の低閾値電圧 MOS トランジスタを完全にモデリングすることができないが、結果には大きなずれが生じないと思われる。 $V_{gs}=V_{ds}=V_{DD}$ の時に pMOS トランジスタと nMOS トランジスタが同じ駆動力(同じ I_{ds})を持つようにゲート幅を合わせる。更に両トランジスタの閾値電

圧の絶対値も等しくなるように、モデルファイルのパラメータ (V_{FB} , V_{TH0}) を変更する。なお、 α 、 n や K_V に関しては SPICE モデルファイルの変更によって合わせられないため、pMOS と nMOS は完全に対称的ではない。

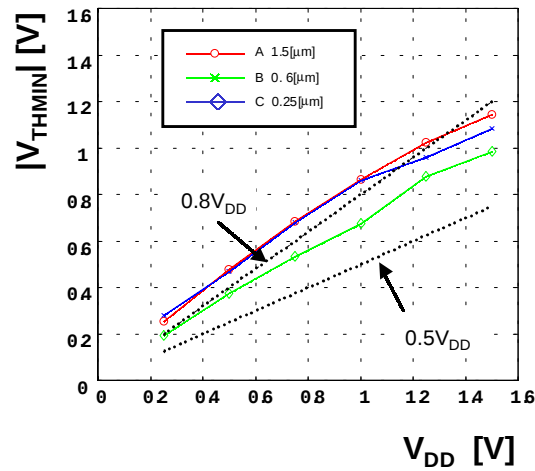


Fig. 24 Simulation results of $|V_{THMIN}|$

Fig. 24 に3種類のプロセスに対するシミュレーション結果を示す。これらのプロセス (A,B および B) の α 乗則のパラメータは前節のTable 12にまとめられている。比較しやすいように理論計算で予測されていた最低動作閾値電圧 $|V_{THMIN}| = 0.5 \sim 0.8V_{DD}$ を傾き 0.5 および 0.8 の2本の点線で同図に示した。

Fig. 24 から最低閾値電 $|V_{THMIN}|$ は約 $0.8V_{DD}$ であることが分かる。プロセス C (0.25[μm]) の $|V_{THMIN}|$ はプロセス A(1.5[μm]) および B(1.5[μm]) より小さいが、これは微細化に伴って α が小さくなると閾値電圧のとれる範囲が狭くなるという理論計算の結果 (Fig. 23) と一致している。

第4章 結論

本研究で得られた結論を以下にまとめる。

1) コンパクトなスタンダードセルライブラリー

プロセス技術の目覚ましい発展に伴って、スタンダードセルライブラリーの作成が頻繁に行われるようになってきた。従来のライブラリーは非常に多くのセルから構成されているため、新たに作成するのは大変時間とコストがかかる。新しいテクノロジーへ移行する際に、ライブラリーを作り直す作業が大きなボトルネックになっている。この問題の解決策として、セル数が少なく、しかも従来と同等なパフォーマンスを持ったコンパクトなライブラリーを提案した。

セルが占める平均面積の割合をそのセルの重要度と定義し、重要度の小さいセルを順番に取り除くことによってライブラリーを縮小化するアルゴリズムを提案した。セル数がそれぞれ 11 と 20 の 2 つのコンパクトなライブラリー、11-CyHP と 20-CyHP (CyHP: Compact yet High Performance) を導出した。11 セルのライブラリー 11-CyHP は、従来のライブラリーに比べて遅延、面積および消費電力がそれぞれ平均 5 %、35 %、58 % 増加する。20 セルのライブラリー 20-CyHP を用いれば、遅延の増加はわずか 2 % であり、面積と消費電力の増加はそれぞれ 5 % と 17 % に押さえられる。

実際に製造された LSI のランダムロジック部を提案のライブラリーで論理合成し従来のライブラリーと比較することによって提案のライブラリーの実用性を示した。

提案のライブラリーを生成する時、配線の遅延などは論理合成ツールの仮配線モデルによって見積もられたが、2 つの大きなサンプル回路の詳細な配置配線を行い、実際の配線遅延を考慮しても同様なパフォーマンスが得られることを示した。

2) 低電源電圧 depletion 型 CMOS の最低動作閾値電圧

低消費電力かつ高速な LSI を実現するために閾値電圧を電源電圧と同時に下げなければならぬ。将来的には、消費電力よりも遅延時間の方が重要なクリティカルパスにおいて従来の enhancement 型 CMOS に代わって depletion 型 CMOS が使われるようになる可能性がある。Depletion 型 CMOS の動作閾値電圧には下限が存在し、この下限よりも閾値電圧を低くすると回路が動作しなくなってしまう。Depletion 型 CMOS のセルが混在したスタンダードセルライブラリーを設計する上で、重要なガイドラインの 1 つとなるこの下限値 (最低動作閾値電圧) を理論計算で求め、SPICE シミュレーションで検証を行った。最低動作閾値電圧 (の絶対値) はほぼ $0.5 \sim 0.8V_{DD}$ (V_{DD} は電源電圧) であることを示した。微細化に伴って、ホットキャリア効果が顕著になると閾値電圧の取れる範囲が更に狭くなることも示した。

付録 A ベンチマーク回路のフォーマット変換

1. ベンチマーク回路のフォーマット

ISCAS89 ベンチマーク回路は独自のフォーマットで書かれているため論理合成ツールが理解できる VHDL または Verilog フォーマットに変換する必要がある。本研究はベンチマークデータと一緒に公開されている変換プログラムを修正して、VHDL フォーマットのネットリストを出力できるようにした。

ISCAS89 ベンチマークのフォーマットは一種のネットリストフォーマットで、入出力端子、ゲートおよびそれらを結ぶネットの接続情報を ASCII 形式で記述する。このフォーマットでは以下の予約語が使われる。

予約語	意味
INPUT	入力端子
OUTPUT	出力端子
AND	多（2 つ以上）入力 AND
NAND	多入力 NAND
OR	多入力 OR
NOR	多入力 NOR
XOR	多入力 XOR
XNOR	多入力 XNOR
BUFF	バッファ
NOT	インバーター
INV	インバーター
DFF	D フリップフロップ

接続情報は以下のいずれかのステートメントで記述される。

INPUT(input_pin_name)

input_pin_name が一つの入力端子であることを示す。

OUTPUT(output_pin_name)

output_pin_name が一つの出力端子であることを示す。

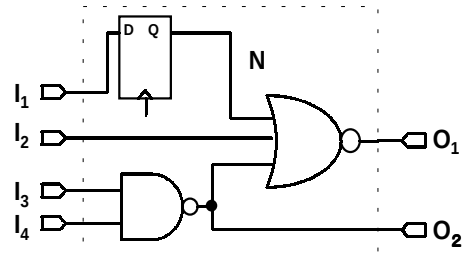
out_net = gate_name(in_net1, in_net2, ..., in_netn)

ゲート gate_name の入力ピンはそれぞれネット in_net1, in_net2, ..., in_netn に、出力ピンはネット out_net に接続されるのを示す。

#から改行文字までの部分はコメントとして扱われる。フリップフロップは D-FF に限定される。クロック端子は省略されるため VHDL フォーマットに変換する時に追加しなければならない。

例えば、右図の回路は次のように ISCAS89 フォーマットで記述できる。

```
# a sample netlist described in ISCAS89 format
INPUT(I1)
INPUT(I2)
INPUT(I3)
INPUT(I4)
OUTPUT(O1)
OUTPUT(O2)
```



```

OUTPUT(O2)
N = DFF(I1)
O1 = NOR(N, I2, O2)
O2 = NAND(I3, I4)

```

2. 変換プログラム

変換プログラムは以下のファイルから構成されている

Table 13 変換プログラムのソースファイル

Makefile	メークファイル。これを実行すると trans という変換プログラムが生成される。
trans.l	ISCAS89 ネットリストの構文を解析するモジュール
trans.y	ISCAS89 ネットリストをトークンに分割するモジュール
vhdl.c	trans.l の出力結果をネットリストを表す配列に変換したり、VHDL 形式でネットリストを書き出したりする関数群
vhdl.h	vhdl.c の関数や定数などを宣言するヘッダファイル
main.c	上記のモジュールを呼び出すメインプログラム。

なお、プログラムソースは紙面上の都合掲載しないが、ウェブサイト

<http://lowpower.iis.u-tokyo.ac.jp/opensource/iscas89translators>

からダウンロードできる。

付録 B ライブラリー生成プログラム

1. 環境設定

Synopsys Design Compiler™ が論理合成ツールとして使われるのでまず Design Compiler™ が動作する環境が必要である。ライブラリー生成プログラムは Perl で実装されているため、Perl[12]のインタープリターが正しくインストールされていることも必要である。

評価用のベンチマーク回路やスターティングライブラリーは予め Synopsys の*.db フォーマットに変換し、特定のディレクトリーに置かなければならない。以下、/usr/data を*.db ファイルを置くディレクトリーと仮定して説明する。このディレクトリーの下に lib と db という2つのサブディレクトリーを作成する。db ファイルのディレクトリーは任意に選べるが、その下にあるサブディレクトリーの名前(db と lib)は変更できない。/usr/data/lib には全スターティングライブラリーの*.db ファイルを、/usr/data/db には全ベンチマークの*.db ファイルをコピーする。

最後にライブラリー生成プログラムの \$stopdir というグローバル変数の初期値を上記の /usr/data を書き換え、プログラムにデータの置かれている場所を教える。

なお、ライブラリーのパスなどは自動的に設定されるので .synopsys_setup.dc ファイルを作る必要はない。

2. プログラムの構成

ライブラリー生成プログラムは以下のTable 14に示されているように、2つのファイルからなっている。

Table 14 ライブラリー生成プログラムのソースファイル

ファイル名	機能
reduce_cell.pl	重要度の小さいセルを順番に取り除き、ライブラリーを縮小化する。
get_area_timing_power.pl	縮小化されたライブラリーのパフォーマンスを評価する。

紙面上の都合これらのファイルは掲載しないが、以下のウェブサイトからダウンロードできる。

<http://lowpower.iis.u-tokyo.ac.jp/opensource/cyhp>

プログラムは次のようにして実行する。

```
%reduce_cell.pl tmp_dir
```

でライブラリーを縮小化する。tmp_dir は任意のディレクトリーで、中間結果がこの下に作成される。reduce_cell.pl が終了した後に、

```
%reduc_cell.pl tmp_dir
```

で tmp_dir に保存された中間計算結果を解析し、コンパクトライブラリーのセル名やパフォーマンスなどを計算する。

2.2 で説明したように、スターティングライブラリーから重要度の小さいセルを順番に取り除くことによってコンパクトライブラリーを生成する。スターティングライブラリー名、ベンチ

マークファイル名などは対応するグローバル変数`$library_name`、`@design_list`の初期値を書き換えることによって指定する。

2.2で説明した `synthesize – analyze – remove` ループの `n` 回目の実行が開始される前に、`tmp_dir/`の下に `n` というサブディレクトリが作られ、`tmp_dir/n` の下に `compile.scr` というファイルが作成される。`compile.scr` は全ベンチマーク回路を指定されたライブラリーで論理合成するためのスクリプトファイルである。また、ループの `n-1` 回目の実行後に、全セルの重要度が計算され、取り除かれるセルは `compile.scr` の中で `set_dont_use` という Synopsys のコマンドで使われないように設定される。`reduce_cell.pl` は内部で

```
dc_shell -f tmp_dir/n/compile.scr > tmp_dir/n/compile.log
```

のような形で、論理合成ツールを呼び出す。`compile.log` ファイルは論理合成が成功したかどうかを判定するために使われる。また、`tmp_dir/n/`の下に `xxx_timing_rpt.txt`, `xxx_area_rpt.txt`, `xxx_power_rpt.txt` という3種類のファイルが作られる。これらのファイルは `xxx` というベンチマーク回路の遅延、面積および消費電力に関する情報を持ち、ライブラリーのパフォーマンスを評価するために `get_area_timing_power.pl` によって使われる。全ベンチマーク回路が `compile.scr` によって論理合成された後に、すべてのセルの重要度が `xxx_area_rpt.txt` ファイルを解析することによって求められる。`tmp_dir/n/all_cell_perc.txt` というファイルに、重要度の小さい順に並べられたセルのリストが出力される。このリストは次の `synthesize – analyze – remove` ループの実行に使われる。

参考文献

- [1] Neil H.E. Weste and Kamran Eshraghian, *Principle of CMOS VLSI Design A Systems Perspective*, Second Edition, Addison-Wesley Publishing Company, 1993.
- [2] Antonio M. Martinez, Suresh Dholalia and Steve Bush, "Compilation of Standard-Cell Library", *IEEE Journal of Solid State Circuits*, Vol. 22, No. 2, April 1987.
- [3] Semiconductor Industry Association, *The National Technology Roadmap for Semiconductors*, 1998.
- [4] Dennis V. Heinbruch, *CMOS3 Cell Library*, Addison Wesley, 1988.
- [5] Baltus Donald G., Varga Thomas, Armstrong Robert C., Duh John and Matheson T.G., "Developing a Concurrent Methodology for Standard Cell library generation", *Proceedings of DAC*, 1997.
- [6] William Kao, Nahid Ansari, Kang Chan, "Automatic Characterization System for Standard Cells", *Proceedings of CICC*, 1987.
- [7] A. Mehmet Cirit, "Characterizing a VLSI Standard Cell Library", *Proceedings of CICC*, 1991.
- [8] Takayasu Sakurai (Ed.), *Low Power High Speed LSI Circuits & Technology*, Realize Inc., 1998.
- [9] Takayasu Sakurai, R. Richard Newton, "A simple MOSFET for Circuit Analysis", *IEEE Transactions on Electron Devices*, Vol. 38, No. 4, April 1991.
- [10] Franc Brglez, David Bryan, Krzysztof Kozminski, "Combinational profiles of sequential benchmark circuits", *Proceedings of ISCAS*, pp. 1929-1934, 1989.
- [11] Design Compiler Reference Manual, Synopsys, 1998.
- [12] Larry Wall and Randal L. Schwartz, *Programming Perl*, Reading, O'Reilly & Associates Inc., 1991.

本研究に関する発表

- [1] ゲン・ドゥック・ミン、野瀬浩一、桜井貴康、“低電源電圧 depletion 型 CMOS の最低動作閾値電圧”、1999 年電子情報通信学会春季総合大会、A-3-12 .
- [2] Nguyen Minh Duc and Takayasu Sakurai, “Compact yet High Performance(CyHP) Library for Short Time-to-Market with New Technologies”, *Proceedings of ASP-DAC 2000*, pp. 475-480.

謝辞

本研究および本論文の作成に当たり、終始ご懇切なるご指導を賜りました指導教官である桜井貴康教授に衷心より御礼を申し上げます。

また本研究を進めるに当たり、数々のご援助、ご助言を頂きました川口博氏、野瀬浩一氏、神田浩一氏をはじめ、桜井研究室の職員と大学院生の方々に深謝いたします。

そして、熱心なご指導とともに、研究環境の整備に当たり、ご尽力を頂きました濱田基嗣氏をはじめ、東芝株式会社の方々に感謝の意を表します。