The Institute of Electronics, Information and Communication Engineers

——サーベイ論文——

プロセッサ設計手法の現状と今後——高性能化を実現する設計フローとCADシステム——

伊藤　則之†　　安永　守利†

Present and Future of Processor Design Methodology ——Design Flow and CAD System to Achieve Higher Performance——

Noriyuki ITO† and Moritoshi YASUNAGA†

あらまし　プロセッサの設計では，高い動作周波数を実現するために様々な設計手法が研究され，そして適用されている．このような高性能実現のためには，アーキテクチャ設計や論理設計だけでなく，半導体上で実現する回路設計や物理設計も重要となる．プロセッサの回路設計や物理設計では，高い動作周波数をできるだけ短い期間で実現するために，マニュアル設計と自動設計が選択的に適用される．この両者を組み合わせて適用するために，すべてが自動設計であるASIC設計とは異なり，設計手法に多様性が生まれる．本論文では，実際のプロセッサ設計に適用された設計手法をサーベイし，高性能化を実現するための回路設計及び物理設計の設計フローとCADシステムをまとめる．また，最後に，今後のプロセッサ設計における設計手法の展望について述べる．

キーワード　プロセッサ，設計手法，CADシステム，タイミング最適化

1. まえがき

高性能プロセッサは近年のIT（Information Technology）化の中で，パーソナルコンピュータからサーバシステムに至るまで幅広く使われており，その役割はとても大きくなっている．プロセッサの設計は，段階的に見ると，アーキテクチャ設計，論理設計，回路設計，物理設計に分けられる．高性能プロセッサを実現するには，それぞれの段階での研究開発が必要となる．このようなプロセッサの設計は，コンピュータの上でCAD（付録の用語説明の*1を参照．以降，*番号は付録の該当する番号を参照のこと）システムを使って行われる．このCADシステムをどのように実現し，更にどのような手順で適用するかという設計手法は，高性能プロセッサを実現する鍵となる．なお，設計手法という言葉は一般的に使われているが，本論文では文献[1]による“ツールセット＋設計フロー＋制約”という定義に従うこととし，CADシステムや設計フローも設計手法の一部として扱う．また，本論文では，用途に応じて命令セットや演算器などを最適化して処理性能を高める組込みプロセッサの設計ではなく，汎用的な高性能プロセッサの設計に適用される設計手法について述べる．組込みプロセッサの設計手法においては，用途に対するハードウェアの最適化の手法が解決すべき大きな課題となるが，高性能プロセッサの設計手法においては，ハードウェア仕様が決定された後のタイミング設計において，先端的な技術を適用してどのように高性能化を図るかという点も大きな課題となる．

アーキテクチャ設計では，パイプライン段数，キャッシュ容量と制御，バス方式などプロセッサの内部構造を決定する．それに続く論理設計では，回路設計や物理設計の入力となるネットリスト（*2）の設計を行う．回路設計や物理設計の段階に入ると，半導体上で物理的に論理回路を設計するために，半導体を仮想的にコンピュータ上で実現し，その環境の中で設計が行われる．回路設計では，標準的なセル（*3）や大規模なマクロ（*4）の設計を行う．物理設計では，フロアプラン，セル配置，電源設計，クロック設計，配線，タイミング解析，ノイズ解析，電力解析などが行われる．

与えられた期間の中で高性能プロセッサを実現する

† 筑波大学大学院システム情報工学，つくば市
Graduate School of Systems and Information Engineering, Tsukuba-shi, 305-8573 Japan

 © （社）電子情報通信学会 2011

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

ために，CAD システムや設計フローに工夫が必要となる．この工夫が設計手法として体系化され，プロセッサ設計に適用される．一般的には，プロセッサ設計では実現性能が重要視され，ASIC（*5）設計では Time-to-Market が重要視される．プロセッサの設計手法では，標準的なセルを使った設計だけではなく，自由に幅や高さの調整ができるトランジスタを使った設計を人手によって行うことも必要となる．一方，ASIC の設計手法では，標準的なセルのみを使い，しかもできるだけ人手を介さずに自動で設計が行われる．こうした違いの理由は，プロセッサ設計では設計手法の中で設計の自由度を高めて高い性能の実現を目指すからであり，ASIC 設計では逆に自由度を制限して自動設計による設計期間短縮を目指すからである．したがって，両者の設計手法は同一ではない．

本論文では，プロセッサの回路設計及び物理設計段階において高性能化を実現する設計手法を中心に，関連する研究結果についてサーベイする．まず，**2.** で本論文がサーベイ対象とするプロセッサについて述べ，**3.** では高性能化のための設計技術を示し，**4.** では階層設計，**5.** では設計フローについて述べる．**6.** でカスタム設計について述べた後，**7.** ではタイミング最適化のための手法を整理し，**8.** では今後の設計技術と CAD 技術について述べる．最後に **9.** においてまとめを行う．

2. 設計事例

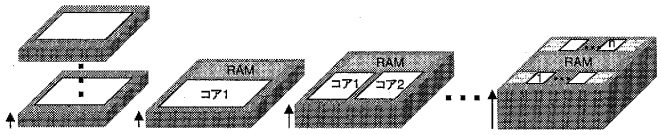
これまで様々なプロセッサが世界で設計されてきており，半導体プロセスの微細化に伴って集積規模が増加し，これに対応する設計手法も変化してきた．ここでは主に 1990 年後半以降に発表されたプロセッサを対象として，その設計手法についてサーベイする．表 1 は，今回のサーベイで調査したプロセッサの一覧である．サーベイ対象としたプロセッサは，CAD システムを含む設計手法が論文として公表されているものとした．

2.1 事例の分類

一つのチップに搭載される機能や規模の観点から，一般的にプロセッサを図 1 のように四つの形態に分類することができる．一つのチップに搭載可能なトランジスタ数だけではプロセッサを実現できない場合には，チップセットとして複数のチップにより一つのプロセッサが構成される．一つのチップに一つのプロセッサのコア（*6）が搭載される場合は，シングルコアと呼ばれる．半導体プロセスの微細化が進み，一つのチップに複数のプロセッサコアの搭載が可能になると，マルチコアと呼ばれるものが出現する．更に集積度が増大するに従い，従来は別のチップとして設計された周辺回路もプロセッサと同じチップに搭載できるようになってきた．この結果，プロセッサコアや周辺回路が IP マクロ（*7）として扱われるようになり，マルチコア SoC（*8）と呼ばれるようになっている．なお，本論文のサーベイでは，マルチコア SoC は調査の対象とはしていない．SoC の設計手法は IP マクロの扱いに関する広い技術範囲を含むためである．

表 1　サーベイ対象プロセッサ
Table 1　Surveyed processors.

プロセッサ	ベンダ	プロセス	動作周波数	コア数	電力	文献
Alpha	旧 DEC	750 nm	299 MHz	1	26 W	[2]
Alpha	旧 DEC	350 nm	600 MHz	1	70 W	[3]
Cell	IBM, 東芝	90 nm	4.0 GHz	8	-	[4], [5]
Cell Broadband	IBM, 東芝	65 nm	~5.2 GHz	8	-	[6]
Itanium2	Intel	130 nm	1.5 GHz	1	~130 W	[7], [8]
Itanium	Intel	65 nm	2.4 GHz	4	-	[9]
POWER4	IBM	180 nm	1.3 GHz~	2	115 W	[10], [11]
POWER6	IBM	65 nm	~5.0 GHz	2	~100 W	[12], [13]
SH-4	日立	250 nm	200 MHz	1	-	[14]
SPARC V9 64b	Sun	150 nm	1.0 GHz	1	80 W	[15]
SPARC V9 64b	Sun	90 nm	1.8 GHz	2	90 W	[16]
SPARC	Sun	65 nm	2.3 GHz	16	~250 W	[17]
SPARC64	富士通	130 nm	1.3 GHz	1	34.7 W	[18], [19]
SPARC64	富士通	90 nm	2.4 GHz	2	~120 W	[20]
SX-9	NEC	65 nm	3.2 GHz	1	-	[21]
S/390 G5, G6	IBM	250 nm	500 MHz, 670 MHz	1	-	[22]
UltraSPARC	Sun	130 nm	1.1 GHz	1	53 W	[23]
UltraSPARC	Sun	130 nm	1.2 GHz	2	23 W	[24]
UltraSPARC T1	Sun	90 nm	1.2 GHz	8	~63 W	[25]
UltraSPARC T2	Sun	65 nm	1.5 GHz	8	~84 W	[26]
Xeon	Intel	65 nm	3.0 GHz~	2	~150 W	[27], [28]
x-86-64 core	AMD	45 nm	2.7 GHz~	4	~105 W	[29]
zSeries	IBM	180 nm	1.0 GHz	1	-	[30]
z900	IBM	180 nm	1.3 GHz	1	-	[31]



(a) チップセット　(b) シングルコア　(c) マルチコア　(d) マルチコア SoC

図 1　規模の面からの分類
Fig. 1　Classification from the point of view of size.

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

2.2 推移と傾向

表1に示した本論文でのサーベイ対象のプロセッサについて，横軸を半導体プロセスのテクノロジー世代，縦軸を動作周波数としてグラフにプロットすると，図2のようになる．グラフの中で，プロセッサ名の後の括弧内に示した数値はコア数を表している．1990年の中頃にチップセットから構成されるプロセッサもあったが[32]，1990年代後半以降の表1のプロセッサは，シングルコアまたはマルチコアとなっている．

2003年のITRS（International Technology Roadmap for Semiconductors）のロードマップでは，高性能プロセッサの動作周波数は2007年には9.3 GHz，2018年には53.2 GHzになると予想されていた[33]．しかし，表1のプロセッサの中で最大の動作周波数は～5.2 GHzであり，9.3 GHzには達していない．また，同じロードマップで2007年の9.3 GHzに対応した消費電力は189 Wとなっているが，表1の大部分のプロセッサでは，最大でも約150 Wである．一般的に，高性能の空気冷却で対応可能な限界の消費電力は約150 Wで，低コストの空気冷却の場合は約40 Wが限界である[34]．こうした状況から，2009年のITRSにおけるSystem Driversの章[35]では，2007年のITRSからプロセッサの動作周波数は4.7 GHzにリセットされ，半導体プロセスの世代ごとに最大でも1.25倍の向上としたと記している．従来は，半導体プロセスの世代ごとにデバイスの性能向上1.4倍，また論理設計での性能向上1.4倍で合計2倍の性能向上と予測していたが，世代の進展に伴う配線ディレイの増大や消費電力の制約から，このような予測の修正を行っている．

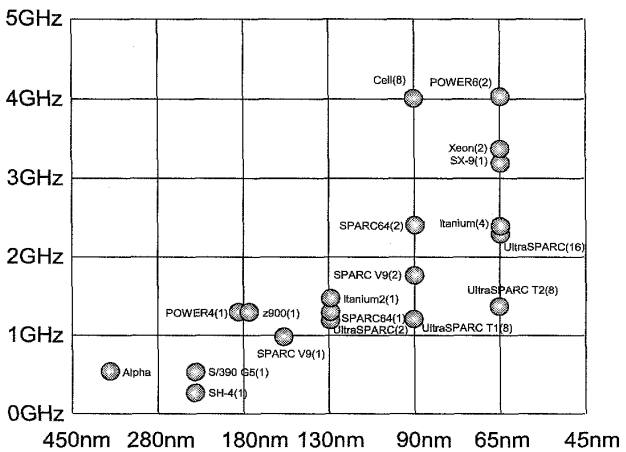


図2 プロセッサ開発の推移
Fig. 2 Trend of processor development.

したがって，液体冷却に比べてコストの低い空気冷却で対応していくには，消費電力を低く抑えることが必要である．消費電力は，動作周波数に比例し，電源電圧の2乗に比例するため，動作周波数及び電源電圧を低くするマルチコア化が始まった[36]．今後は更にこの傾向が進展することが指摘されている[36]～[39]．

3. 設計技術

ASICやプロセッサなどを設計するための設計技術を図3にまとめた．これは，ITRS 2003年版のDesignの章[40]にある技術領域の体系を参考にして，現在の設計技術をまとめたものである．縦軸は，設計からその後の解析，検証，テストのフェーズを表している．一方，横軸は，アーキテクチャ設計，論理設計，回路設計，物理設計という各設計の段階を表している．図3の中で，縦軸が「設計」で，横軸が「論理設計」から「物理設計」の間に太枠で示した「高性能化設計」の内容が，ASIC設計技術とプロセッサ設計技術との間で一番大きな違いがある部分である．また，この高性能化設計は，図中の「タイミング解析」及び「歩留り解析」とも深い関係がある．本論文では，このようなプロセッサ設計における特徴的技術領域についてサーベイしている．

プロセッサ設計では，アーキテクチャと論理が決定された後，回路設計及び物理設計の過程でも，高い動作周波数を実現することが重要である．この高い動作周波数を実現するために，ASIC設計において適用されるセルベースの自動設計だけでなく，人手による設

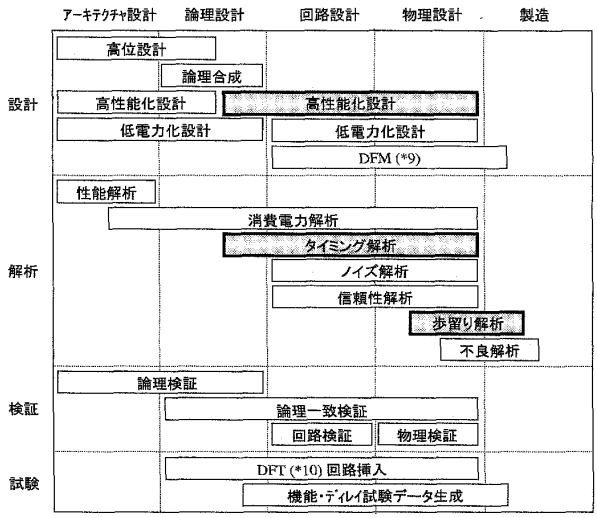


図3 設計技術の分類
Fig. 3 Classification of design techniques.

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

計手法も適用される．

3.1 動作周波数

一般的にチップの動作周波数は，FF（*11）などのすべての記憶素子の間で信号が正しく転送される時間によって決まる．図 4 において，送り側の FF から受け側の FF まで信号がクロックに同期して正しく転送される条件は，式 (1) 及び式 (2) のようになる．

$$T_{clk-s} + T_{data} + T_{setup} \leq T_{clk-r} + T_{cycle} \quad (1)$$

$$T_{clk-s} + T_{data} \geq T_{clk-r} + T_{hold} \quad (2)$$

ここで，T_{cycle} はクロックの周期であり，T_{clk-s} 及び T_{clk-r} はクロックの共通分岐点 P からそれぞれ送り側 FF 及び受け側 FF のクロック入力ピンまでのディレイ（*12）であり，T_{data} は送り側 FF のクロック入力ピンから受け側 FF のデータ入力ピンまでのディレイである．また，T_{setup} は受け側 FF のクロック入力ピンでのクロック変化に先立ってデータ入力ピンでデータが確定しておくべき必要最小限の時間であり，セットアップ時間と呼ばれる．T_{hold} は送り側 FF においてクロック変化後にデータ入力ピンの信号を変化させずに安定させておくべき必要最小限の時間であり，ホールド時間と呼ばれる．なお，文献 [41] では，セットアップ時間とホールド時間の求め方について，精度を高めるために二つの時間の相互依存関係を考慮した方式について解説している．

式 (1) は，一つの目のクロックで送られたデータが二つ目のクロックで正しく受け取られるための条件であり，セットアップ条件と呼ばれる．セットアップ条件では，T_{clk-s} 及び T_{data} は一番遅いケースの値を使い，T_{clk-r} は一番速いケースの値を使う．式 (2) は，本来二つ目のクロックで受け取られるはずのデータが一つ目のクロックで誤って受け取られてしまうことがないための条件であり，ホールド条件と呼ばれる．このホールド条件では，T_{clk-s} 及び T_{data} は一番速いケースの値を使い，T_{clk-r} は一番遅いケースの値を使う．

プロセッサの動作周波数を高めることは，目標の T_{cycle} の値に対して式 (1) の条件を満たすような設計を行うことを意味する．例えば，動作周波数の目標が 2 GHz の場合，T_{cycle} は 500 ps となる．このように，すべての記憶素子の間のディレイが目標以内であるかどうかの解析は，タイミング解析と呼ばれる．

式 (1) を変形すると，式 (3) のように表すことができる．目標性能として与えられた T_{cycle} に対して

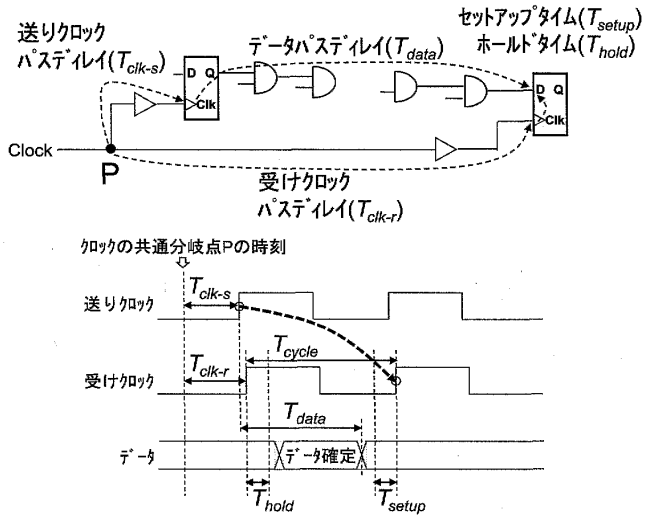


図 4 FF 間の信号の伝送
Fig. 4 Signal propagation between FFs.

式 (3) の条件を満たすためには，$(T_{clk-s} - T_{clk-r})$，T_{data}，T_{setup} のそれぞれの値を小さくすることが必要である．

$$(T_{clk-s} - T_{clk-r}) + T_{data} + T_{setup} \leq T_{cycle} \quad (3)$$

3.2 クロック設計技術

式 (3) における $(T_{clk-s} - T_{clk-r})$ はスキューと呼ばれ，これを小さくするためには，クロック分配回路において，チップ全体にクロックを供給するクロック供給ポイントから各 FF までのクロック到達時間の差を小さくする必要がある．また，T_{setup} を小さくするには，FF の中でクロックのパスのディレイを小さくすることが必要である．

スキュー $(T_{clk-s} - T_{clk-r})$ の項を 0 に近づけるために，プロセッサ設計では H トリー型分配やメッシュ型分配，またその組合せが適用される．これらのクロック分配のアーキテクチャについて，文献 [42] は消費電力を含めて詳しく比較及び分析を行っている．この文献によれば，多くのプロセッサではメッシュ型分配が採用されている．理由は，スキューが H トリー型分配に比べて小さく，製造ばらつきによるスキュー変動も少ないためである．しかし，消費電力の面では，メッシュ型分配の場合には H トリー型分配に比べて最大 30%も増加することが示されている．こうした消費電力の増加を抑えるため，FF のクロック入力に接続する配線をできるだけ短くして充放電の量を小さくするように FF の配置を改善する手法も提案されている [11]．

また，T_{clk-r} が T_{clk-s} より大きくなるように意

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

図的に設計することにより，($T_{clk-s} - T_{clk-r}$) の値は更に小さくなる．この手法はインテンショナルスキュー [43], [44] と呼ばれる．文献 [43] は，製造ばらつきによって目標周波数で動作するチップの割合が低下することをインテンショナルスキューで防止するという応用について述べている．一方，文献 [44] は，クロック分配回路の中にチップの外からスキューを変更できる回路を入れておき，実際のチップの製造後にクリティカルパス (*13) のディレイを改善するようにスキューを調整する手法について述べている．このインテンショナルスキューの手法を使った場合には T_{clk-r} の値が大きくなるので，この値が T_{clk-s} となる次段のパスでは逆に制約が厳しくなるので，次段のパスにディレイの余裕がある場合に適用することが可能となる．

クロック信号は非常に高い周波数で動作しているため，レイアウトの面で近くにある信号にクロストークによるノイズを与えないように設計を行う必要がある．特に，プロセッサ設計では，動作周波数が GHz のレベルであるため，シールド配線によるノイズへの対策が特に重要となる [8], [12], [16], [20].

3.3 高性能化のための技術

T_{data} を更にステージディレイに分解することができる．ステージディレイは，図 5 に示すように，パス上のセルの入力ピンから次段のセルの入力ピンまでのディレイと定義される．

ステージディレイは，更にセルディレイ T_{cell} と配線ディレイ T_{wire} に分割することができる [45], [46]. T_{cell} は，出力ピンに負荷が付いた状態において，入力ピンでの波形の電圧 50%ポイントから出力ピンでの波形の電圧 50%ポイントまでのディレイである．T_{wire} は，セルの出力ピンでの波形の電圧 50%ポイントから負荷となっているレシーバのセルの入力ピンでの波形の電圧 50%ポイントまでのディレイである．

T_{cell} は，そのセルに入力される波形のスルー (*14) $T_{slew-in}$，セルを構成する各トランジスタのゲート幅 W_g，しきい値電圧 (*15) V_{th} 及び出力ピンから見える負荷を表す実効容量 (*16) C_{eff} の関数となる．C_{eff} には，レシーバ側にある遠端の容量がその手前にある抵抗により小さく見える遮へい効果 (resistive shielding [47]) が含まれる．W_g が大きくなるにつれてセルの出力抵抗が小さくなるため T_{cell} が小さくなる．V_{th} が小さくなるにつれて出力の応答が早く始まるため T_{cell} は小さくなる．また，$T_{slew-in}$ と C_{eff} がそれぞれ大きくなれば T_{cell} は大きくなる特性をもつ．

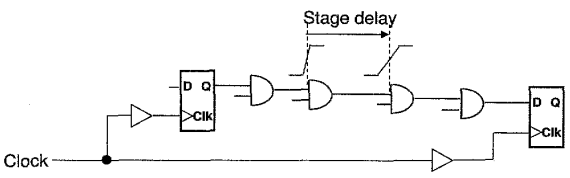


図 5 ステージディレイ
Fig. 5 Stage delay.

T_{wire} は，セルの出力波形のスルー及び負荷特性 (ファンアウト数や各ファンアウトまでの配線の長さや形状などによる抵抗 R と容量 C の分布) の関数となる．この関数は，直感的には R と C の積に比例する．したがって，負荷の R 及び C を小さくすることにより，T_{wire} を小さくすることができる．また，T_{wire} を小さくすることは，C_{eff} を小さくすることになるため，T_{cell} も小さくすることができる．

一つのパスに送り側の FF も含めて N 段ある場合の T_{data} は式 (4) のように表すことができる．

$$T_{data} = \sum_{i=1}^{N} (T_{cell,i} + T_{wire,i}) \tag{4}$$

したがって，T_{cell} や T_{wire} を小さくして動作周波数を上げるための一般的な手法として，以下のような (a) から (h) までのものがある．なお，**7.** において，プロセッサの回路設計や物理設計において特に重要となる下記の (d), (f), (g) について更に詳述する．

(a) 論理段数を少なくする [48]

並行して別々に開発された二つのブロックの間を通過するパスがクリティカルパスになった場合など，そのパスを全体として論理の再構築を行うことにより，論理段数を減らすことができる可能性がある．

(b) ファンアウトを最適化する [11], [49], [50]

ファンアウト数が多い場合，また少ない場合でもファンアウト先の個々のセルの配置の位置が離れている場合，ファンアウトを分割することにより一つのセルの出力負荷を小さくすることができる．

(c) セル間の距離を最適化する [11]

一つのパス上に複数のセルがあり，それらが既に配置されている場合，一つのセルの出力ピンから次のセルの入力ピンまでの距離は，パス全体としてなるべく均一になるようすることにより最適化の可能性がある．

(d) リピータ (*17) を入れる [51], [52]

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

セルの配置が終了した段階で，一つのセルの出力ピンから次のセルの入力ピンまでの距離が長い場合，途中にリピータを入れることによりディレイを小さくできる可能性がある．出力ピンに接続される配線への充放電に要する時間がディレイの大きな割合を占めており，距離が長い場合に途中にリピータを入れることにより，リピータも充放電に寄与できるためである．ただし，距離が短い場合にリピータを入れると，リピータを通過するディレイの方が大きくなってディレイを逆に大きくする場合もあるので注意が必要である．

(e) 隣接配線の間隔を広げる [53]～[55]

既にセルの配置が完了し，ネット（*18）の配線を行う場合，クリティカルパス上のネットの配線では，その配線の片側または両側に他の配線が通常より近づかないように間隔を確保することにより，隣接配線による容量を小さくできるため，ディレイを小さくすることができる．

(f) 上位配線層で配線する [56], [57]

上位配線層になるほど，配線の幅・高さ・間隔がそれぞれ大きくなる．そのため，配線抵抗は上位配線層の方が小さい．容量に関しては隣接容量なども含めると，配線層によらずほぼ同じになる．したがって，上位配線層で配線した方が，下位配線層で配線するよりもディレイを小さくすることができる．

(g) セルのサイズを最適化する [58]

一つのセルが駆動する負荷が大きいとき，そのセルのサイズを大きくすることにより出力の抵抗が小さくなる．この結果，負荷を充放電する時間が短くなるために，ディレイを小さくすることができる．また，負荷側のセルのサイズをもし小さくすることが可能であれば，負荷となるセルの入力ピン容量が小さくなるため，ディレイを小さくすることができる．

(h) Low-V_{th} セルを使う [59], [60]

V_{th} とは，トランジスタのしきい値電圧である．この V_{th} が低い Low-V_{th} セルは標準 V_{th} のセルより動作が高速であるが，漏れ電流（*19）は大きい．逆に，V_{th} が高い High-V_{th} セルは標準 V_{th} のセルより動作が遅いが，漏れ電流は小さい．性能と電力を考えながら，Low-V_{th} セルと High-V_{th} セルを効果的に使い分ける必要がある．

T_{data} を小さくする際に，面積・電力が制約として付加される．このような制約のもとで，すべての記憶素子の間のパスに対して，式 (3) の条件を満たすように設計する必要があるが，そのために許される設計期間も設定される．つまり，許された設計期間の中で，目標とする動作周波数を実現する必要がある．

こうしたすべての制約の中で，高い動作周波数を実現するために，チップ全体を複数のブロックに分割して，各ブロックに自動設計からマニュアル設計までの最適な設計手法を適用して，並列に設計を行う．したがって，高い動作周波数実現が主目的となるプロセッサ設計では，個々の高性能化技術のほかに，階層設計，設計フロー，カスタム設計が重要な役割を果たす．

4. 階層設計

4.1 基本概念

プロセッサ設計においては，許された期間の中で高い動作周波数を実現するため，設計作業の並列化，最適な設計手法の選択，設計変更時の影響の局所化の 3 点を考慮する必要がある．このような 3 点を実現するために，チップは分割され，階層化される．一つの階層で複数のブロックに分割され，分割された一つのブロックが更に複数のブロックに分割されるため，その全体の構造を階層と呼ぶ．この節では，物理設計における階層設計について説明し，論理設計における階層設計との関係は，**4.3** において説明する．

階層設計の例として，図 6 は二つのコアを搭載した一般化したプロセッサの構造を示したものである．同一のプロセッサコアが 2 個，キャッシュRAM（*20）を含む領域，またそれらの間を制御するコア・キャッシュ制御領域があり，チップの外部と信号をやり取りする I/O 領域から構成される．つまり，この例では，一つのチップは，6 個のブロックに分割されている．また，プロセッサコア 1 は更に三つのブロック 1 から 3 に分割され，ブロック 3 は更にブロック 3-1 及び 3-2 の二つに分割されている．この例では，ブロック 3-1

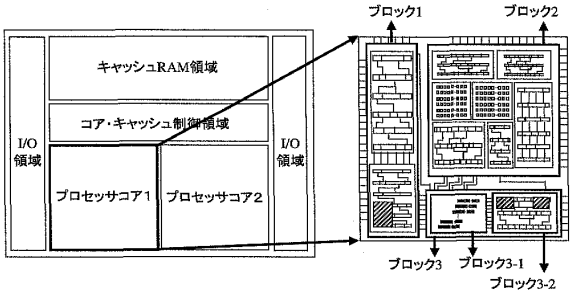


図 6 階層設計
Fig. 6 Hierarchical design.

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

や 3-2 は，更にブロックに分割されることなく，セルやマクロから構成される．

このように，一つのチップが複数のブロックに分割され，それぞれのブロックは更に複数のブロックに分割されているので，チップを階層のトップとした物理的な階層構造が作られている．階層のトップであるチップ自身に対してもブロックが定義され，このブロックをチップトップと呼ぶ．この例の場合，チップトップ，プロセッサコア，ブロック 3，ブロック 3-1 と階層数は 4 となる．階層数や同一階層の中でのブロックの分割数は，プロセッサ設計の事例ごとに異なる．

設計手法として，フルカスタム設計，セミカスタム設計，自動設計という 3 種類の手法がある．フルカスタム設計はトランジスタレベルでネットリストを設計し，レイアウト（*21）も人手で行う手法である．セミカスタム設計はネットリストをトランジスタレベルまたはセルベースで設計し，レイアウトの一部などに自動設計を適用する手法である．フルカスタム設計及びセミカスタム設計の両方を合わせてカスタム設計と呼ぶ．自動設計は，論理は RTL（*22）で記述し，論理合成ツールによって RTL 記述からネットリストに自動変換し，その後で物理合成ツールによってネットリストを自動的にレイアウトを行う方法である．各プロセッサの設計において，どのようなスタイルの設計が適用されているかをまとめたものが表 2 である．なお，自動，セミカスタム，フルカスタムの欄に数値が入っているものは，全体の中でのそれぞれの適用比率を示している．

表 2　設計スタイル
Table 2　Design style.

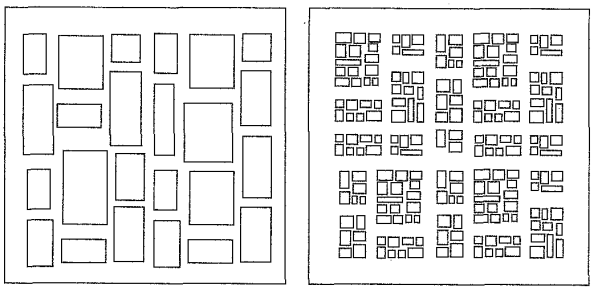
プロセッサ	ベンダ	自動	セミカスタム	フルカスタム	階層数	文献
Alpha	旧 DEC	10%	—	90%	—	[2], [3]
Cell	IBM, 東芝	40%	—	60%	5	[4]
Cell Broadband	IBM, 東芝	○	—	○	3	[6]
Itanium	Intel	○	○	—	—	[9]
POWER4	IBM	○	○	○	4	[10]
POWER6	IBM	○	—	○	4	[12], [13]
SH-4	日立	○	○	○	2	[14]
SPARC V9 64b	Sun	○	—	○	2	[15], [16]
SPARC	Sun	○	○	○	—	[17]
SPARC64	富士通	—	○	○	4	[18]〜[20]
SX-9	NEC	○	○	—	2	[21]
S/390 G5, G6	IBM	○	○	○	3	[22]
UltraSPARC T1, T2	Sun	○	—	—	4	[25], [26]
Xeon	Intel	○	—	○	2	[27]
x-86-64 core	AMD	○	○	○	—	[29]
z900	IBM	○	○	○	3	[31]

フルカスタム設計，セミカスタム設計，自動設計という 3 種類の適用の割合は，プロセッサ設計における目標性能，設計期間，人的リソースなどから決められる [61]．Alpha プロセッサ設計では，全体の約 90%がフルカスタム設計されていた．Cell プロセッサ設計でも，フルカスタム設計が約 60%となっている．また，SPARC64 プロセッサ設計ではネットリスト設計及びセル配置には自動設計は適用されていない．一方，UntraSPARC T1，T2 プロセッサ設計では大部分が自動設計で，カスタム設計は RAM，IO 及びアナログ回路に限定されている．また，AMD の x86-64 プロセッサコアの設計では，自動設計とフルカスタム設計の中間であるセミカスタム設計を主に適用している．

一般的には，プロセッサ設計における目標動作周波数に対して，その動作周波数に影響をもつ部分に優先的にフルカスタム設計が適用され，論理に変更が発生しやすい部分には自動設計が適用されることが多い．自動設計を適用したにもかかわらず目標の動作周波数を実現できない場合には，クリティカルパスの受け側 FF から入力側に論理をトレースして，送り側 FF に到達するまでの範囲の論理を切り出し，その部分をカスタム設計し直したものに置き換えるなどの手法を使っている場合もある [9], [62]．

4.2　ブロックサイズ

ブロックのサイズにより図 7 の (a) 及び (b) の二つのタイプに分類することができる．ここで，(a) のブロックサイズ大は，チップが数十個までのブロックで構成される場合を意味し，(b) のブロックサイズ小は，チップが数百から数千までのブロックで構成される場合を意味する．高い動作周波数を実現する際には，ブ



(a) ブロックサイズ大　(b) ブロックサイズ小

図 7　ブロックサイズの種類
Fig. 7　Types of block sizes.

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

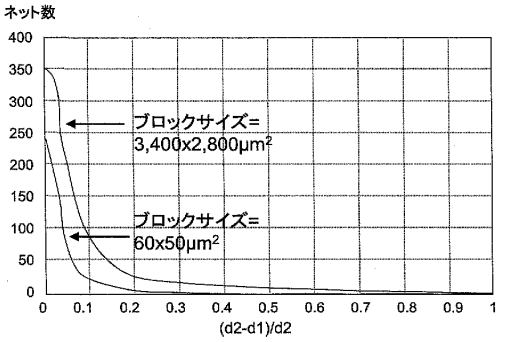


図 8　異なるブロックサイズでの d1/d2 の分布 [63]
Fig. 8　d1/d2 distributions of different block sizes [63].

ロックサイズについての考察が重要となる．これは，ブロックサイズの大小に応じて，配線ディレイの考慮の手法が異なってくるからである．

図 8 は，180 nm の半導体プロセスを使った実際の設計データを用いて，ブロックサイズと配線ディレイの関係を示したものである [63]．サイズの異なる二つのブロックにおいて，ドライバ及びそれに接続されるレシーバの両方が同じブロックに含まれ，しかもレシーバが一つだけの 2 ピンのネットについて，それぞれの配線ディレイの割合の分布を比較評価する．

配線の抵抗 R 及び容量 C の両方を無視してレシーバの入力ピン容量のみを考慮したときのステージディレイを d1 とし，配線の R 及び C の両方も考慮したときのステージディレイを d2 とする．ステージディレイは，ネットのドライバとなるセルの入力ピンからネットのレシーバとなるセルの入力ピンまでのディレイである．ステージディレイにおいて配線ディレイが占める割合とその度数の対応を図 8 に示す．図 8 は，$(d2-d1)/d2$ を横軸に，度数を縦軸にした分布グラフである．

$(d2-d1)/d2$ は 0 から 1 の範囲で分布する．0 に近いところは配線ディレイの影響がほとんどないことを意味し，1 に近づくにつれて配線ディレイの影響が大きくなることを示す．なお，図 8 に対応したブロックの諸元を表 3 に示す．

図 8 及び表 3 により，ブロックサイズが小さくなるほど，ブロック内での配線ディレイの影響は小さくなることが分かる．つまり，**3.3** の高速化技術として挙げた（a）から（h）までの技術の中で，ブロックサイズを小さくすることにより，配線ディレイの改善を目的とした（c）から（f）の技術については，ブロックの中での考慮を不要とすることも可能である．しかし，ブロックサイズが小さい場合，チップ上でブロックを配置し，ブロック間を配線するチップ組立て（chip assembly）の設計において，ブロック間をまたがるパスのディレイを最適化する必要がある．

表 3　ブロックの諸元 [63]
Table 3　Profile of each block [63].

ケース	ブロックサイズ（μm^2）	セル数/ブロック	ブロック数
1	3,400 × 2,800	70,000	2
2	60 × 50	20	7,200

ブロックサイズを大きくする設計の場合（図 7 の (a) のタイプ）は，ブロック内を自動設計することを意図するケースが多い．Itanium [64]，SH-4 [14]，SPARC V9 64b [14] では，ブロック数は数十個のレベルである．また，Xeon [27] でも，ブロックのサイズを意図的にできるだけ大きくしている．このような大きいサイズのブロックに自動設計を適用する場合，自動設計ツールはタイミングを最適化しやすい．しかし，ブロックのサイズが大きくなる場合は配線によるディレイを考慮する必要があるが，自動設計ツールでは **3.3** の（f）に示した上位配線層の有効な利用が難しいため，自動設計においても上位配線層を考慮する取組みが近年行われている [56], [65], [66]．

ブロックサイズを大きくする別のケースとして，一つのブロック内の物理設計においてカスタム設計を適用することにより，リピータ挿入や上位配線層利用を一緒に考慮する場合もある．ブロックのサイズは，論理機能をベースにカスタム設計を並列に進めることを考慮して決定される．このような手法は SPARC64 プロセッサの設計で適用されており，ブロック数は数十である [19], [67]．

ブロックサイズを小さくする設計の場合（図 7 の (b) のタイプ）は，一つのブロック内の物理設計では，**3.3** の（d）及び（f）に示したリピータ挿入や上位配線層利用を考慮することのない自動設計適用を意図している．つまり，自動設計が苦手な機能を使うことなく自動設計が適用できる．この場合，リピータ挿入や上位配線層利用は上位階層のチップトップでのチップ組立て設計において，リピータ挿入や上位配線層利用のための専用ツールを開発して適用する．z900 プロセッサでは，物理的な階層構造の中でマクロが一番小さいブロックであり，数百のマクロから構成されている [31]．POWER4 プロセッサでは，1,015 種類のマクロがチップ全体で合計 4,341 個配置されている [10]．また，POWER6 プロセッサでは，1,165 種類のマク

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

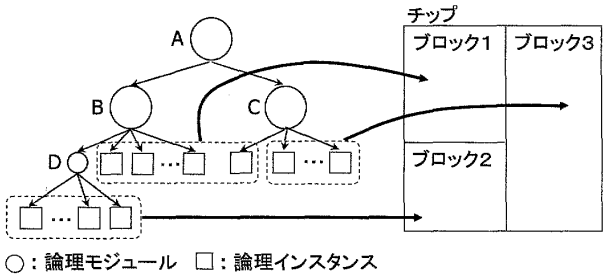


図 9 論理と物理のブロック境界
Fig. 9 Boundaries of logical and physical blocks.

ロがチップ全体で合計 11,998 個配置されている [13].

4.3 論理境界と物理境界の関係

論理設計では，システム構成が分かるように論理機能のまとまりを意識して論理階層を作り，それぞれの階層で論理を RTL またはネットリストで記述する．一方，物理設計では，チップ内での信号の流れやクリティカルパスの配線がどこを通るかなどを意識して物理的なブロック分割を行い，それに基づいた物理階層を作る [68]～[70]．物理設計の最初に位置づけられるフロアプランでは，物理階層の各ブロックへの論理の割付けを行う．物理階層では，分割された単位をブロックと呼ぶこととし，論理階層では，分割された単位をモジュールと呼ぶこととする．

図 9 は，論理を物理階層のブロックに割り当てる様子を示している．論理は，モジュール A がトップのモジュールであり，そのモジュール A はモジュール B と C から構成されている．モジュール C は，論理インスタンスのみから構成される．論理インスタンスは，物理設計におけるセルやマクロに対応している．モジュール B は，論理インスタンス及びモジュール D から構成される．モジュール D は，論理インスタンスのみから構成される．

この論理を物理的なチップ上に配置するために，チップに対して物理的な階層構造を作る．図 9 の例では，チップがブロック 1 から 3 までの三つのブロックに分割される．ブロック 1 に対しては，モジュール B に含まれる論理インスタンスのすべてと，モジュール C の中の一部の論理インスタンスを割り付ける．ブロック 2 に対しては，モジュール D に含まれる論理インスタンスのすべてを割り付ける．ブロック 3 に対しては，モジュール C に含まれる論理インスタンスの中で，ブロック 1 に割り当てられなかった残りの論理インスタンスが割り当てられる．

ここで，論理階層の境界と物理階層の境界の関係について考える．ブロック 2 に対しては，モジュール D に含まれる論理インスタンスのすべてを割り付けているため，モジュール D の入出力とブロック 2 の入出力が一致している．このような場合，論理境界と物理境界が一致しているという．これに対して，ブロック 1 では，モジュール B の論理の一部とモジュール C の論理の一部が割り当てられているために，ブロック 1 の入出力はモジュール B の入出力と一致しないし，モジュール C の入出力とも一致しない．このような場合，論理境界と物理境界が一致していないという．

論理境界と物理境界を一致させる場合は，一致している単位で論理と物理の一致検証である LVS (*23) を実行することができるという長所をもつ．機能のまとまりを物理的にも一つのマクロとして作成する場合には，このように両者の境界を一致させる手法がとられる [10]．短所としては，最適なレイアウトを阻害する可能性があるという点である．例えば，文献 [71] では，論理をチップ上でレイアウトする際，物理的階層を作らずにフラットにしてレイアウトした結果に対して，各論理階層の中の論理がチップ上のどこにレイアウトされているかを分析した．その結果，一つの論理階層の論理がチップ上で必ずしも同じ領域にまとまってレイアウトされているとは限らないことが示されている．

これに対して，実際の設計では，論理境界と物理境界を一致させない手法がとられることもある [2], [72]．物理境界を論理境界とは無関係に設定できるようにすることにより，個々のブロックでのレイアウト作業を他のブロックとは独立に進めることができる独立性をもちながら，論理モジュールを変更することなく容易に一つのブロックから別のブロックへ論理を部分的に移動することが可能となる点が長所となる．レイアウトの段階において，ブロックのセル密度調整やリピータの最適配置などが可能となる．短所としては，ブロック単位で LVS の実行ができず，境界が一致するところまで検証の単位を拡大する必要がある点である．なお，一般的な ASIC の設計では，ブロック分割の目的がブロックごとの並列設計ではなく，ブロックのサイズや配置位置の最適化が目的であるため，論理階層と物理階層の一致からスタートするが，最終的には両者の境界が一致しないことを許容している [73], [74]．

更に，もう一つの手法としては，論理階層の境界と物理階層の境界は一致させるが，レイアウト設計の際に，論理階層の境界や物理階層の境界とは全く無関係

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

にチップの一部分を切り出してレイアウトを修正できる手法を適用する場合もある．論理とは関係しないダミーメタル（*24）の挿入の際に，この手法を適用して処理を高速化する例が文献 [21] に紹介されている．

5. 設計フロー

5.1 フローの分類

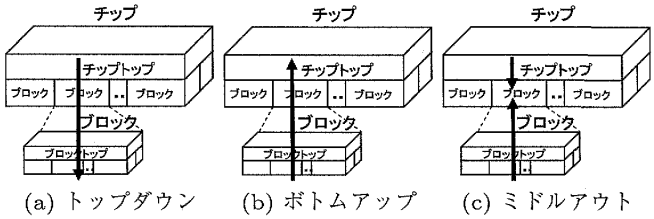
プロセッサ設計においては，動作周波数を高めることが重要な設計目標の一つである．物理的な階層設計を行う際，どこから設計に着手するかということに関して，図 10 に示すようにトップダウン（top-down），ボトムアップ（bottom-up），またその両方を取り入れたミドルアウト（middle-out）という 3 種類の手法がある [75]．

トップダウン設計では，チップ全体のフロアプランを行った後，ブロックの境界に面積やタイミングなどに関する適切なバジェット（*25）を作成し，図 10 の (a) の矢印のように下位階層の方向に設計を進めるために，各ブロックにおいてはバジェットを守った設計が行われる．ボトムアップ設計では，下位階層のブロックからレイアウト設計を行い，各ブロックのレイアウトが終わった後，図 10 の (b) の矢印のように上位階層で各ブロックの最適な配置が行われる．このボトムアップ設計は，ビルディングブロック設計と呼ばれることもある．

トップダウン設計及びボトムアップ設計には，それぞれ長所と短所がある [76], [77]．トップダウン設計における長所は，チップ設計の全体を把握しながら設計を進めることができるので，全体での設計のやり直しが発生しないことである．逆にその短所は，チップをブロック分割しながら設計を進める際に，面積やタイミングなどのバジェットを作成する必要があるが，この作成が簡単ではないことである．一方，ボトムアップ設計の長所と長所は，トップダウン設計の場合の逆になる．

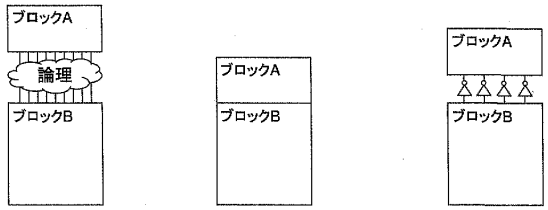
プロセッサ設計では，トップダウン設計及びボトムアップ設計の両者の長所を生かして高い性能を短期間で実現するために，図 10 の (c) の矢印のようにトップダウン設計とボトムアップ設計を並行して行いながら設計を収束させていくようなミドルアウト設計になることが多い [67], [75]．

なお，チップ上でのブロックの配置について，ブロック間の距離により，図 11 のように 3 種類の手法に分類できる [78]．(a) のチャネル（channeled）方式は，ブロック間を広く空けて配置し，その空間を一般論理のセルの配置に利用する．この方式は，一般的な ASIC で使われる手法であり，フロアプランやチップ全体でのクロック分配回路の配置などが容易になるという利点があるが，チップ面積は大きくなる傾向にある．(b) のアバットメント（abutted）方式では，ブロック間に隙間はなく，二つのブロックの境界が重なるように配置され，しかもブロック間で接続する信号のピンは互いに同じ位置と同じ配線層で重なるように配置される．この方式では，ブロック境界部分で両側のブロックのピンを位置及び配線層ともに一致するように設計する必要があるため，物理設計が難しい．また，チップ全体でのクロック分配回路の配置なども難しくなる．しかし，面積が小さくできる点や，ブロック間の距離を小さくできるなどの利点があるため，高性能なプロセッサなどに適用される [4], [5], [18], [19], [25], [28]．(c) のニアチャネル（near channeled）方式は，ブロック間に少し空間を確保し，チャネル方式及びアバットメント方式の両方の長所を取り入れている．この方式は，高性能なプロセッサにおけるリピータの配置などに適用される [64], [79]．



(a) トップダウン　(b) ボトムアップ　(c) ミドルアウト

図 10　階層設計フローの種類
Fig. 10　Types of hierarchical design flows.



(a) チャネル方式　(b) アバットメント方式　(c) ニアチャネル方式

図 11　ブロック配置の種類
Fig. 11　Types of block placements.

5.2 プロセッサの設計フロー

図 12 は，プロセッサ設計の典型的なフローである．論理設計では，論理を入力し，論理を検証する．物理設計では，フロアプラン，セル配置，電源設計，クロック設計，配線，タイミング解析，ノイズ解析，電力解析などを行った後，製造データを GDS（*26）という

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

ファイル形式で作成する．その後，製造時に良品・不良品を判定する診断データの生成も行われる．タイミング解析の後，ディレイがまだ目標に到達していないパスが存在する場合，それら個々のパスについてディレイを繰り返し改善していく．

図 12 のフローの物理設計部分について，ブロック単位及びチップ全体でのそれぞれの設計作業を時系列に並べると表 4 のようになる [19], [22]．表 4 におけるチップ全体及びブロック単位の設計作業は並行して行われることが多く，ミドル・アウトの設計フローとなる．

フェーズ 1 ではチップ全体のフロアプランやクロック分配プランが行われ，それと並行して設計が先行するブロックでのセルの初期配置及びタイミング解析が行われる．フェーズ 2 では，目標の動作周波数を実現するために，ブロック単位でのレイアウト設計とタイミング解析が繰り返し行われ，更にチップ全体のレイアウト設計とタイミング解析が並行して行われる．フェーズ 3 では，ブロック単位とチップ全体で平行して，タイミング解析とともにノイズや電源網など各種検証が行われる．こうした検証の後，ブロック単位で DRC (*27)・LVS の実行を行って，検出されたエラーについて対策を実施する．また，このときにチップ全体では DRC・LVS の実行トライアルが開始される．最終フェーズでは，チップ全体での DRC・LVS の実行を行って，検出されたエラーについて対策を行い，最後に製造データを作成する．

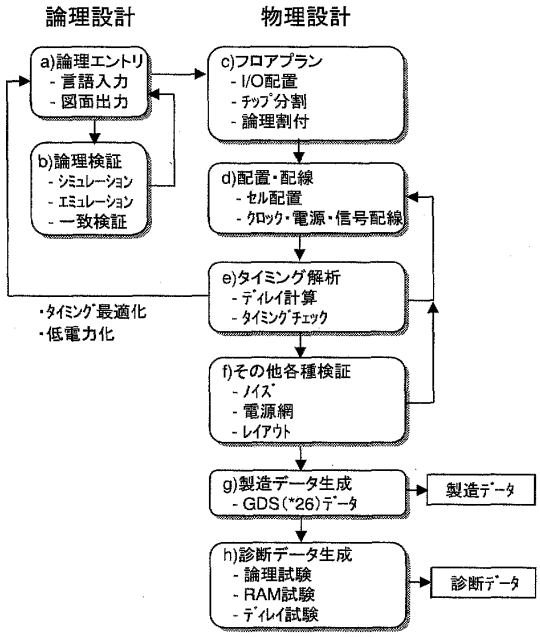


図 12　典型的な設計フロー
Fig. 12　Typical design flow.

表 4　時系列の設計項目
Table 4　Design along time frame.

フェーズ	チップ全体	ブロック単位
フェーズ 1	・フロアプラン ・クロック分配プラン	・初期セル配置ベースの タイミング解析
フェーズ 2	・レイアウト設計 ・タイミング解析	
フェーズ 3	・各種検証 - タイミング - ノイズ - 電源網	
	・DRC・LVS トライ	・DRC・LVS 実行
最終フェーズ	・DRC・LVS 実行 ・製造データ生成	-

5.3　階層間での配線層共有

チップが複数のブロックから構成される場合，一般的には，すべてのブロックに対して接続されるグローバルな電源配線やクロック配線はチップトップで行われる．この場合，チップ上の平面的座標を (x, y) とし，配線層番号を z としてチップを三次元空間 (x, y, z) として表現すると，座標 (x, y, z) にある配置・配線の領域をチップトップ及び下位階層のブロックで共有することが可能となる．このとき，図 13 に示すような共有空間は両者から見える必要がある．両方の階層から見えるようにする手法として，以下の (a) 及び (b) の二つの手法がある．

(a) アブストラクト (abstract) とカバー (cover)

一般的に，アブストラクトとカバーという方法によって，共有される配置・配線領域の情報を他の階層からも参照できるようにする [10]．カバーは，上位階層で使われている配置・配線領域を下位階層に知らせるための伝達であり，アブストラクトはセルや下位

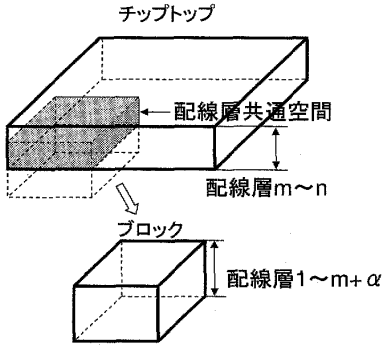


図 13　配線層空間の共有
Fig. 13　Share of routing layer space.

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

階層で使われている配置・配線領域を上位階層に知らせるための伝達である．こうした手法では，伝達される主な情報は，配置・配線領域だけでなく，配線かビア（*28）かを区別する識別情報などを含んでいる．また，アブストラクトやカバーで伝達される配置・配線領域を誰が利用することが可能かを示すために，mine/yours/nobody などの識別情報も一緒に付与するケースもある [13].

(b) 透明な共有空間

共有空間にある他の階層の情報が，どの階層からも見える手法である [19]．この手法を実現するために，上位階層のデータベースと下位階層のデータベースの間で，セル情報，ネット情報，配線情報の伝達が相互に行われる．伝達された情報は，伝達先の階層の中で通常のネットやセルとは区別して参照することができ，どの階層からきたのかという情報とともに管理される．これにより，チップを複数のブロックに分割した並行設計において，他のブロックと共有する部分にある他のブロックにおける最新の配線状況を詳細に見ながら設計を進めることができる．例えば，ブロック内でノイズを意識した配線設計を行っている場合，チップトップで既に使われている配線チャネル（*29）の場所だけでなく，配線が属しているネットの名前及びそのネットに属する配線の幅や経路も見ながら配線設計が可能となる．

5.4 下位階層ブロックのタイミングのモデル化

あるブロックの設計において，ブロック単独で行うタイミング解析では，トランジスタレベルで設計されたブロックはトランジスタレベルで解析される．セルベースで設計される場合には，セルベースで解析するケースとトランジスタレベルで解析する二つのケースに分かれる．セルベースで設計しているにもかかわらずトランジスタレベルでタイミング解析を行っているのは，POWER シリーズ及び Itanium 2 の設計においてである．この理由は，セルベース解析との混在を避けることによってタイミング解析精度を確保するため [10], [13]，タイミング解析の精度確保と半導体プロセス変更時の再実行の容易さのため [8]，セルベース設計後にトランジスタサイズ最適化を行うため [13] などである．

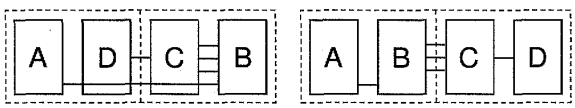
タイミング解析が終了したブロックに対して，そのブロックを含む上位階層のブロックでタイミング解析を行うためのタイミングのモデル化については，二つの選択肢がある．一つ目の選択肢は，ブロックに対して一般のセルのモデル化と同様に，ブロックの各入力ピンから出力ピンまでのディレイをタイミングライブラリとして作成する方法である [2], [7], [10], [13]．二つ目の選択肢は，ブロック内をセルまたはトランジスタのレベルまで展開してタイミング解析を行う方法である [19]．タイミングライブラリを作成する方法が一般的であるが，解析精度を高める場合などにはブロック内を展開して解析が行われる．

6. カスタム設計

カスタム設計とは論理回路を実現するネットリストを設計者が設計し，配置や配線も設計者が人手で行う設計手法である．文献 [80] では，カスタム設計は，回路のスタイルやトポロジー，トランジスタのサイズ，またトランジスタやその間の配線などの物理設計に対しての制御である，と定義している．

自動設計よりカスタム設計の方が高性能化できる理由は，様々説明されている．文献 [81] では，設計のそれぞれの段階に存在する不確実さに対処してそれを最小化していく能力は，自動設計より人手によるカスタム設計の方が勝るからと説明している．文献 [82] では，自動設計は与えられた条件の解空間の中での検索であり，自分で解空間を広げるなどのイノベーションをしないからと説明している．また，文献 [83] では，カスタム設計は機械による科学ではなく，人間による芸術だからであると説明している．

例えば，自動設計が必ずしも最適な解を導かない簡単な例を図 14 に示す．これは文献 [84] に示されている例である．4 個のセル A，B，C，D からなるネットリストが与えられ，点線で囲まれた一つの領域には 2 個のセルだけが入るケースで，領域間を横切る接続の本数を小さくするという目標関数で最適化される自動配置では，(a) のような結果が得られる．しかし，セル間接続の距離の合計が小さい方が高性能化の面では好ましいため，実際には (b) の方がより良い配置となる．(b) のような結果を自動配置で得るためには，す

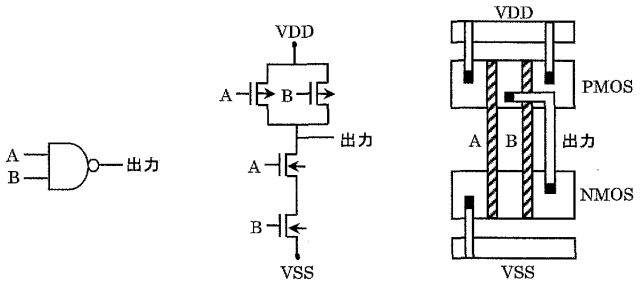


(a) 自動配置の結果　(b) 最適な配置

図 14　最適な配置 [84]
Fig. 14　Optimal placement [84].

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers



(a) 素子モデル (b) トランジスタモデル (c) レイアウトモデル

図 15 トランジスタレベルの設計
Fig. 15 Transistor-level design.

表 5 トランジスタの 4 種類の調整
Table 5 Four kinds of transistor tuning.

調整対象	内容と効果
パワーレベル	すべてのトランジスタのサイズを一律に変更する．サイズを大きくすることにより，抵抗が小さくなり，駆動能力が高くなる．
ベータレシオ	VDD 側に並列に接続される pMOS と VSS 側に直列に接続される nMOS のトランジスタのサイズを変えることにより，立上りと立下りのディレイを変えることができる．
テーパレシオ	VSS 側に直列に接続される nMOS のトランジスタのサイズを VSS 側から順に小さくすることで，直列の VDD に近い側の入力ピンから出力ピンのディレイを小さくすることができる．
V_{th}	すべてのトランジスタの V_{th} を低い V_{th} セルにすることにより，トランジスタの入力から出力までのディレイを小さくできる．その代わり，漏れ電流が大きくなる．

べてのセル間接続の距離の合計も目標関数に含める必要がある．実際のケースでは，**3.3** の (a) から (h) までの手法を組み合わせながら最適解を見つける必要があるため，現状では熟練した設計者によるカスタム設計の方が自動設計より高性能化実現に向いている．

トランジスタレベルでネットリストを設計し，レイアウトも人手で行うフルカスタム設計では，図 15 の (a) の素子モデルは使わずに，ネットリスト設計では (b) のトランジスタモデルを使用する．また，レイアウト設計も (c) のようにトランジスタレベルでレイアウトを行う．

例えば，トランジスタレベルで NAND のセルを設計する場合，ディレイを小さくして高性能化を実現する方法として，表 5 のような 4 種類の調整対象を組み合わせることができる [30], [31], [85]．標準的に用意されたセルを使って高性能化を図る場合，**3.3** の (a) から (h) までの手法に限定されるが，トランジスタレベルで設計を行うフルカスタム設計の場合，更に表 5 のような手法を組み合わせた最適化が可能になる．しかし，設計期間はその分だけ長くなる．なお，表 5 に示すようなトランジスタの最適化を支援する CAD 技術も実用化されている [86]〜[88]．

ここで，カスタム設計と自動設計において，その設計期間や実現性能についてどのくらいの差があるかについて，紹介する．文献 [89] では，ある一つのブロックの設計に要する期間と人的リソースそれぞれについて，カスタム設計と自動設計を比較している．カスタム設計では 8 人で 1 週間であるのに対して，自動設計では 3.5 人で 6 時間となっており，人数と時間の積ではカスタム設計は自動設計の約 15 倍の大きさとなっている．文献 [90] では，自動設計の実現性能はカスタム設計より約 30%低いことが示されている．また，文献 [65] では，カスタム設計した回路に対して新たに自動設計を適用することにより面積は 30%小さくなったが，性能は 10%ほど下がったことが示されている．

なお，カスタム設計手法をより効率化する手法が，複数の文献で説明されている．文献 [91] では，プロセッサ設計の経験から，ある設計作業の段階では次の段階の設計作業で問題となる点を考慮できるようなレイアウトプランニングを導入し，カスタム設計における設計のやり直しを減らす手法について述べている．文献 [92] では，一般的にカスタム設計が適用されているデータパス回路の設計において，論理合成や物理合成などをできるだけ適用して自動化するための手法が述べられている．文献 [93] では，カスタム設計を行うブロックにおいて，更に分割して自動設計できる部分と本当にカスタム設計が必要な部分に分けて，自動設計とカスタム設計の両方を取り入れたセミカスタム設計の手法について述べている．

7. タイミング最適化手法

プロセッサ設計では，カスタム設計に伴う並列設計のため，チップは必ず複数のブロックに分割されて設計される．このため，プロセッサ設計でのタイミング最適化は，ブロック及びチップトップの両方の階層で行われる．チップトップは，チップ上でブロック及びその間の接続から構成される．このような分割設計において，タイミングを最適化する三つの手法について説明する．

7.1 セルサイジング

トランジスタレベルで設計を行うフルカスタム設計では，表 5 に示した手法を組み合わせて高い動作速度

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

を実現することができる．しかし，既に準備されたセルのライブラリを使って行うセミカスタム設計や自動設計では，最適なセルのサイズの選択が重要となる．例えば，インバータや NAND などのセルにおいて，内部のトランジスタの幅が基準サイズの N 倍のものが複数準備されている際，どのサイズのセルを使うことが最適であるかを決定する必要がある．

これを解決する代表的な手法として，ロジカルエフォート（Logical Effort）という手法 [58] がある．この手法では，回路の中のセル i のディレイ d_i は，式 (5) のようにモデル化される．

$$d_i = \tau \cdot (g_i h_i + p_i) \tag{5}$$

ここで，τ は各半導体プロセスにおける基準サイズのインバータの出力抵抗と入力容量の積から決定される定数である．この τ は，d_i を半導体プロセスに依存する部分と依存しない部分に分離するために導入される．

p_i は寄生ディレイ（parasitic delay）と呼ばれ，基準サイズのインバータにおいて出力ピンに負荷の接続がない場合のディレイを p_{inv} と表記し，他の種類のセルにおいて出力ピンに負荷の接続がない場合のディレイは p_{inv} の何倍になっているかというかたちで表記する．

g_i はロジカルエフォートと呼ばれ，各セルの入力ピン容量を基準サイズのインバータの入力ピン容量と同じにした場合，各セルの出力ピンが一定負荷を充放電するための時間がそのインバータの充放電時間の何倍になるかを表す値である．この値は，セルの論理タイプとそのセル自身の入力ピン数により決まる値であり，負荷容量の増分に対する充放電時間の増分の割合を表す．文献 [58] ではインバータの場合を 1 と定義するため，2 入力 NAND の場合は 4/3 となる．その理由は，インバータは立上りと立下りのディレイを同じくするために，pMOS のトランジスタサイズが nMOS の 2 倍となっている．これに対して 2 入力 NAND の場合，pMOS 及び nMOS の両方のトランジスタサイズが，インバータの nMOS の 2 倍となっている．この結果，2 入力 NAND の入力ピン容量をインバータの入力ピン容量と同じにした場合，2 入力 NAND の出力電流はインバータの出力電流の 3/4 倍となり，充放電時間は 4/3 倍になるためである．

h_i はエレクトリカルエフォートと呼ばれ，セルに接続されている負荷の容量がそのセル自身の入力ピン容量の何倍になっているかを表す値である．この値は，ファンアウト数とも呼ばれる．

このロジカルエフォートのディレイモデルを使って段数 N のパスのディレイ PD を計算すると式 (6) のようになる．

$$PD = \tau \sum_{i=1}^{N} (g_i h_i + p_i) \tag{6}$$

ロジカルエフォートのディレイモデルでは，p_i はセルの論理タイプにのみよって決まり，セルのサイズには依存しない．したがって，PD を最小にするためには，$g_i h_i$ の総和の部分を最小にすればよい．相加平均 $\geq$ 相乗平均であるから，すべての i に対して $g_i h_i = g_{i+1} h_{i+1} \equiv \hat{f}$ の場合に相加平均 = 相乗平均となり，$g_i h_i$ の総和が最小となる．この結果，PD の最小値 PD_{min} は式 (7) で表される．

$$PD_{min} = \tau N \left(\prod_{i=1}^{N} (g_i h_i) \right)^{1/N} = \tau N \hat{f} \tag{7}$$

このロジカルエフォートの手法では，与えられたパス上の各セルの最適なサイズを求めるためには，まずそのパスついてパス最終段の負荷 C 及びパスの最初のセルの入力ピン容量 $C_{in,3}$ から $\hat{f}$ を求め，パスの最終段の負荷から前方にネットリストをトレースしながら，パス上の各セルにおいて $g_i h_i = \hat{f}$ となるように順次最適なセルのサイズを決定していく．図 16 に示すような最終段のセルが負荷 C を駆動する 3 段のパスの場合，パス上の各セルの最適なサイズを決定する場合には，負荷 C の値からセル 1 のサイズを決め，決定されたセル 1 の入力ピン容量からセル 2 のサイズを決定する．次に，各セルのサイズを決定するための具体的な計算方法を説明する．最初にこのパスにおける $\hat{f}$ を求めると，式 (8) のようになる．

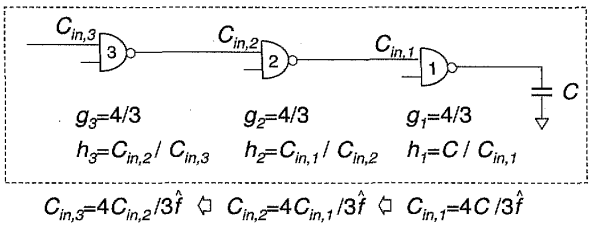


図 16　ロジカルエフォート
Fig. 16　Logical effort.

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

$$\hat{f} = \left(\prod_{i=1}^{3}(g_i h_i)\right)^{1/3} = \left(\left(\frac{4}{3}\right)^3 \cdot \frac{C}{C_{in,3}}\right)^{1/3} \quad (8)$$

負荷 C を駆動するセル 1 のサイズを決めるには，$C_{in,1}$ を決めればよい．$g_1 h_1 = (4/3)\cdot(C/C_{in,1}) = \hat{f}$ であるので，$C_{in,1} = C\cdot(4/3)/\hat{f}$ となる．したがって，セル 1 ではこの値に近い入力ピン容量のセルのサイズを選択する．このセルの選択により $C_{in,1}$ が決定されると，更に前段のセル 2 のサイズを同様に決める．$C_{in,3}$ がパス最終段の負荷 C と同じケースでは，式 (8) により $\hat{f} = 4/3$ となるため，$C_{in,1} = C\cdot(4/3)/(4/3) = C$ となり，同様に $C_{in,2} = C\cdot(4/3)/(4/3) = C$ となる．つまり，2 入力 NAND であるセル 3 の入力ピン容量と同じ負荷を 3 段の 2 入力 NAND で駆動する場合，これら 3 段の 2 入力 NAND セルのサイズは等しいときにパスのディレイが最小となることを示している．

このようにセルの最適サイズを決定したとき，そのサイズに近いセルが準備されていることが重要である．このようなロジカルエフォートを考慮してライブラリ設計している事例が文献 [18], [31] に示されている．なお，文献 [58] におけるロジカルエフォートでは，配線容量や入力波形のスルーは考慮されない．これは，ロジカルエフォートは論理設計段階におけるセルのサイズを最適化することを目的として出発した理論であるためである．ロジカルエフォートはセルのサイジングに関して非常に有用な手法であるため，近年ではレイアウトの段階でもその理論を適用できるように，最近では配線容量や入力波形のスルーも考慮した手法が研究されている [94], [95]．また，このロジカルエフォートによるセルサイジングを低電力化に適用することも研究されている [96]．

7.2 リピータ挿入

チップ上で信号を長距離伝送する必要がある場合，途中にリピータを挿入して配線を短くし，ディレイを改善する手法が採られる．リピータ挿入の基本的なアルゴリズムとして，van Ginneken によるものがある [52]．プロセッサのようにチップが複数のブロックに分割されて設計される際，リピータをどのように挿入するかということも重要な技術となる．

リピータを入れた場合に入れない場合よりディレイが改善する最小距離をクリティカルリピータ距離 L_{CRL} と呼ぶ．この距離 L_{CRL} は，式 (9) で表される [56]．

$$L_{CRL} = K R_g C_g / \sqrt{R_w C_w} \quad (9)$$

ここで，R_g，C_g はそれぞれリピータのセル内部の抵抗及び容量で，R_w，C_w はそれぞれセルの出力ピンに接続される配線の単位長当りの抵抗及び容量である．また，K は入力ピンで許容される波形のスルーやリピータのセルを構成するトランジスタのしきい値電圧などの設計条件で決まる定数である．

半導体プロセスの微細化とともに，トランジスタのディレイに比べて配線のディレイが大きくなるため，この距離 L_{CRL} は微細化とともに短くなっている．したがって，二つのブロック間の信号がリピータなしで二つのブロック間にある別のブロックの上を横切ることを許す手法をとる場合，半導体プロセスの進展に伴って小さくなる L_{CRL} に対応して，リピータが横切るブロックのサイズも小さくする必要がある．表 6 [51] は，90 nm の Metal-6 での L_{CRL} を 1 として，半導体プロセスごとに相対的な値で示したものである．Metal-3 は Metal-6 より下位の配線層であり，単位長さ当りの抵抗が Metal-6 より大きいために，同じ半導体プロセスの世代では Metal-3 の L_{CRL} の値が Metal-6 の値より小さい．また，L_{CRL} は 32 nm の世代では 90 nm の世代に比べて約 1/5 となっている．

二つのブロックに間にリピータを入れる手法は，図 17 に示すように (a) から (c) の三つある．

(a) ブロック間にリピータステーションと呼ばれる配置領域を確保して，そこにリピータを配置する [27], [64]．リピータはブロック A と B を接続するネットリストに追加される．リピータ挿入のために，ブロック A か

表 6 半導体プロセス世代での L_{CRL} の比較 [51]
Table 6 Comparison of L_{CRL}s in process nodes [51].

配線層	90 nm	65 nm	45 nm	32 nm
Metal-3	0.43	0.24	0.14	0.08
Metal-6	1	0.56	0.32	0.19

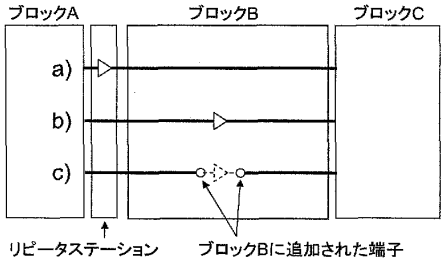


図 17 リピータの配置手法
Fig. 17 Types of repeater placements.

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

らCのそれぞれのネットリストについては変更する必要はなく，ブロックAからCを含む上位階層のブロックのネットリストを変更するだけでよい．しかし，ブロック間にリピータ挿入のための場所を確保する必要があるし，リピータの配置位置はブロックの間の場所に限られる．

(b) ブロックの上にリピータを配置できるように，リピータ配置可能な領域をブロック上に作成して配置する [10], [13]．リピータは (a) の手法と同様にブロックAからCを含む上位階層のブロックのネットリストに記述される．また，挿入されたリピータは，ブロックBの上などの最適な場所に重ねて配置することができる．しかし，重なるブロックではリピータの配置のための場所を確保する必要がある．

(c) ブロック上に入出力ピンを新たに追加して，そのピンを経由してブロックの中に配置されたリピータを経由するように接続する [26]．ブロックB自身のネットリストにおいて，ブロックBの入力ピンからリピータの入力ピンに接続し，リピータの出力ピンがブロックBの出力ピンに接続されるような記述を行い，リピータの配置はブロックBの設計の中で行う．リピータ挿入のために，リピータを入れる信号が存在するブロック及び下位階層のブロックの両方のネットリストへの変更が必要となる．リピータの配置は，該当する下位階層のブロックで行う．

図17において，ブロックBの幅が L_{CRL} より小さい場合には，L_{CRL} の定義よりリピータを挿入する必要がない．したがって，各ブロックの幅と高さが L_{CRL} の値を超えないことを制約とすることにより，一つのブロックをリピータなしで横切ることができるため，このような手法をとっているケースもある [27]．

7.3 上位配線層割付

リピータ挿入によるタイミング最適化のほかに，配線の抵抗が小さい上位の配線層で配線する手法でタイミング最適化を行う場合もある．POWER4 [10] の場合は，チップトップの配線の約30%に相当する35,000本の信号が上位配線層において自動ではないカスタム設計として配線されている．そのほかに，プロセッサ設計事例 [5], [13], [16] などにおいて，上位配線層利用によるタイミング最適化が適用されている．また，まずリピータ挿入を行い，それでも解決できない場合に上位配線層割付を適用するような組合せのケースもある [5]．

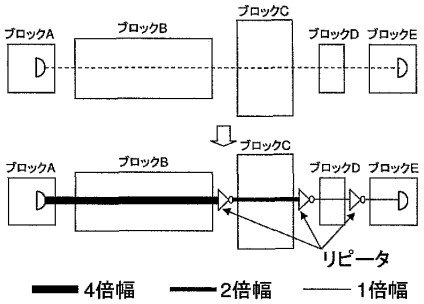


図18 上位配線層とリピータの組合せ [56]
Fig. 18 Combination of upper layer and repeater [56].

一般的な従来の自動設計ツールでは，このような上位配線層の有効な利用が難しいため，実現される性能がカスタム設計に比較して小さい．そのため，自動設計ツールに与える各配線層の抵抗と容量の値をすべて上位配線層の値にして自動設計した結果と，各配線層の抵抗と容量の値をすべて下位配線層の値にして自動設計した結果を比較して，ディレイの差が大きい部分に実際の上位配線層を適用する手法が用いられているケースもある [65]．また，自動設計ツールでも配線層の使い分け機能を追加して，タイミング最適化を図るケースも出ている [56], [66]．

ところで，式 (9) において，上位配線層の方が $R_w C_w$ の値が小さくなるため，距離 L_{CRL} は，上位配線層の方が長くなる．表6の90 nmではMetal-3より上位配線層であるMetal-6では，L_{CRL} が約2.3倍になっている．したがって，リピータと上位配線層を組み合わせることにより，図18のブロックBを横切る配線のように，リピータなしで接続できる距離を長くすることが可能となる [27], [56]．

8. 今後の設計手法とそれを支えるCADシステム

半導体プロセスは，今後更に表7 [97] のように推移していくと考えられている．このように微細化が進む状況において，これまでと同じ設計手法だけではプロセッサ設計は難しくなってくると予想される．この章では，今後の設計手法の課題とそれに対する取組みについてまとめる．

8.1 統計的設計

これまでの設計を決定的設計 (deterministic design) と呼ぶのに対して，製造ばらつきを考慮した設計を統計的設計 (statistical design) と呼ぶ．今後のプロセッサの高性能化を実現する上で，製造ばらつき

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

表 7 半導体プロセスの推移 [97]
Table 7 Trend of semiconductor process [97].

High Volume Manufacturing	2006	2008	2012	2014	2016	2018
Technology Nodes (nm)	65	45	32	16	11	8
Capacity (billion transistors)	4	8	16	64	128	256

を考慮した設計は不可欠となってくる [98], [99].

8.1.1 統計的タイミング解析

統計的設計として最近研究が最も進んでいるものとして，統計的タイミング解析がある．半導体プロセスの微細化の進展により，製造ばらつきの影響が大きくなってきたためである．例えば，5 nm の製造ばらつきは，250 nm 世代では 2%であるが，45 nm 世代では 11%にもなる．したがって，微細化に伴い，製造ばらつきがディレイに与える影響は大きくなっている．

最悪条件に基づくタイミング解析は，パス上の個々のセルや配線が製造ばらつきによってすべて同時に遅くなる最悪の場合を想定している．例えば，セルが直列に N 段だけ接続されたパスにおいて，i 段目のセルの製造ばらつきが分布上において平均的な場合及び $+3\sigma$ という最悪の場合のディレイ値をそれぞれ T_i 及び W_i としたとき，配線のディレイを考慮しない最悪条件に基づくタイミング解析では，パスのディレイ PD は式 (10) のようになる．しかし，この値は，$+3\sigma$ のポイントにばらつく確率 0.135%が N 回続けて発生することを想定しており，過度に悲観的な解析となっている．

$$PD_{max} = \sum_{i=1}^{N} W_i \tag{10}$$

このような悲観性の問題を解決するために，ASIC の設計では，文献 [100] の事例のように，式 (11) に示す二乗和平方根（Root-Sum-Square：RSS）を導入することにより，従来手法のタイミング解析手法の中で悲観性を緩和している．

$$PD_{RSS} = \sum_{i=1}^{N} T_i + \sqrt{\sum_{i=1}^{N} (W_i - T_i)^2} \tag{11}$$

この例では，この解析をベースにしたタイミングエラーをすべて解決した後，ディレイを確率分布として扱う新たな統計的タイミング解析を適用した．この結果，新たに発生したタイミングエラーは，全パス数の約 0.02%のみであることが報告されている．

65 nm 以降では，ディレイを確率分布として扱う統計的タイミング解析が導入され始めている．プロセッサ設計の場合，統計的タイミング解析が設計手法の中でどのように適用されるかについて，四つの形態が考えられる．統計的タイミング解析における計算方法などの詳細については，サーベイ文献 [101] を参照していただきたい．

(a) チップのタイミング歩留り評価

65 nm 以降の半導体プロセスでは，最悪条件に基づく悲観的なタイミング解析のもとではタイミング目標を実現することは困難となっている．ASIC は最悪条件で設計されるが，プロセッサは最悪条件ではなく平均的条件で設計されることが多い [102]．この場合，プロセッサが実際に製造された後，異なる周波数で実際に動作させながら，個々のチップを動作可能な最大動作周波数ごとに分類する．この手法は周波数選別 (speed binning) と呼ばれる [103]．この結果，最大動作周波数が大きいプロセッサの価格は高く，最大動作周波数が小さいプロセッサの価格は安くするというように，最大動作周波数の大きさに応じた価格付けを行うことができる．

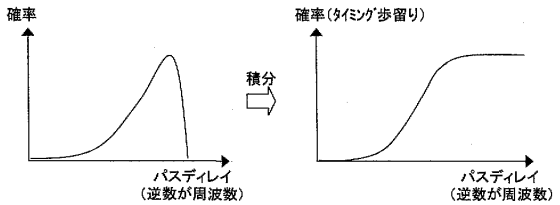
平均的条件で設計されたプロセッサでは，チップを実際に製造した後，目標動作周波数以上で動作するチップの個数が全体のどのくらいの割合（以降，この割合をタイミング歩留りと呼ぶ）になるかを評価する必要があり，ここに統計的タイミング解析が適用される．各セルのディレイは確率分布としてモデル化され，パスのディレイは各セルの確率分布の統計的加算（*30）により確率分布として計算される（図 19 の (a)）．パスの途中に合流がある場合には統計的 MAX (*31) [104] により確率分布が計算される．こうして求められたすべてのパスのディレイ確率分布について統計的 MAX を計算し，計算結果の分布を積分することにより，図 19 の (b) のような累積分布が得られる．この累積分布は設計データから計算可能であるため，設計段階においてタイミング歩留りの評価が可能となる [105]．もし，タイミング歩留りが期待の値より小さい場合には，平均的条件で実行する従来のタイミング解析でのディレイ目標を上げるか，次の (b) や (c) の手法を用いて改善するパスを決める．

(b) パスの製造ばらつきに対する感度評価

パスの製造ばらつきに対する感度の厳密な定義は文献 [106] に詳述されるが，この感度の直感的な意味は，

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers



(a) 周波数の確率分布　(b) 周波数の累積分布

図 19　タイミング歩留り分布
Fig. 19　Timing yield distribution.

ある一つのパスのディレイが製造ばらつきにより変動したとき，実際に製造したチップの最大動作周波数が変動する確率と考えることができる．この感度は，そのパスのディレイを改善した場合，タイミング歩留りにどの程度寄与するかという度合と考えることもできる．

例えば，あるパスの段数が小さく，そのパスのディレイがクリティカルパスのディレイに比べてとても小さい場合，このパスのディレイが製造ばらつきにより変動しても，チップの最大動作周波数を変動させる可能性は小さい．この場合，パスの製造ばらつきに対する感度は 0 に近い値となる．このようなパスは，チップのタイミング歩留りに影響しないため，改善するパスの対象にしなくてよい．このことから，感度が高いパスからディレイ改善することが効果的である．

(c) パスのクリティカル度評価

パスのクリティカル度についても，(b) と同様に厳密な定義は文献 [106] に詳述されるが，このクリティカル度の直感的な意味は，次のように考えることができる．各パスのディレイを計算する際，パス上の各セルのディレイをそのセルのディレイ分布からランダムに一つの値をサンプリングして計算する．このサンプリングを複数回繰り返したとき，各パスについてそのパスがクリティカルパスになる回数を集計する．各パスについて，その回数をサンプリングした回数で割って得られた確率と考えることができる．

つまり，チップが多数製造されたとき，クリティカル度が高いパスが実際にクリティカルパスになる頻度も高いと考えられる．このため，設計段階でタイミング歩留りを高めるためには，クリティカル度が高いパスからディレイ改善を行うことが効果的である．

(d) ブロックごとのタイミング歩留り評価

統計的タイミング解析によりチップ全体のタイミング歩留り分布を作成するが，統計的タイミング解析を実際に運用していく中で，チップ全体のタイミング歩留りに一番影響をもっている回路部分を特定して，その部分を改善していくことが重要となる．

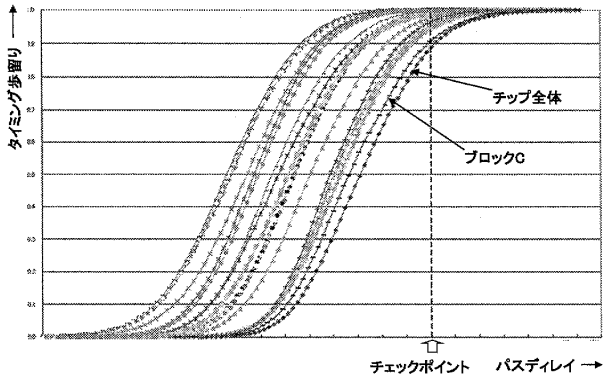


図 20　各ブロックに対するタイミング歩留り [107]
Fig. 20　Timing yield of each block [107].

チップ全体のタイミング歩留りに一番影響をもっている部分を特定するために，まずチップを構成するブロック単位のタイミング歩留り分布を全ブロックについて計算して，それらを比較することにより，タイミング歩留りに一番影響をもっているブロックを特定する．具体的な適用例としては，チップ全体のタイミング歩留りに影響をもつ約 10 万パスについて，個々のブロックのパスに分類して，同一のブロックのパス全体に対して統計的 MAX 演算を適用してブロック単位のタイミング歩留り分布を計算する．なお，ブロック間パスの全体についてもタイミング歩留り分布を計算する．

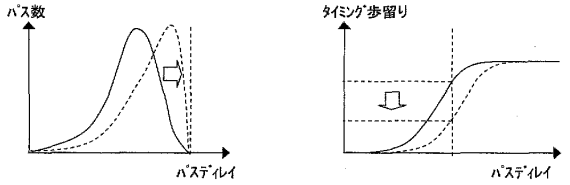
実際のプロセッサ設計に適用した結果が図 20 に示される [107]．あるディレイのチェックポイントにおいて（このチェックポイントは説明の都合上，仮に設定したディレイ値のポイントである），タイミング歩留りが一番悪いブロックは C と特定できる．したがって，このブロックのタイミング最適化を更に進めることにより，チップ全体のタイミング歩留りを向上させることができる．

8.1.2　漏れ電流とタイミング歩留りの統計的考慮

半導体プロセスの微細化により，トランジスタの漏れ電流が大きくなり，更にこの漏れ電流は製造ばらつきにより大きく変動する．文献 [108] では，実際のプロセッサを製造した結果，最大動作周波数が約 30%の範囲でばらつくが，その中にあるチップの漏れ電流は約 20 倍の範囲でばらついていることを示している．また，漏れ電流が大きいトランジスタの動作は一般的に速い．このため，チップの製造後の漏れ電流測定が

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers



(a) パスディレイ分布の変化　(b) タイミング歩留りへの影響

図 21　電力とタイミング歩留りの trade-off
Fig. 21　Trade-off between power and timing yield.

必要であり，漏れ電流が規格値以上のチップは，高い周波数で動作する可能性があっても消費電力や信頼性の面から商品化できない．したがって，製造ばらつきによって変動する漏れ電流を設計段階で統計的に解析し，漏れ電流に対する動作周波数の分布を意識した設計手法の研究も進められている [109].

また，タイミング設計段階において，すべてのパスのディレイが目標動作周波数に対応する目標ディレイ以下となるようにした後，各パスのディレイが目標ディレイを超えない範囲で，漏れ電流の少ない High-V_{th} などのセルへと置き換えながら，ディレイに余裕のあるパスを遅くして低電力化する手法がある．その結果，図 21 の (a) に示すように，すべてのパスは目標ディレイは超えない範囲で遅い方に分布全体が偏る．この結果，実際に製造した場合，製造ばらつきによってタイミング歩留りは図 21 の (b) のように低下する可能がある．したがって，パスのディレイの余裕量だけでなく，**8.1.1** で説明したパスの製造ばらつきに対する感度を使い，ある一定以下の感度のパスについてのみ漏れ電流の少ないセルに置き換えるなどの手法の研究も進められている [110], [111].

8.2　CAD システム構築

8.2.1　事例の分析

旧 DEC のプロセッサ設計 [3], [83] では，高性能化を実現するために全面的にトランジスタレベルでの設計を採用した．CAD システムは，設計要求，CAD 研究，トレンド，テクノロジー要求に基づいて開発されるべきものであるとして，CAD システム全体の自社開発に力を入れてきた．特に CAD システムに求めたものは，大量の設計情報から問題箇所を見つけることであるとして，検証 CAD に特に力を入れてきた．

IBM のプロセッサ設計では，1980 年代のメインフレーム設計の時代から主に自社で CAD システムを開発するという戦略をとっている．文献 [112] には，自社で開発した CAD ツールが年代ごとに記されており，一番初期のものは 1960 年代に高速計算機用回路のトランジスタレベルでの解析ツールまでさかのぼる．EDA ベンダ（*32）のツールが出てくる前からプロセッサ設計を行ってきたため，CAD ツールの範囲も，論理設計，物理設計，タイミング解析，製造テストなど全領域を自社開発の CAD システムで対応してきた．これらツールはプロセッサ設計のみならず，高性能な ASIC 設計にも共通に使われている．ただし，最近の POWER6 では，EDA ベンダのセル自動配置や自動配線のツールも適用され始めている [13].

Intel のプロセッサ設計では，1980 年代にメモリからプロセッサにビジネスの主軸を移す際，CAD システムの重要性を考えて多くの投資を行った．そのため，1981 年以前は，CAD システム構成の 70%が社外のもの，20%が社外のものに手を加えたもの，10%が自社開発であったが，1985 年までにその割合はそれぞれ 15%，30%，55%と，自社開発のツールが 55%まで増えている [113]．近年の Intel でのツールの自社開発の基準は，既に運用されている CAD システムに対してスムーズに導入できない場合は自社開発するというものである [114].

Sun のプロセッサ設計では，多くの EDA ベンダの中で一番よいツールを選び出して，それらでプロセッサ設計のための CAD システムを構築するという考え方に基づいてる [115]．このようなオープンな CAD システムを多く使い，EDA ベンダツールにないものは自社で開発するという手法でプロセッサ設計を行っている．それでも自社開発が必要な CAD ツールとして，高性能化のための電気的ルールチェックやクロックスキュー解析などを自社開発している．また，EDA ベンダに過度に依存してしまうと，新たなツールへの移行が難しくなるという危険性も指摘している．

富士通のプロセッサ設計では，半導体プロセスに依存せずに，しかも既に成熟している部分など，自社開発のメリットがないと判断したところに EDA ベンダツールを活用し，プロセッサの高性能化の鍵を握る設計手法の核となるカスタム設計をベースにしたレイアウト設計やタイミング改善に関連する部分は自社開発するという戦略をとっている [20]．このため，セルベースの物理設計に関連するところでは，サインオフ（*33）のための DRC・LVS ツール以外はほとんど自社で開発している．また，トランジスタレベルでの設計環境では，ネットリストやレイアウトの入力には EDA ベンダツールが使われ，性能や回路品質にかか

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

わる解析・検証系のツールには自社開発ツールも適用されている．

8.2.2 今後の方向

これまでCADシステムを自社開発しているケースでも，その開発負担は増えている．例えば，Intelでは2003年までの過去7年間に集められたデータから，トランジスタ数は10Mから100M，RTL記述の行数は400Kから1,000K，製造前バグ数は500から8,000，検証チームは2倍にそれぞれ拡大していることを示している．これは，設計の複雑さの増大にCADツールと設計手法の進歩が追いついていないことを示している[116]．

また，今後のプロセッサ設計では，カスタム設計の比率が小さくなり，自動設計がより拡大するとの指摘もある[97]．理由としては，設計条件が複雑化し，最適化項目がこれまでの単なる動作周波数や面積だけでなく，更に電力・歩留り・信頼性までを考慮して実現する必要があるため，人手ベースとするカスタム設計ではこれら複数の制約を同時に考えながら設計することに限界があると考えられているからである．

このような状況から，自社開発ツールとEDAベンダツールの融合の必要性が高まってきた[36],[117]．こうした融合を促進するために，EDAベンダがOpenAccessというかたちでデータベースを公開し，データベースをアクセスするためのAPI（*34）も整えてきた．ASP-DAC（Asia and South Pacific Design Automation Conference）2006ではOpenAccessのセッションで紹介されている[118]～[121]．

9. むすび

本論文では，1990年代以降に発表されたプロセッサについて，その設計手法をサーベイした．まず，サーベイする対象となるプロセッサ事例を紹介した後，回路設計や物理設計の段階の設計手法について，実例を引用しながら説明した．

半導体プロセスの微細化により，搭載可能なトランジスタ数が増加するとともに，製造ばらつきも更に大きくなる．このような状況において，今後のプロセッサ設計における新たな設計手法についての展望についても述べた．今後も高性能なプロセッサ設計を続けていくためには，世界のプロセッサ設計で適用されている設計手法に関する知見をサーベイし，理解することは重要である．

なお，1980年代初頭の段階での日本におけるCADシステムのサーベイは文献[122]を，また2000年代初頭までのプロセッサ・ASIC設計企業のCADシステムのサーベイは文献[112]を参照して頂きたい．2000年代初頭時点でのEDAベンダの立場からの設計手法やCADシステムのサーベイについては文献[123]を参照して頂きたい．

謝辞 貴重な御指摘，御意見を賜りました査読員の先生の皆様に心から感謝申し上げます．

文　献

[1] A. McKnight, “Flexible design methodology management,” IEE Colloquium on Design Management Environments in CAD, pp.1/1–1/6, 1991.

[2] W.J. Grundmann, D. Dobberpuhl, R.L. Allmon, and N.L. Rethman, “Designing high performance CMOS microprocessors using full custom techniques,” Design Automation Conference, pp.722–727, 1997.

[3] P.E. Gronowski, W.J. Bowhill, R.P. Preston, M.K. Gowan, and R.L. Allmon, “High-performance microprocessor design,” IEEE J. Solid-State Circuits, vol.33, no.5, pp.676–686, 1998.

[4] D. Pham, H.W. Anderson, E. Behnen, M. Bolliger, S. Gupta, P. Hofstee, P. Harvey, C. Johns, J. Kahle, A. Kameyama, J. Keaty, B. Le, S.g Lee, T. Nguyen, J. Petrovick, M. Pham, J. Pille, S. Posluszny, M. Riley, J. Verock, J. Warnock, S. Weitzel, and D. Wendel, “Key features of the design methodology enabling a multi-core SoC implementation of a first-generation CELL processor,” Asia and South Pacific Design Automation Conference, pp.871–878, 2006.

[5] J.A. Kahle, M.N. Day, H.P. Hofstee, C.R. Johns, T.R. Maeurer, and D. Shippy, “Introduction to the Cell multiprocessor,” IBM J. Res. & Dev., vol.49, no.4/5, pp.589–604, 2005.

[6] O. Takahashi, E. Behnen, S.R. Cottier, P. Coulman, S.H. Dhong, B. Flachs, P. Hofstee, C.J. Johnson, and S. Posluszny, “Cell broadband engine processor design methodology,” Custom Intergrated Circuits Conference, pp.711–716, 2007.

[7] S. Rusu, J. Stinson, S. Tam, J. Leung, H. Muljono, and B. Cherkauer, “A 1.5-GHz 130-nm Itanium^{TM} 2 processor with 6-MB on-die L3 cache,” IEEE J. Solid-State Circuits, vol.38, no.11, pp.1887–1895, 2003.

[8] J. Stinson and S. Rusu, “A 1.5 GHz third generation Itanium^{TM} 2 processor,” Design Automation Conference, pp.706–709, 2003.

[9] B. Stackhouse, S. Bhimji, C. Bostak, D. Bradley, B. Cherkauer, J. Desai, E. Francom, M. Gowan, P. Gronowski, D. Krueger, C. Morganti, and S. Troyer, “A 65 nm 2-billion transistor quad-core Itanium processor,” IEEE J. Solid-State Circuits, vol.44, no.1, pp.18–31, 2009.

[10] J.D. Warnock, J.M. Keaty, J. Petrovick, J.G. Clabes, C.J. Kircher, B.L. Krauter, P.J. Restle, B.A. Zoric,

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

and C.J. Anderson, "The circuit and physical design of the POWER4 microprocessor," IBM J. Res. & Dev., vol.46, no.1, pp.27–51, 2002.

[11] Y.H. Chan, P. Kudva, L. Lacey, G. Northrop, and T. Rosser, "Physical synthesis methodology for high performance microprocessors," Design Automation Conference, pp.696–701, 2003.

[12] B. Stolt, Y. Mittlefehldt, S. Dubey, G. Mittal, M. Lee, J. Friedrich, and E. Fluhr, "Design and implementation of the POWER6 microprocessor," IEEE J. Solid-State Circuits, vol.43, no.1, pp.21–28, 2007.

[13] R. Berridge, R.M. Averill III, A.E. Barish, M.A. Bowen, P.J. Camporese, J. DiLullo, P.E. Dudley, J. Keinert, D.W. Lewis, R.D. Morel, T. Rosser, N.S. Schwartz, P. Shephard, H.H. Smith, D. Thomas, P.J. Restle, J.R. Ripley, S.L. Runyon, and P.M. Williams, "IBM POWER6 microprocessor physical design and design methodology," IBM J. Res. & Dev., vol.51, no.6, pp.685–714, 2007.

[14] T. Hattori, Y. Nitta, M. Seki, S. Narita, K. Uchiyama, T. Takahashi, and R. Satomura, "Design methodology of a 200 MHz superscalar microprocessor: SH-4," Design Automation Conference, pp.246–249, 1998.

[15] R. Heald, K. Aingaran, C. Amir, M. Ang, M. Boland, A. Das, P. Dixit, G. Gouldsberry, J. Hart, T. Horel, W.J. Hsu, J. Kaku, C. Kim, S. Kim, F. Klass, H. Kwan, R. Lo, H. McIntyre, A. Mehta, D. Murata, S. Nguyen, Y.P. Pai, S. Patel, K. Shin, K. Tam, S. Vishwanthaiah, J. Wu, G. Yee, and H. You, "Implementation of a 3rd-generation SPARC V9 64 b microprocessor," IEEE International Solid-State Circuits Conference, pp.412–413, 2000.

[16] R. Heald, K. Aingaran, C. Amir, M. Ang, M. Boland, P. Dixit, G. Gouldsberry, D. Greenley, J. Grinberg, J. Hart, T. Horel, W.J. Hsu, J. Kaku, C. Kim, S. Kim, F. Klass, H. Kwan, G. Lauterbach, R. Lo, H. McIntyre, A. Mehta, D. Murata, S. Nguyen, Y.P. Pai, S. Patel, K. Shin, K. Tam, S. Vishwanthaiah, J. Wu, G. Yee, and E. You, "A third-generation SPARC V9 64-b microprocessor," IEEE J. Solid-State Circuits, vol.35, no.11, pp.1526–1538, 2000.

[17] G.K. Konstadinidis, M. Tremblay, S. Chaudhry, M. Rashid, P.F. Lai, Y. Otaguro, Y. Orginos, S. Parampalli, M. Steigerwald, S. Gundala, R. Pyapali, L.D. Rarick, I. Elkin, Y. Ge, and I. Parulkar, "Architecture and physical implementation of a third generation 65 nm, 16 Core, 32 thread chip-multithreading SPARC processor," IEEE J. Solid-State Circuits, vol.44, no.1, pp.7–17, 2009.

[18] H. Ando, Y. Yoshida, A. Inoue, I. Sugiyama, T. Asakawa, K. Morita, T. Muta, T. Motokurumada, S. Okada, H. Yamashita, Y. Satsukawa, A. Konmoto, R. Yamashita, and H. Sugiyama, "A 1.3 GHz fifth generation SPARC64 microprocessor," Design Automation Conference, pp.702–705, 2003.

[19] N. Ito, H. Komatsu, Y. Tanamura, R. Yamashita, H. Sugiyama, Y. Sugiyama, and H. Hamamura, "A physical design methodology for 1.3 GHz SPARC 64 microprocessor," International Conference on Computer Design, pp.204–210, 2003.

[20] N. Ito, H. Komatsu, A. Kanuma, A. Yoshitake, Y. Tanamura, H. Sugiyama, R. Yamashita, K. Nabeya, H. Yoshino, H. Yamanaka, M. Yanagida, Y. Ozeki, K. Ishizaka, T. Kono, and Y. Isoda, "Design methodology for 2.4 GHz dual-core microprocessor," Asia and South Pacific Design Automation Conference, pp.896–901, 2007.

[21] 今野良展, 井川康宏, 沢野知紀, 金丸恵祐, 小野公揮, 熊崎雅仁, "SX-9 の CAD 技術," NEC 技法, vol.61, no.4, pp.30–34, 2008.

[22] R.M. Averill III, K.G. Barkley, M.A. Bowen, P.J. Camporese, A.H. Dansky, R.F. Hatch, D.E. Hoffman, M.D. Mayo, S.A. McCabe, T.G. McNamara, T.J. McPherson, G.A. Northrop, L. Sigal, H.H. Smith, D.A. Webber, and P.M. Williams, "Chip integration methodology for the IBM S/390 G5 and G6 custom microprocessors," IBM J. Res. & Dev., vol.43, no.5/6, pp.681–706, 1999.

[23] G.K. Konstadinidis, K. Normoyle, S. Wong, S. Bhutani, H. Stuimer, T. Johnson, A. Smith, D.Y. Cheung, F. Romano, S. Yu, S.H. Oh, V. Melamed, S. Narayanan, D. Bunsey, C. Khieu, K.J. Wu, R. Schmitt, A. Dumlao, M. Sutera, J. Chau, K.J. Lin, and W.S. Coates, "Implementation of a third-generation 1.1-GHz 64-bit microprocessor," IEEE J. Solid-State Circuits, vol.37, no.11, pp.1461–1469, 2002.

[24] T. Takayanagi, J.L. Shin, B. Petrick, J. Su, and A.S. Leon, "A dual-core 64b UltraSPARC microprocessor for dense server applications," Design Automation Conference, pp.673–677, 2004.

[25] A.S. Leon, K.W. Tam, J.L. Shin, D. Weisner, and F. Schumacher, "A power-efficient high-throughput 32-thread SPARC processor," IEEE J. Solid-State Circuits, vol.42, no.1, pp.7–16, 2007.

[26] U.G. Nawathe, M. Hassan, K.C. Yen, A. Kumar, A. Ramachandran, and D. Greenhill, "Implementation of an 8-core, 64-thread, power-efficient SPARC server on a chip," IEEE J. Solid-State Circuits, vol.43, no.1, pp.6–20, 2008.

[27] R. Varada, M. Sriram, K. Chou, and J. Guzzo, "Design and integration methods for a multi-threaded dual core 65 nm Xeon™ processor," International Conference on Computer Aided Design, pp.607–610, 2006.

[28] R. Varada, S. Tam, J. Benoit, and K. Chou, "SOC design challenges in a multi-threaded 65 nm dual core Xeon™ MP processor," International SOC Conference, pp.217–220, 2006.

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

[29] W. Keshlear, S. Oliver, R. Colyer, J. Schreiber, T. Antoniadis, T. Mickelson, T. Puzey, and M. Bates, "Design optimizations for reduced power and higher operating frequency in a custom x86-64 processor core," Custom Integrated Circuits Conference, pp.17–20, 2009.

[30] G.A. Northrop and P.F. Lu, "A semi-custom design flow in high-performance microprocessor design," Design Automation Conference, pp.426–431, 2001.

[31] B.W. Curran, Y.H. Chan, P.T. Wu, P.J. Camporese, G.A. Northrop, R.F. Hatch, L.B. Lacey, J.P. Eckhardt, D.T. Hui, and H.H. Smith, "IBM eServer z900 high-frequency microprocessor technology, circuits, and design methodology," IBM J. Res. & Dev., vol.46, no.4/5, pp.631–644, 2002.

[32] J. Koehl, U. Baur, T. Ludwig, B. Kick, and T. Pflueger, "A flat, timing-driven design system for a high-performance CMOS processor chipset," Design, Automation and Test in Europe, pp.312–320, 1998.

[33] M.J. Flynn and P. Hung, "Microprocessor design issues: Thoughts on the road ahead," IEEE Micro, vol.25, no.3, pp.16–31, 2005.

[34] D.J. Frank, R. Puri, and D. Toma, "Design and CAD challenges in 45 nm CMOS and beyond," International Conference on Computer Aided Design, pp.329–333, 2006.

[35] ITRS 2009 Edition, System Drivers (オンライン), 入手先 http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_SysDrivers.pdf (参照 2011-3-18).

[36] J. Parkhurst, J. Darringer, and B. Grundmann, "From single core to multi-core: preparing for a new exponential," International Conference on Computer Aided Design, pp.67–72, 2006.

[37] D. Sadowy, "New microprocessor features for lower power and increased performance: Doing "Moore's" with less," International Symposium on Electronics and the Environment, pp.146–150, 2005.

[38] B. Shekhar, N.P. Jouppi, and P. Stenstrom, "Microprocessors in the era of terascale integration," Design, Automation & Test in Europe Conference, pp.1–6, 2007.

[39] S. Borkar, "Thousand core chips - a technology perspective," Design Automation Conference, pp.746–749, 2007.

[40] ITRS 2003 Edition, Design (オンライン), 入手先 http://www.itrs.net/Links/2003ITRS/Design2003.pdf (参照 2010-6-14).

[41] E. Salman, A. Dasdan, F. Taraporevala, K. Kucukcakar, and E.G. Friedman, "Exploiting setup-hold-time interdependence in static timing analysis," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.26, no.6, pp.1114–1125, 2007.

[42] C. Yeh, G. Wilke, H. Chen, S. Reddy, H. Nguyen, T. Miyoshi, W. Walker, and R. Murgai, "Clock distribution architectures: A comparative study," International Symposium on Quality Electronic Design, pp.85–91, 2006.

[43] Y. Wang, W.S. Luk, X. Zeng, J. Tao, C. Yan, J. Tong, W. Cai, and J. Ni, "Timing yield driven clock skew scheduling considering non-Gaussian distributions of critical path delays," Design Automation Conference, pp.223–226, 2008.

[44] S. Tam, R.D. Limaye, and U.N. Desai, "Clock generation and distribution for the 130-nm Itanium^{TM} 2 processor with 6-MB on-die L3 cache," IEEE J. Solid-State Circuits, vol.39, no.4, pp.636–642, 2004.

[45] P.R. O'Brien and T.L. Savarino, "Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation," International Conference on Computer Aided Design, pp.512–515, 1989.

[46] S. Abbaspour and M. Pedram, "Calculating the effective capacitance for the RC interconnect in VDSM technologies," Design Automation Conference, pp.43–48, 2003.

[47] C.V. Kashyap, C.J. Alpert, and A. Devgan, "An "effective" capacitance based delay metric for RC interconnect," International Conference on Computer Aided Design, pp.229–234, 2000.

[48] J. Werber, D. Rautenbach, and C. Szegedy, "Timing optimization by restructuring long combinatorial paths," International Conference on Computer Aided Design, pp.536–543, 2007.

[49] W. Chen, C.T. Hsieh, and M. Pedram, "Simultaneous gate sizing and fanout optimization," International Conference on Computer Aided Design, pp.374–378, 2000.

[50] D. Baneres, J. Cortadella, and M. Kishinevsky, "Layout-aware gate duplication and buffer insertion," Design, Automation & Test in Europe Conference, pp.1–6, 2007.

[51] P. Saxena, N. Menezes, P. Cocchini, and D.A. Kirkpatrick, "Repeater scaling and its impact on CAD," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.23, no.4, pp.451–463, 2004.

[52] L.P.P.P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay," IEEE International Symposium on Circuits and Systems, pp.865–868, 1990.

[53] J.A. He and H. Kobayashi, "Simultaneous wire sizing and wire spacing in post-layout performance optimization," Asia and South Pacific Design Automation Conference, pp.373–378, 1998.

[54] J. Cong, L. He, C.K. Koh, and Z.D. Pan, "Interconnect sizing and spacing with consideration of coupling capacitance," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.20, no.9, pp.1164–1169, 2001.

[55] K. Moiseev, S. Wimer, and A. Kolodny, "Timing optimization of interconnect by simultaneous net-ordering, wire sizing and spacing," IEEE Interna-

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

tional Symposium on Circuits and Systems, pp.329–332, 2006.

[56] Z. Li, C.J. Alpert, S. Hu, T. Muhmud, S.T. Quay, and P.G. Villarrubia, "Fast interconnect synthesis with layer assignment," International Symposium on Physical Design, pp.71–77, 2008.

[57] ITRS 2009 Edition, Interconnect (オンライン), 入手先 http://www.itrs.net/Links/2009ITRS/2009Chapters_2009Tables/2009_Interconnect.pdf (参照 2011-3-18).

[58] I. Sutherland, B. Sproull, and D. Harris, Logical effort: designing fast CMOS circuits, Morgan Kaufmann Publishers, ISBN 1-55860-557-6, 1999.

[59] M. Hirabayashi, K. Nose, and T. Sakurai, "Design methodology and optimization strategy for dual-V_{TH} scheme using commercially available tools," International Symposium on Low Power Electronics and Design, pp.283–286, 2001.

[60] T. Kuroda, "Optimization and control of V_{DD} and V_{TH} for low-power, high-speed CMOS design," International Conference on Computer Aided Design, pp.28–34, 2002.

[61] C.J. Anderson, "Beyond innovation: Dealing with the risks and complexity of processor design in 22 nm," Design Automation Conference, pp.103–103, 2009.

[62] S. Raman and M. Lubyanitsky, "Cone resynthesis ECO methodology for multi-million gate designs," International Conference on VLSI Design, pp.195–199, 2009.

[63] P. Gopalakrishnan, A. Odabasioglu, L. Pileggi, and S. Raje, "Overcoming wireload model uncertainty during physical design," International Symposium on Physical Design, pp.182–189, 2001.

[64] R. McInerney, K. Leeper, T. Hill, H. Chan, B. Basaran, and L. McQuiddy, "Methodology for repeater insertion management in the RTL, layout, floorplan and fullchip timing databases of the Itanium™ microprocessor," International Symposium on Physical Design, pp.99–104, 2000.

[65] K. Ueno, H. Murakami, N. Yano, R. Okuda, T. Himeno, T. Kamei, and Y. Urakawa, "A design methodology realizing an over GHz synthesizable streaming processing unit," VLSI Circuits, pp.48–49, 2007.

[66] P. Saxena, "The evolution of interconnect management in physical synthesis," International Symposium on VLSI Design Automation and Test, pp.27–30, 2009.

[67] 伊藤則之, 小松裕成, 棚村佳保, 山下良一, 杉山広行, 吉竹昭博, 鍋谷研一, 杉山八六, "2.16 GHz SPARC64 マイクロプロセサ設計用タイミング・レイアウト設計手法," DA シンポジウム, pp.255–260, 2005.

[68] D. Behrens, K. Harbich, and E. Barke, "Hierarchical partitioning," International Conference on Computer Aided Design, pp.470–477, 1996.

[69] R.G. Bushroe, S. DasGupta, A. Dengi, P. Fisher, S. Grout, G. Ledenbach, N.S. Nagaraj, and R. Steele, "Chip hierarchical design system (CHDS9): A foundation for timing-driven physical design into the 21st century," International Symposium on Physical Design, pp.212–217, 1997.

[70] S. Grout, G. Ledenbach, R.G. Bushroe, P. Fisher, D. Cottrell, D. Mallis, S. DasGupta, J. Morrell, J. Sayah, R. Gupta, P.T. Patel, and P. Adams, "Hierarchy – A CHDStd tool for the coming deep submicron complex design crisis," Asia and South Pacific Design Automation Conference, pp.257–260, 1998.

[71] J. Cong, "Timing closure based on physical hierarchy," International Symposium on Physical Design, pp.170–174, 2002.

[72] N. Ito, H. Komatsu, Y. Tanamura, R. Yamashita, H. Sugiyama, Y. Sugiyama, and H. Hamamura, "A physical design methodology for 1.3 GHz SPARC 64 microprocessor," International Conference on Computer Design, pp.204–210, 2003.

[73] A. Mehrotra, L. van Ginneken, and Y. Trivedi, "Design flow and methodology for 50 M gate ASIC," Asia and South Pacific Design Automation Conference, pp.640–647, 2003.

[74] F.Y.C. Mang, W. Hou, and P.H. Ho, "Techniques for effective distributed physical synthesis," Design Automation Conference, pp.859–864, 2007.

[75] J.Y. Sayah, R. Gupta, D.D. Sherlekar, P.S. Honsinger, J.M. Apte, S.W. Bollinger, H.H. Chen, S. DasGupta, E.P. Hsieh, A.D. Huber, E.J. Hughes, Z.M. Kurzum, V.B. Rao, T. Tabtieng, V. Valijan, and D.Y. Yang, "Design planning for high-performance ASICs," IBM J. Res. & Dev., vol.40, no.4, pp.431–452, 1996.

[76] B. Schurmann, J. Altmeyer, and M. Schutze, "On modeling top-down VLSI design," International Conference on Computer Aided Design, pp.508–515, 1994.

[77] M.S. McCorquodale, F.H. Gebara, K.L. Kraver, E.D. Marsman, R.M. Senger, and R.B. Brown, "A top-down microsystems design methodology and associated challenges," Design, Automation and Test in Europe, pp.292–296, 2003.

[78] N. Kaul, "Design planning trends and challenges," International Symposium on Physical Design, 2010 (オンライン), 入手先 http://www.sigda.org/ispd/slides10/1_02.pdf (参照 2010-6-14).

[79] R. Kumar, "Interconnect and noise immunity design for the Pentium™ 4 processor," Design Automation Conference, pp.938–943, 2003.

[80] D.H. Allen, S.H. Dhong, H.P. Hofstee, J. Leenstra, K.J. Nowka, D.L. Stasiak, and D.F. Wendel, "Custom circuit design as a driver of microprocessor performance," IBM J. Res. & Dev., vol.44, no.6, pp.799–

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

722, 2000.

[81] S.E. Rich, M.J. Parker, and J. Schwartz, "Reducing the frequency gap between ASIC and custom designs: A custom perspective," Design Automation Conference, pp.432–437, 2001.

[82] M. Nemani and V. Tiwari, "Macro-driven circuit design methodology for high-performance datapaths," Design Automation Conference, pp.661–666, 2000.

[83] M. Reilly, "Designing an Alpha microprocessor," Computer, vol.32, no.7, pp.27–34, 1999.

[84] Q. Liu and M.M. Sadowska, "A study of netlist structure and placement efficiency," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.24, no.5, pp.762–772, 2005.

[85] L. Ding and P. Mazumder, "Optimal transistor tapering for high-speed CMOS circuits," Design, Automation & Test in Europe Conference, pp.708–713, 2002.

[86] A.R. Conn, P.K. Coulman, R.A. Haring, G.L. Morrill, and C. Visweswariah, "Optimization of custom MOS circuits by transistor sizing," International Conference on Computer Aided Design, pp.174–180, 1996.

[87] A.R. Conn, I.M. Elfadel, W.W. Molzen, Jr., P.R. O'Brieny, P.N. Strenski, C. Visweswariah, and C.B. Whan, "Gradient-based optimization of custom circuits using a static-timing formulation," Design Automation Conference, pp.452–459, 1999.

[88] C. Visweswariah and A.R. Conn, "Formulation of static circuit optimization with reduced size, degeneracy and redundancy by timing graph manipulation," International Conference on Computer Aided Design, pp.244–251, 1999.

[89] Y. Liao, G. Mehta, M. Liu, Y. Su, and N. Raman, "Fully automated physical implementation methodology for Tolapai-the first IA based SoC," International Conference on Solid-State and Integrated-Circuit Technology, pp.1815–1818, 2008.

[90] 伊藤則之，小松裕成，吉田裕司，"SPARC64^{TM} プロセッサ設計用 CAD システムの開発，" 信学論 (D)，vol.J92-D, no.9, pp.1495–1507, Sept. 2009.

[91] B. Krishna and G. Kleinfeld, "Circuit design environment and layout planning," Intel Technology J. Q1, pp.1–9, 1999.

[92] T. Chan, A. Chowdhary, B. Krishna, A. Levin, G. Meeker, and N. Sehgal, "Challenges of CAD development for datapath design," Intel Technology Journal Q1, pp.1–17, 1999.

[93] N.N. Eleyan, K. Lin, M. Kamal, B. Mohammad, and P. Bassett, "Semi-custom design flow: Leveraging place and route tools in custom circuit design," International Conference on IC Design and Technology, pp.143–147, 2009.

[94] A. Morgenshtein, E.G. Friedman, R. Ginosar, and A. Kolodny, "Unified logical effort - a method for delay evaluation and minimization in logic paths with RC interconnect," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.18, no.5, pp.689–696, 2010.

[95] C.C. Wang and D. Markovic, "Delay estimation and sizing of CMOS logic using logical effort with slope correction," IEEE Trans. Circuits Syst., vol.56, no.8, pp.634–638, 2009.

[96] N. Miura, N. Kato, and T. Kuroda, "Practical methodology of post-layout gate sizing for 15% more power saving," Asia and South Pacific Design Automation Conference, pp.434–437, 2004.

[97] S. Borkar, "Design perspectives on 22 nm CMOS and beyond," Design Automation Conference, pp.93–94, 2009.

[98] T. Karnik, V. De, and S. Borkar, "Statistical design for variation tolerance: Key to continued Moore's law," International Conference on Integrated Circuit Design and Technology, pp.175–176, 2004.

[99] U. Schlichtmann, M. Schmidt, M. Pronath, V. Glockel, H. Kinzelbach, M. Dietrich, U. Eichler, and J. Haase, "Digital design at a crossroads: How to make statistical design methodologies industrially relevant," Design, Automation & Test in Europe Conference, pp.1542–1547, 2009.

[100] L.B. Marshall and E.A. Foreman, "Timing closure in 65-nanometer ASICs using statistical static timing analysis design methodology," presented at User Track of Design Automation Conference, 2009. (オンライン)，入手先 http://www.dac.com/46th/proceedings/slides/B_23.pdf (参照 2010-6-14).

[101] S. Tsukiyama, "Toward stochastic design for digital circuits - statistical static timing analysis," Asia and South Pacific Design Automation Conference, pp.760–765, 2004.

[102] S.V. Kumar, C.V. Kashyap, and S.S. Sapatnekar, "A framework for block-based timing sensitivity analysis," Design Automation Conference, pp.688–693, 2008.

[103] A. Datta, S. Bhunia, G.H. Choi, S. Mukhopadhyay, and K. Roy, "Speed binning aware design methodology to improve profit under parameter variations," Asia and South Pacific Design Automation Conference, pp.712–717, 2006.

[104] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," International Conference on Computer Aided Design, pp.607–614, 2003.

[105] 小松裕成，杉山広行，池田　弘，井口克己，備前直美，吉田裕司，伊藤則之，松岡英俊，新田　泉，本間克己，澁谷利行，"統計的タイミング解析とプロセサ設計への適用，" DA シンポジウム，pp.1–6, 2006.

[106] X. Li, J. Le, M. Celik, and L.T. Pileggi, "Defining statistical sensitivity for timing optimization of logic circuits with large-scale process and environmental variations," International Conference on Computer Aided Design, pp.844–851, 2005.

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

[107] N. Ito, H. Komatsu, H. Sugiyama, K. Iguchi, N. Bizen, and Y. Yoshida, "SSTA and its application to SPARC64^{TM} processor design," presented at User Track of Design Automation Conference, 2009.（オンライン），入手先 http://www.dac.com/46th/proceedings/slides/04U_2.pdf（参照 2010-6-14）.

[108] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variation and impact on circuits and microarchitecture," Design Automation Conference, pp.338–342, 2003.

[109] R.R. Rao, A. Devgan, D. Blaauw, and D. Sylvester, "Analytical yield prediction considering leakage/performance correlation," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.25, no.9, pp.1685–1695, 2006.

[110] A. Srivastava, D. Sylvester, and D. Blaauw, "Statistical optimization of leakage power considering process variations using dual-Vth and sizing," Design Automation Conference, pp.773–778, 2004.

[111] M. Mani, A. Devgan, and M. Orshansky, "An efficient algorithm for statistical minimization of total power under timing yield constraints," Design Automation Conference, pp.309–314, 2005.

[112] J. Darringer, E. Davidson, D.J. Hathaway, B. Koenemann, M. Lavin, J.K. Morrell, K. Rahmat, W. Roesner, E. Schanzenbach, G. Tellez, and L. Trevillyan, "EDA in IBM: Past, present, and future," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.19, no.12, pp.1476–1497, 2000.

[113] R.A. Burgelman, Strategy is destiny: how strategy-making shapes a company's future, pp.57–59, 147–149, The Free Press, ISBN 0-684-85554-2, 2002.

[114] S. Nachtsheim, "The Intel design automation system," Design Automation Conference, pp.459–464, 1984.

[115] A. Cao, A. Adalal, J. Bauman, P. Delisle, P. Dedood, P. Donehue, M. Dell'OcaKhouja, T. Doan, M. Doreswamy, P. Ferolito, O. Geva, D. Greenhill, S. Gopaladhine, J. Irwin, L. Lev, J. MacDonald, M. Ma, S. Mitra, P. Patel, A. Prabhu, R. Puranik, S. Rozanski, N. Ross, P. Saggurti, S. Simovich, R. Sunder, B. Sur, W. Vercruysse, M. Wong, P. Yip, R. Yu, J. Zhou, and G. Zyner., "CAD methodology for the design of UltraSPARC^{TM}-I microprocessor at Sun Microsystems Inc.," Design Automation Conference, pp.19–22, 1995.

[116] G. Spirakis, "Leading-edge and future design challenges - Is the classical EDA ready?," Design Automation Conference, pp.416–416, 2003.

[117] T. Karnik, S. Borkar, and V. De, "Sub-90 nm technologies–challenges and opportunities for CAD," International Conference on Computer Aided Design, pp.203–206, 2002.

[118] M. Guiney and E. Leavitt, "An introduction to OpenAccess – An open source data model and API for IC design," Asia and South Pacific Design Automation Conference, pp.434–436, 2006.

[119] Y. Inoue, "Open access overview "industrial experience"," Asia and South Pacific Design Automation Conference, pp.437–438, 2006.

[120] H. Ofek, "EDA vendor adoption," Asia and South Pacific Design Automation Conference, p.439, 2006.

[121] D.A. Papa, I.L. Markov, and P. Chong, "Utility of the OpenAccess database in academic research," Asia and South Pacific Design Automation Conference, pp.440–441, 2006.

[122] T. Sudo, T. Ohtsuki, and S. Goto, "CAD systems for VLSI in Japan," Proc. IEEE, vol.71, no.1, pp.129–143, 1983.

[123] D. MacMillen, M. Butts, R. Camposano, D. Hill, and T.W. Williams, "An industrial view of electronic design automation," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.19, no.12, pp.1428–1448, 2000.

付　　録

*1 CAD　Computer Aided Design の略で，コンピュータによる設計支援するソフトウェア．本論文では特に，プロセッサなどのチップ設計を支援するソフトウェアを意味する．

*2 ネットリスト　論理回路を構成する論理ゲートやマクロブロックなどの構成要素の接続情報．なお，ネットとは接続すべき構成要素のピンの集合を意味し，ネットをリストアップすることにより論理回路が表現される．

*3 セル　複数のトランジスタを組み合わせて作られた特定の機能（AND や OR など）をもつ回路のこと．この回路を複数組み合わせることにより論理回路が実現される．ゲートと呼ばれることもあるが，本論文ではセルで統一する．

*4 マクロ　標準的に用意されたセル以外で，セルより大きい論理規模で設計された回路のこと．

*5 ASIC　Application Specific Integrated Circuit の略で，ディジタルカメラやプリンタなどの特定用途向け IC のこと．

*6 コア　従来は CPU と呼ばれた機能であり，一つのチップに複数の CPU 機能が搭載され始めたため，一つの CPU 機能はコアと呼ばれるようになった．

*7 IP マクロ　IP は Intellectual Property の略で知的財産を意味し，マクロは機能ブロックを意味する．IP マクロは，新たに設計するのではなく，既に設計済みのマクロを意味する．

NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

*8 SoC　System on Chipの略で，システム全体が一つのチップに搭載されたもの．
*9 DFM　Design For Manufacturabilityの略で，チップ製造の容易性を考慮した設計を意味する．
*10 DFT　Design For Testabilityの略で，チップ製造後に個々のチップに対して行う良品・不良品を判断する試験を容易化する設計を意味する．
*11 FF　Flip-flopの略で，外からの制御により，論理的な0または1の1ビットの状態を保持することができる回路．
*12 ディレイ　信号が伝搬する際の遅延時間を意味する．遅延と呼ばれることもあるが，本論文ではディレイに統一する．
*13 クリティカルパス　ある設計単位でのタイミング解析の結果，ディレイが最も長いパスのこと．
*14 スルー　タイミング解析において，スルー（slew）は波形のなまり具合を意味し，波形の電圧レベルが例えば20%と80%に間の遷移時間を意味する．
*15 しきい値電圧　トランジスタの入力側で電圧が0Vから変化するとき，出力が変化する電圧のことであり，スレショルド電圧と呼ばれる．
*16 実効容量　セルの出力ピンに配線が接続される場合，セルの入力ピンから出力ピンまでのディレイはセルの出力ピンに接続される配線に影響を受ける．この影響を計算するために，出力ピンに接続される配線の影響を一つの容量Cで表したもの．
*17 リピータ　論理回路をレイアウトした際に，二つのセルの間の距離が長い場合，配線ディレイやクロストークノイズを低減させるために入れる1入力1出力のセルのこと．反転のない論理の場合はバッファと呼ばれ，反転がある論理の場合はインバータと呼ばれる．これら二つの総称がリピータである．
*18 ネット　ネットとは，論理回路の中で，接続すべき論理セルのピンの集合のこと．
*19 漏れ電流　トランジスタにおいて，動作とは関係なく漏れ流れる電流のこと．ソースとドレーンの間の漏れ電流やゲートと基板の間の漏れ電流などがある．半導体プロセスの微細化に伴って影響が大きくなってきた．
*20 キャッシュRAM　チップの外にある記憶装置より読み込んだデータの中で参照頻度の高いものをチップの中で保存しておくための高速な記憶領域のこと．
*21 レイアウト　物理設計の中で，セルの配置及びネットの配線をまとめてレイアウトと呼ぶ．
*22 RTL　Register Transfer Levelの略．論理を記憶素子であるレジスタの間で，組合せ回路を経由した信号の転送を記述することにより，論理を表現する抽象度のレベル．
*23 LVS　Layout Versus Schematicの略で，スケマティックとレイアウトが別々に行われるので，両者が一致していることの検証を意味する．なお，スケマティックとは，ネットリストに対応する回路図を意味する．
*24 ダミーメタル　チップを製造する際に，各配線層を形成した後で化学的機械的研磨により表面の平たん化を行う．ここでの平たん性を向上させるために，チップのレイアウト設計後に配線の疎密をチェックして，配線密度が一定以上になるように隙間に入れる配線をダミーメタルと呼ぶ．
*25 バジェット　トップダウンの階層設計において，下位階層に付与される制約のこと．面積，タイミングに関する制約のほかに，電力やノイズなど様々な制約がある．
*26 GDS　Graphic Database Systemの略．チップの製造データを電子ファイルで表現する際の業界標準フォーマットの名前．正式名称はGDS-II.
*27 DRC　Design Rule Checkの略で，本論文でのDRCはレイアウトに関する物理的なルールのチェックを意味する．
*28 ビア　複数の配線層を使って配線する場合，隣り合う上下の配線層の配線を電気的に接続するために，二つの配線層の間に置くもの．
*29 配線チャネル　各配線層において，配線が通ることができると指定されたところ．各配線層の平面において，チャネルで配線の通れるところを規定することにより，配線の際の経路の検索の空間を限定することができる．
*30 統計的加算　統計的加算は，二つのパスのディレイの確率密度関数をそれぞれ$f_1(t)$，$f_2(t)$としたとき，$f_1(t)$と$f_2(t)$の畳込み積分で計算される．
*31 統計的MAX　統計的MAXは，二つのパスのディレイの累積分布関数をそれぞれ$F_1(t)$，$F_2(t)$としたとき，$d(F_1(t)F_2(t))/dt$となる．
*32 EDAベンダ　EDAはElectric Design Automationの略で，チップなどの設計を支援する分野．EDAベンダは，EDAツールを販売する会社．
*33 サインオフ　半導体製造側に製造データを渡す際に，製造側がデータの受入れを承認すること．製造側

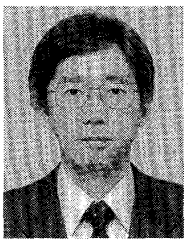
NII-Electronic Library Service

The Institute of Electronics, Information and Communication Engineers

が指定するルールとCADツールを使ってDRCを実行して，エラーがないことの確認を行うことで，承認される．

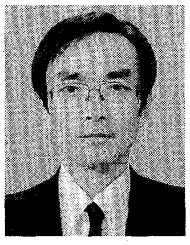
*34 API　Application Program Interfaceの略．データベースのフォーマットの詳細が分からなくても，アプリケーションプログラムからデータを取り出せるように準備した関数ライブラリのこと．

（平成22年8月24日受付，23年3月21日再受付）



伊藤　則之　（正員）

昭57京大・工・情報卒．同年富士通（株）入社．以来，メインフレーム，プロセッサ設計などコンピュータの設計手法及び設計自動化の研究開発に従事．現在，社会人として，筑波大学大学院システム情報工学コンピュータサイエンス専攻の博士後期課程に在学中．



安永　守利　（正員）

昭56筑波大・基礎工学類・物理工学卒．昭58同大大学院工学研究科修士課程了．同年（株）日立製作所入社，以来同社中央研究所において大型計算機ハードウェアの研究，ウェーハスケール集積回路研究に従事．平3〜4米国カーネギーメロン大学客員研究員兼務．平6筑波大学電子・情報工学系助教授．平16同大学院システム情報工学研究科教授．超高速VLSI，超高速信号伝送，進化ハードウェア，フォールトトレラントシステム，ニューラルネットワーク，遺伝的アルゴリズムの研究に従事．著書「Massively Parallel Artificial Intelligence（MIT Press）」「電磁気学演習（培風館）」（共著）ほか．博士（工学）．IEEE，情報処理学会会員．

NII-Electronic Library Service