MOS LSIの信号の流れる方向を決定するアルゴリズム

栗林元隆[†] 山田正昭[†] 竹内秀輝[†]
辻本順一[††] 外岡康宏[†††] 黒岩健太郎[†††]

本論文では，MOSトランジスタの信号の流れる方向を決定する効果的なアルゴリズムについて述べる．トランジスタレベルでのタイミング解析において，フォールスパスを除去する問題は，トランジスタの信号の流れる方向を決定する問題と深く関係している．本論文で提案したアルゴリズムは，フォールスパスの問題を解くのに非常に貢献している．計算機実験によれば，本提案アルゴリズムを実装したタイミング解析ツールは，ほとんど100%のトランジスタの信号の方向を決定した．このように高い確率で信号の流れる方向を決定できることにより，フォールスパスを除去するのみならず，パス探索時に双方向の場合を考える必要がなくなり，タイミング解析の処理時間も短縮できる．

A Signal Flow Direction Determination Algorithm for MOS LSI

Mototaka Kuribayashi,[†] Masaaki Yamada,[†] Hideki Takeuchi,[†]
Junichi Tsujimoto,[††] Yasuhiro Tonooka[†††]
and Kentarou Kuroiwa[†††]

This paper describes an efficient signal flow determination algorithm for MOS transistor circuits based on fundamental idea of signal flow direction. The false path problem at the transistor level is closely related to the problem of determining transistor signal flow. The proposed algorithm greatly contributes to solving such problems. According to experimental results, the algorithm can successfully determine signal flow direction for almost 100% of transistors. Such a high percentage of signal flow determination not only contributes to eliminating false paths, but also reduces the running time for timing analysis.

1. はじめに

この論文では，マイクロプロセッサーなどのデジタルロジック回路のトランジスタレベルのスタテックタイミング解析で重要となる信号の流れる方向を決定する手法を提案する．この手法は，STANT（Static Timing ANalyzer at Transistor-level）と呼ばれるタイミング解析ツールに実装されている．

スタテックタイミング解析は，信号の伝播に最も時間がかかるクリティカルパスを求めてくれる．設計者は，タイミング解析ツールがレポートしたクリティカルを調べて，タイミングの改善を行う．タイミングの改善は，たとえば，バッファを挿入すること，リピーターを挿入すること，トランジスタの幅を変えるなどである．

スタテックタイミング解析は，デジタルMOS VLSIのタイミング検証で広く使われてきた[1)~6),8),9)]．スタテックタイミング解析は，入力ベクトルを必要としないのみならず，計算時間も短くてすむ．一方，ダイナミックなシミュレータは，クリティカルパスを活性化する入力ベクトルを必要とする．しかしながら，回路の規模が増加するにつれて，ダイナミックにシミュレートすることが不可能になってきいる．また，SPICEなどの回路シミュレータは，処理時間の増大のために，チップレベルで大規模回路を取り扱うことは不可能になってきている．スタテックタイミング解析は，回路の時間的な動作を考慮せず，回路を簡単な遅延モデルとして扱い回路の解析を行うので，大規模回路を扱うことができる．

トランジスタレベル（スイッチレベル）のタイミング解析は，最近広く用いられてきている．その理由の1つは，設計者が500～800 MHzで動作するようなマイクロプロセッサーなどにダイナミック回路などのよ

† 株式会社東芝半導体設計・評価技術センター
Semiconductor DA & Test Engineering Center, Toshiba Corporation
†† 株式会社東芝デバイス技術研究所
ULSI Device Engneering Laboratory, Toshiba Corporation
††† 株式会社数理システム
Mathematical Systems Incorporation

うなより高速に動作する回路を使い始めたからである．もう1つの理由は，今日のようなディープサブマイクロンの設計では，より精度の高いタイミング解析が必要とされているから，ゲートレベルでなくトランジスタレベルで解析したい，ということである．

本論文の第1の目的は，「トランジスタの信号の流れる方向」に関して明確な定義を与えることである．これまでに提案されたきた方法[4),7),11)]は，信号の流れる方向の定義を明示していない．ほとんどの場合，信号の流れる方向は直観的にあきらかであるが，ある方法の正当性を証明するためには定義を陽に示す必要がある．本論文では，まず信号の流れる方向の定義を明確にする．

MOSトランジスタは，その性質として，信号が双方向に（bidirectional）伝播する．トランジスタの信号の流れがどちらか1方向と分かってしまえば，クリティカルパスの探索は，その1方向のみ考えればよいので，より簡単になる．一方，ゲートレベルやブロックレベルでの解析では，信号の流れの方向は自明である．

信号の流れの方向を誤って決定すれば，フォールスパスがレポートされるかもしれない．バレルシフター（barrel shifter）のようなパストランジスタがつながった回路では，そのような問題も生じやすい．**図1** (a) に例を示す．また，トランジスタの信号の流れる方向を適切に決めることができなければ，トランジスタは双方向であるために，両方向のパス探索をする必要があり，処理時間が大きくかかる．

Crystal[3)]の方法は，人手で信号の流れる方向を指定するものである．図1は，2入力のマルチプレクサ（multiplexer）の例である．Crystalは，パストランジスタを流れる信号についての情報がなければ，実際にはけっして起こらない図1 (a) の破線のパスを調べに行く．図1 (a) のパスは，フォールスパスである．もし，ユーザが，信号はいつも図1 (b) のように，2つのマルチプレクサのパストランジスタへ，左側から右側へ流れると指示をするならば，Crystalは，誤ったパスを調べることはない．マルチプレクサの場合には，信号はいつも1方向に流れるが，ユーザは各パストランジスタについて，図1 (b) のように，どちら側が信号のソースであるか示さなければならない．人手による信号の流れの方向の指定は，非常に大変な仕事となる．たとえば，ユーザは2240個のトランジスタからなる32ビットのバレルシフターでは，854個のパストランジスタを指定しなければならない．

一方，TV[4)]は，9つのルールを用いて，nMOSのパストランジスタの信号の流れの方向を自動的に決定する．ルールの1つは，電源（VDD）・グランド（VSS）に接続されているトランジスタのチャネル（channel）は，信号の流れのソースになるというものである．他のルールは，キルヒホッフ（Kirchoff）の法則などである．TVの開発者Jouppiは，これらのルールを文献7)で一般化している．しかしながら，実回路では，このルールベースの手法は，すべてのトランジスタの信号の流れる方向を決定するには十分でない．

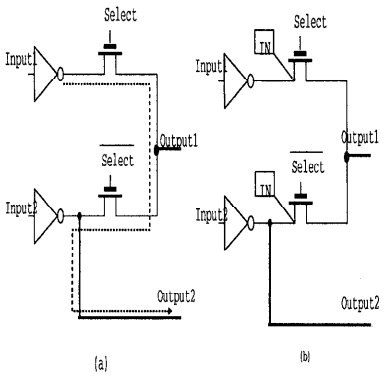


図1 Crystalの信号の流れる向きのコントロールの例
Fig. 1 Crystal's control of signal flow direction.

文献11)は，トランジスタの信号の流れる方向を2段階の処理で決定している．まず，structure-basedと呼ばれる処理では，グラフ理論に基づくアルゴリズムで決定する．次に，もしstructure-basedで方向が決定できないトランジスタがあれば，rule-basedと呼ばれる処理で決定される．そのルールは，(1) Inverter Driven, (2) Gate Driven, (3) Source or Sink, (4) Exclusion Detection, (5) Precharge Node Driven, (6) Logic Implicationの6つある．このうち，(1), (2), (3)の3つのルールは簡単なルールであり，文献4), 7)と同様または拡張したものである．この6つのルールを適用する順番は性能に影響を与える．提案されている順序は，(5)を適用した後，(1) → (2) → (3) → (4)を繰り返し適用する．そして，最後に，最も時間がかかる(6)が適用される．barrel shifterは，(6)が適用されるまで信号の流れる方向は決定できない．この方法は，様々なルールを適用することで信号の流れる方向を決定するものであり，統一的な原理に基づくものでない．

本論文の第2の目的は，トランジスタの信号の流れる方向を決定する効率的なアルゴリズムを提案することである．本論文で提案する手法は，文献11)と比較したときに，ルールベースがなくても1つの統一的な

原理で扱えるというメリットを持つ．その原理のために，回路中の1つのトランジスタのソースまたはドレインにNon-Z特性と呼ぶ概念を与える．そのトランジスタを削除した状態で，ノードがけっしてハイインピーダンスにならないノードはNon-Z特性を帯びているとする．反対にNon-Z特性を帯びていないノードは，Not-Non-Z特性を帯びているノードとする．本論文では，回路をレシオレス回路に限ったときに，トランジスタの信号の流れる方向は，Non-Z特性を帯びたノードからNot-Non-Z特性を帯びたノードであるという，1つの統一的な原理に基づいたNon-Z手法を提案する．

提案する信号の流れる方向の決定は，Non-Z手法で回路中のトランジスタを決定することができる．しかし，マイクロプロセッサーなどの論理回路では，従来から提案されているルールベースの処理は効果的である．そこで，本論文では実用性に重きを置いた手法を採用する．まず，単純な論理回路をパターンマッチングで抽出して，信号の流れの方向を決定する．次に，文献4)のようなルールベースな手法を使って，信号の流れの方向を決定する．上記の前処理を行った後，信号の流れる方向の決定していないトランジスタに対して，Non-Z手法を適用する．

なお，本論文で提案するNon-Z手法は，ダイナミック回路や双方向のトランジスタを含む回路にも適用できるが，レシオ回路には適用できない．

2章では，信号の流れる方向の定義およびNon-Z特性・Not-Non-Z特性の定義を述べる．3章では，本論文で提案する信号の流れる方向の決定アルゴリズムについて述べる．4章では，前章で述べた，信号の流れる方向を決定するアルゴリズムのコアとなるNon-Z手法というアルゴリズムについて述べる．計算機実験の結果は，5章で述べる．最後に，6章で結論を述べる．

2. 信号の流れる方向の定義とNon-Z特性の定義

2.1 信号の流れる方向の定義

［定義］

あるトランジスタのソース・ドレインノードをA，Bとし，これらが互いに異なる電位にあるとする．このトランジスタがONになったとき，つねにノードBの電位がノードAの電位に変化し，ノードAの電位が変化しないとき，信号の流れる方向は，ノードAからノードBであるとする．

図2は，信号の流れる方向の定義を説明するための図である．一般性を失うことなく，TRはNMOS-

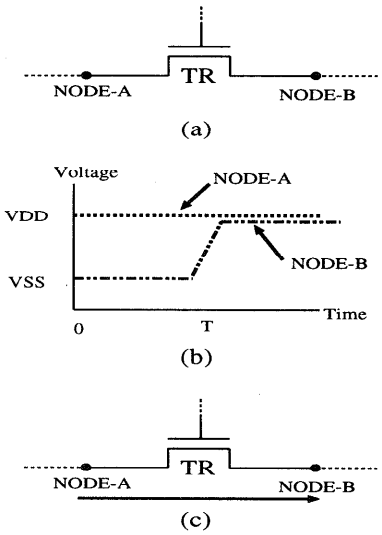


図2 信号の流れる方向の定義を説明する図
Fig. 2 Definition of signal flow direction on a transistor.

FETまたはPMOSFETであるとする．また，一般性を失うことなく，時刻0でノードNODE-AがHIGHであり，ノードBがLOWであるとする．また，時刻0では，トランジスタTRはOFFしているとする．図2(a)は，トランジスタTRの2つのノードNODE-AとNODE-B接続を示している．図2(b)は，横軸が時間を縦軸がノードの電位を表す．破線は，ノードNODE-Aの電位の時間変動を示す．2点鎖線は，ノードNODE-Bの電位の時間変動を示す．時刻がTになったときに，トランジスタTRがONしたとする．このとき，ノードNODE-Bの電位は，HIGHに変化していく．このときに，信号は，ノードNODE-AからノードNODE-Bに流れたと定義する．信号の流れる方向は，図2(c)の矢印の向きであるとする．

2.2 Non-Z特性とNot-Non-Z特性の定義

［定義］

回路中の1つのトランジスタのソースまたはドレインに「Non-Z特性」という概念を次のように与える．そのトランジスタを削除した状態で，ノードがけっしてハイインピーダンス〔high-impedance（Hi-Z）〕にならないノードは，「Non-Z特性」を帯びていると定義する．反対にNon-Z特性を帯びていないノードを「Not-Non-Z特性」を帯びているノードと定義する．

図3は，Non-Z特性・Not-Non-Z特性を説明するための例である．

ノードAは，いつもVDDかVSSへのDCパスがあるので，ノードAはNon-Z特性を帯びたノードである．一方，ノードBは，Non-Z特性を帯びない

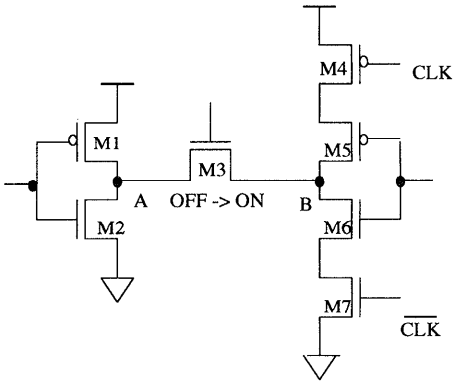


図3 Non-Z 特性，Not-Non-Z 特性を説明するための例
Fig. 3 An example explaining Non-Z property and Not-Non-Z property.

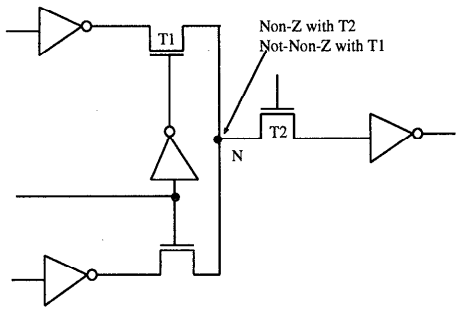


図4 あるノードが Non-Z 特性と Not-Non-Z 特性をとる例
Fig. 4 An example that a node takes both Non-Z property and Not-Non-Z property.

ノード，すなわち，Not-Non-Z 特性を帯びたノードである．もし，CLK が HIGH であれば，トランジスタ M4，M7 は，OFF になり，ノード B は，ハイインピーダンス状態になる．

最初，トランジスタ M3 は，OFF であるとする．そして，現時刻に ON に変わったとする．変化の後，ノード B は，ノード A と同一の電位になる．したがって，信号の流れの方向は，A から B である．この例では，信号はトランジスタ M3 に関して Non-Z 特性を帯びたノード A から Not-Non-Z 特性を帯びたノード B に信号が流れる．

Non-Z 特性は 1 つのトランジスタのソースまたはドレインに対して，与えられるものである．ノードごとに与えられるものではない．**図 4** は，Non-Z 特性を説明するための図である．図 4 のように，2 つのトランジスタ T1 と T2 が接続していたとする．接続するノードを N とする．ノード N は，トランジスタ T1 に関して Not-Non-Z 特性を帯びたノードである．一方ノード N は，トランジスタ T2 に関して Non-Z 特性を帯びたノードである．すなわち，ノード N は，Non-Z 特性を帯びることもあるし，Not-Non-Z 特性を帯びることもある．

2.3 レシオレス回路とレシオ回路

レシオレス回路（rationless circuits）とは，VDD から VSS に直流パスを含まない回路である．ゲートは，パストランジスタなどを除けば，駆動素子と負荷素子で構成されるのが通常である．レシオレス回路は，出力信号の電位が両者のコンダクタンスの比できまることになるが，一方が OFF 状態となって出力レベルが定まる形式の回路である．これに対して，ともに ON の状態の素子のコンダクタンスの比で出力レベルが決まる形式の回路はレシオ回路と呼ばれている．レシオレス回路には，PMOS と NMOS とが対になって構成された CMOS 回路，クロック系 CMOS 回路，NMOS のダイナミック回路などである．

次の節では，レシオレス回路に関しては，トランジスタ上に流れる信号の方向は，一般的に，Non-Z 特性を帯びたノードから Not-Non-Z ノードを帯びたノードに流れることを示す．

2.4 信号の流れる向きに関する定理

［定理］

レシオレス回路では，トランジスタ上に信号が流れる方向は，Non-Z 特性を帯びたノードから Not-Non-Z 特性を帯びたノードである．

（定理の証明）

図 5 は，本定理を証明するために用いる図である．1 つのトランジスタ TR があり，そのソースノード・ドレインノードを L，R とする．ノード L は，トランジスタ TR に関して Non-Z 特性を帯びたノードであり，ノード R はトランジスタ TR に関して Not-Non-Z 特性を帯びたノードであると仮定する．

Glu は，VDD からノード L への DC パスに接続しているトランジスタのグループとする．Gld は，VSS からノード L への DC パスに接続しているトランジスタのグループとする．同様に Gru は，VDD からノード R への DC パスに接続しているトランジスタのグループとする．Grd は，VSS からノード R への DC パスに接続しているトランジスタのグループとする．

時刻 0 でトランジスタ TR は OFF しているとする．

まず，ノード L は時刻 0 で，HIGH であり，ノード R は時刻 0 で LOW であると仮定する．いま，ある時刻にトランジスタ TR が OFF から ON に変化したとする．このとき，ノード L は Non-Z 特性を帯びているので，L から VDD への直流パスが存在する．このとき，ノード R から VSS へのパスが存在すると仮定する．そうすれば，VDD → R → TR → L → Grd

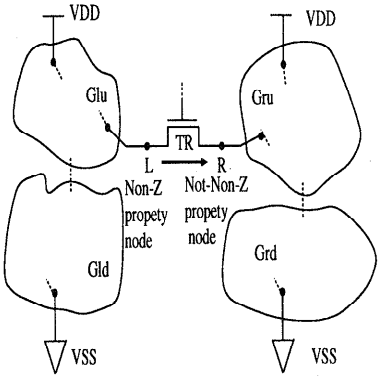


図5 信号はNon-Z特性を帯びたノードからNot-Non-Z特性を帯びたノードに流れることを示す例
Fig. 5 A signal flows from a Non-Z property node to a Not-Non-Z property node.

→ VSSの直流パスが現れるので，この回路はレシオレス回路でないことになり，矛盾が生じる．したがって，TRがONになったときには，RからVSSへの直流パスは存在しない．それゆえ，TRがOFFからONするときに，HIGHという電位がノードRに伝播されることになる．つまり，トランジスタ上を信号が流れる方向は，Non-Z特性を帯びたノードLからNot-Non-Z特性を帯びたノードRへである．

同様に，ノードLは時刻0で，LOWであり，ノードRがHである場合も証明することができる．

それゆえ，レシオレス回路は，トランジスタ上を信号が流れる方向は，Non-Z特性を帯びたノードからNot-Non-Z特性を帯びたノードへである．

3. 信号の流れの方向を決定するアルゴリズム

Non-Z手法は，すべてのトランジスタの信号の流れの方向を決定するのに適用することができる．しかし，マイクロプロセッサーなどの論理回路には，ルールに基づく方法でかなりの部分が決定できる．ルールに基づく手法は処理が速いので有効である．したがって，実用性を重視したタイミング解析ツールSTANTは，まず前処理として，以下の(1)と(2)を行う．その後で，信号の流れる方向が決定していないトランジスタに対してNon-Z手法を適用する．

なお，すべてのトランジスタは，最初，双方向であると仮定して進める．

(1) インバーターやNAND，NORなどの単純ゲートは，パターンマッチングして見つける．単純ゲートの出力ノードは，信号の流れの始点と見なすことができるので，このノードにつながるパストランジスタの信号の流れる方向を決定するのに使う．

(2) 8つの基本的なルールを適用する．

(3) Non-Z手法を適用する．

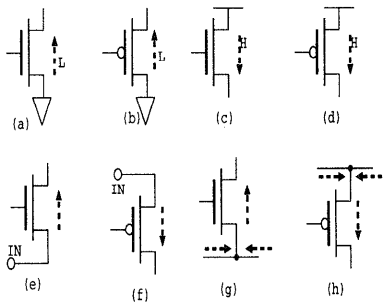


図6 信号の流れる方向を決定する基本的なルール
Fig. 6 Fundamental rules determining a signal flow direction.

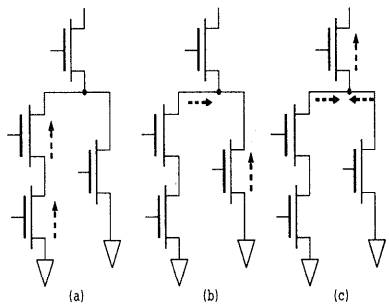


図7 信号の流れる方向を決定する基本的なルールの適用
Fig. 7 Example of applying fundamental rules.

基本的なルールは，TV[4]のルールと同様なものである．図6は，基本的なルールを示す．図6のルール(a)から(d)は，NMOSまたはPMOSのトランジスタのソース・ドレインがVDDまたはVSSに接続されている場合を示す．図6のルール(e)から(f)は，NMOSまたはPMOSのトランジスタのソース・ドレインがけっしてハイインピーダンスにならない入力ノードに接続している場合であり，その信号の流れの方向は決定することができる．図6のルール(g)から(h)は，すべての他の信号の流れる方向が1方向のみである場合，その信号の流れる方向を決定することができる．

図7は，上記のルールを応用した例を示す．STANTは，トランジスタの信号の流れる方向以下の手順で行う．

(1) 左下sssのNMOSの信号の流れる方向をルール(a)を用いて決定する．その次に，ルール(g)を適用して，左上のNMOSの信号の流れる方向を決定する．

(2) 右のNMOSの信号の流れる方向をルール(a)

を用いて決定する．

(3) ルール (g) を適用して，一番上の NMOS の信号の流れる方向を決定する．

4. Non-Z 手法

図 8 は，信号の流れる方向を決めるアルゴリズムのフローチャートを示す．まず，ステップ S1 では，3 章で述べた，インバーターや NAND や NOR などの単純ゲートがパターンマッチングで認識される．これらの出力に接続しているパストランジスタの方向を決定する．次に，ステップ S2 では，3 章で述べた 8 つの基本的なルールの適用と，それらを組み合わせて応用した図 7 のような処理で信号の向きを決定する．

その後，信号の流れる方向が決まっていないすべてのトランジスタを 1 つずつ選択していく（ステップ S3）．各トランジスタ上を流れる信号の方向は，以下に詳しく述べる “Non-Z 手法” で決定される．ステップ S4 とステップ S5 では，Non-Z-procedure が呼ばれる．Non-Z-procedure は，引数を 3 つとり，第 1 の引数が Non-Z 特性を帯びたノードか，または Not-Non-Z 特性を帯びたノードか検証して回答を戻す．1 つのトランジスタに対して，両ノードについて，それぞれ第 1 引数として与えて検証する．したがって，ノードのペア (X,Y) と (Y,X) についてこの Non-Z procedure が呼ばれる．第 3 の引数 L は，探索の深さを表すもので，ユーザが設定するパラメータである．ステップ S6 とステップ S7 では，信号の流れる方向は，Non-Z 特性を帯びたノードから Not-Non-Z 特性を帯びたノードに流れると決定される．もし，どちらのノードも Not-Non-Z 特性を帯びたものであれば，このアルゴリズムでは決定できないのでエラーメッセージを出力する（ステップ S8）．

図 9 は，Non-Z 手法のアルゴリズムを記述したものである．Non-Z 手法は，「注目しているトランジスタについて，第 1 の引数 X が Non-Z 特性を持つノードか，Not-Non-Z 特性を持つノードか」を決定するものである．決定の原理は，「ノード X がハイインピーダンス（Hi-Z）になるという」仮定を立てて，仮定どうしに矛盾が起こるか，矛盾が起こらないかを調べる．第 2 の引数 Y は，注目している 1 つのトランジスタを記憶しておき，アルゴリズムの中でノード X とノード Y をソース・ドレインに持つトランジスタを除外するために用いる，補助的なノードである．第 3 の引数 “level” は，探索の深さ示す．その最大値 Max_level は，アルゴリズムの開始前に与えておく．この値は経験的な値であり，実際の回路では，2 や 3 で十分な効果が得られている．Max_level を大きくすると処理時間がかかるので，処理時間と信号の方向決定数のトレードオフになってくる．

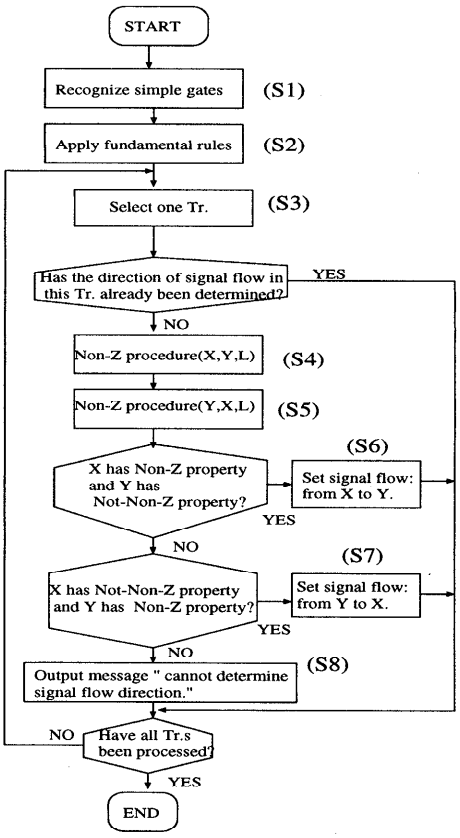


図 8 信号の流れる方向を決定するアルゴリズム
Fig. 8 Algorithm determing signal flow direction.

以下，図 9 を用いて Non-Z 手法を説明する．Non-Z 手法（以下，Non-Z-pro(X,Y,level) と記す）は，ノード X が Non-Z 特性を帯びたノードであるか，Not-Non-Z 特性を帯びたノードであるかを返す処理である．まず，「あるノードが OFF している」などの仮定の集合 Q と，トランジスタの集合 T は，グローバルな変数とする．Non-Z-pro(X,Y,level) は，その中でサブルーチン is-All-off(T,X,level) をコールしている．is-All-off(T,X,level) は，再帰的な処理を行うものであり，トランジスタの集合 T から 1 つずつトランジスタを取り出して，そのトランジスタが OFF しているか，または反対側のノードが Hi-Z であるかを調べる．トランジスタの集合 T の個数は，再帰処理の過程で 1 つずつ減って，最後の 1 つのトランジスタまで調べられる．

以下，図 9 に従い，Non-Z-pro(X,Y,level) の処理を説明する．step-1 では，ノード X が VDD や VSS や

```
Q: Assumption Queue
T: Set of Transistor

Non-Z-proc(X,Y,level)
{
  (step-1)
  if("X is VDD or VSS or primary-input or primary-output")
   return "X is Non-Z property node";

  (step-2)
  if( level > Max_level)
    retrun "X is Not-Non-Z property node";

  (step-3)
   T <-- All tr.s connecting X via channel, except tr. between X and Y;

   (step-4)
   if(is-All-off(T,X,level))
      return "X is Not-Non-Z property node";
   else
      return "X is Non-Z property node";
}



is-All-off(T,X,level)
{
   if(T is empty) return true;  (step-a)

   Remove a transistor Ti from T;(step-b)

   Add assumption "Ti is off" to Q;  (step-c)

   if( any inconsistency in Q){   (step-d)
      remove assumption "Ti is off" from Q;(step-e)

      if(Non-Z-pro(the other S/D node of Ti,X,level+1)
        == "Non-Z property node")   (step-f)
        return false; (step-g)
      else
        return is-All-off(T,X,level);  (step-h)
   }
   else
        return is-All-off(T,X,level);  (step-i)
```

図9 Non-Z-procedure(X,Y,LEVEL)
Fig. 9 Non-Z-procedure(X,Y,LEVEL).

プライマリー入力およびプライマリー出力であれば，Non-Z-pro(X,Y,level) は，「X は Non-Z 特性を帯びたノード」との戻り値を返して処理を終了する．step-2 では，探索の深さが最大値を超えれば，処理を中断して，「ノード X は Not-Non-Z 特性である」を帰り値として，この処理から抜ける．step-3 では，ノード X にチャネルで接続するすべてのトランジスタを見つけて，それらをトランジスタの集合 T に入れる．ここでは，この X と Y をソース・ドレインに持つトランジスタは除外する．

step-4 では，トランジスタの集合 T と，ノード X と探索の深さ level をサブルーチン is-All-off に与える．全トランジスタの 1 つ 1 つについて，そのトランジスタが OFF しているか，あるいは X と反対側のノードが Hi-Z になりうるか検証される．このサブルーチンが "true" を返せば，「ノード X は Not-Non-Z 特性を帯びたノードである」を返し，"false" を返せば，「ノード X は Non-Z 特性を帯びたノードである」を返す．

以下，サブルーチン is-All-off(T,X,level) の処理を，図 9 に従って説明する．step-a では，トランジスタの集合 T が空がどうか調べ，空であれば "true" を返す．step-b では，トランジスタの集合 Q から 1 つのトランジスタ Ti（i は添え字）を取り出して削除する．step-c では，「Ti が off」という仮定を集合 Q に追加する．step-d では，仮定の集合 Q の中に互いに矛盾するものがあるかを調べる．矛盾があれば，step-e で，先に Q に入れた仮定を Q から取り除く．step-f では，トランジスタ Ti のノード X とは反対側のソース/ドレインノードが Non-Z 特性のノードであるか Not-Non-Z 特性のノードであるかを調べるために Non-Z-proc をコールする．このとき level は，1 つ進み level+1 が第 3 引数に渡される．その結果，Non-Z 特性のノードであると返されてきたら，step-g で "false" を返す．逆に，Not-Non-Z 特性のノードであると返されてきたら，step-i の処理に行く．step-i では，このサブルーチンの入力のときのトランジスタの集合 T の要素数が 1 つ減った集合 T を is-All-off に与えて再帰的にコールする．また，step-d で Q に矛盾がない場合にも，このサブルーチンの入力の時のトランジスタの集合 T の要素数が 1 つ減った集合 T を is-All-off に与えて再帰的にコールする．

図 10 は，Non-Z 手法が効果的に適用される回路例である．IN0～IN3 は，プライマリー入力である．O0～O3 は，プライマリー出力である．S0～S2 は，セレクト操作を行うプライマリー入力である．T1～T24 は，NMOS トランジスタである．IVI0～IVI3，IVO0～IVO3，IVS0～IVS3 は，インバーターである．

信号の流れる方向を決定するアルゴリズムを示す図 8 のステップ S1 やステップ S2 だけでは，トランジスタ T9 の信号の流れる方向は，N6 → N7 か N7 → N6 か決定できない．したがって，図 10 (a) の破線で示された I0 → N1 → T1 → N2 → T4 → N7 → T9 → N6 → T10 → N11 → T15 → N10 → T16 → N15 → T23 → N16 → O3 というパスがレポートされる．このパスは，フォールスパスがある．トランジスタ T9 を流れる信号の方向が，N7 → N6 となっている点と，トランジスタ T15 を流れる信号の方向が N11 → N10 となっている点が誤っている．

Non-Z 手法を用いれば，図 10 (b) のような正しいパスが求められる．すなわち，信号は，トランジスタ T9 上を流れる信号の方向は，ノード N6 → ノード N7 と

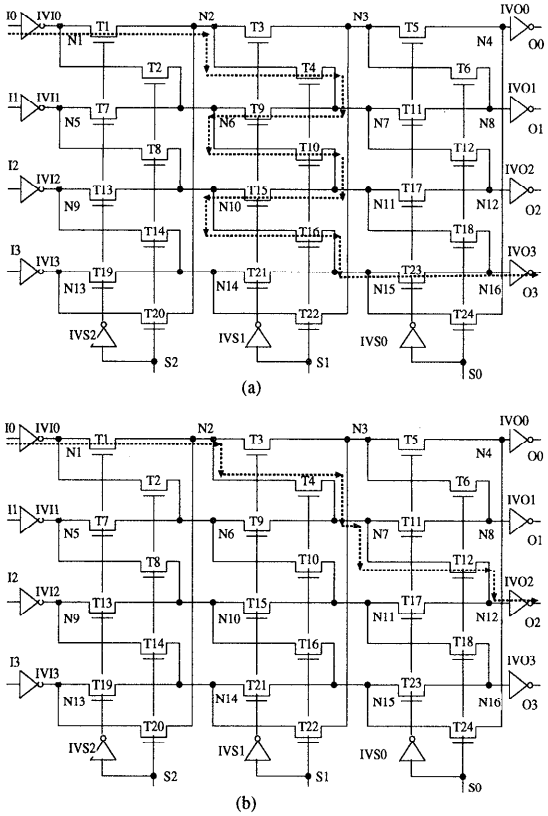


図 10 Non-Z procedure が効果的に適用される一例 barrel shifter

Fig. 10 The Non-Z procedure determined effectively signal flow direction in a barrel shifter.

決定できる．また，トランジスタ T15 上を流れる信号の方向は，ノード N10 → ノード N11 と決定される．このようにして，Non-Z 手法を用いれば，図 10 (b) に破線で示すような，I0 → N1 → T1 → N2 → T4 → N7 → T12 → N12 → O2 という正しいパスが求められる．

以下，一例として，トランジスタ T9 上を流れる信号の方向は，N6 → N7 へであることを示す．そのためには，ノード N6 がトランジスタ T9 に関して Non-Z 特性を帯びたノードであり，ノード N7 がトランジスタ T9 に関して Not-Non-Z 特性を帯びたノードであることを示せばよい．

まず，ノード N6 がトランジスタ T9 に関して Non-Z 特性を帯びたノードであることを導いていく．トランジスタ T9 に関して Non-Z-proc(N6,N7,1) を適用する．説明を分かりよくするために，図 9 に示した一般的な Non-Z-proc と一部対応がとれていない部分があるので注意願いたい．仮定が入れられる配列 Q がどのように変化していくかを示したのが**図 11** (a-1)～(a-8) である．また，トランジスタ T9 に関して Non-Z-procedure(N7,N6,1) を適用する．配列 Q がどのように変化していくかを示したのが図 11 (b-1)～(b-3) である．

まず，ノード N6 に接続するトランジスタとして T2, T7, T10 が選ばれる．まず，「T2 が OFF，すなわち S2 が LOW」という仮定が配列 Q に入れられる．次に，「T7 が OFF，すなわち S2 が HIGH」という仮定が追加される．次に，「T10 が OFF，すなわち S1 が LOW」という仮定が入れられる．この状態を示したのが，図 11 (a-1) である．配列 Q に入っている仮定が互いに矛盾しないか検証する．その結果，「T2 が OFF，すなわち S2 が LOW」と，「T7 が OFF，すなわち S2 が HIGH」は矛盾していることが分かる．

次に，「T2 のノード X と反対側のノードが Hi-Z」という仮定を配列 Q に入れる．次に，「T7 が OFF，すなわち S2 が HIGH」という仮定が追加される．次に，「T10 が OFF，すなわち S1 が LOW」という仮定が入れられる．この状態が，図 11 (a-2) である．配列 Q に入っている仮定に矛盾がないか検証する．その結果，「T2 のノード N6 と反対側のノードが Hi-Z」になることは，T2 のノード N6 と反対側のノードは，インバーター IVI0 の出力であり，Hi-Z にはなりえない．ここで，矛盾が生じた．

以下同様にして，Non-Z-procedure に従って処理を進めていくと，図 11 (a-3)～(a-8) が得られる．これで，すべての仮定が調べられ，それらがすべて矛盾することが分かる．以上から，ノード N6 がトランジスタ T9 に関して Non-Z 特性を帯びたノードであるという結果が得られる．

次に，ノード N7 がトランジスタ T9 に関して Not-Non-Z 特性を帯びたノードであることを導く．トランジスタ T9 に関して Non-Z-proc(N7,N6,1) を適用する．仮定が入れられる配列 Q がどのように変化していくかを示したのが図 11 (b-1)～(b-3) である．図 11 (b-1) と図 11 (b-2) は，配列 Q 内の仮定に矛盾がある．図 11 (b-3) の場合は，矛盾する仮定は存在しない．また，仮定「T4 is OFF, i.e. Gate of T4 is LOW」は，T4 のゲートノードは，プライマリーインプット S1 が LOW であることに行きついている．また仮定「T12 is OFF, i.e. Gate of T12 is LOW」は，T12 のゲートノードは，プライマリーインプット S0 が LOW であることに行きついている．上記 2 つの仮定と仮定「N8 of T11 is Hi-Z」とは矛盾しない．以上から，ノード N7 がトランジスタ T9 に関して Not-Non-Z 特性を帯びたノードであるという結果が得られる．

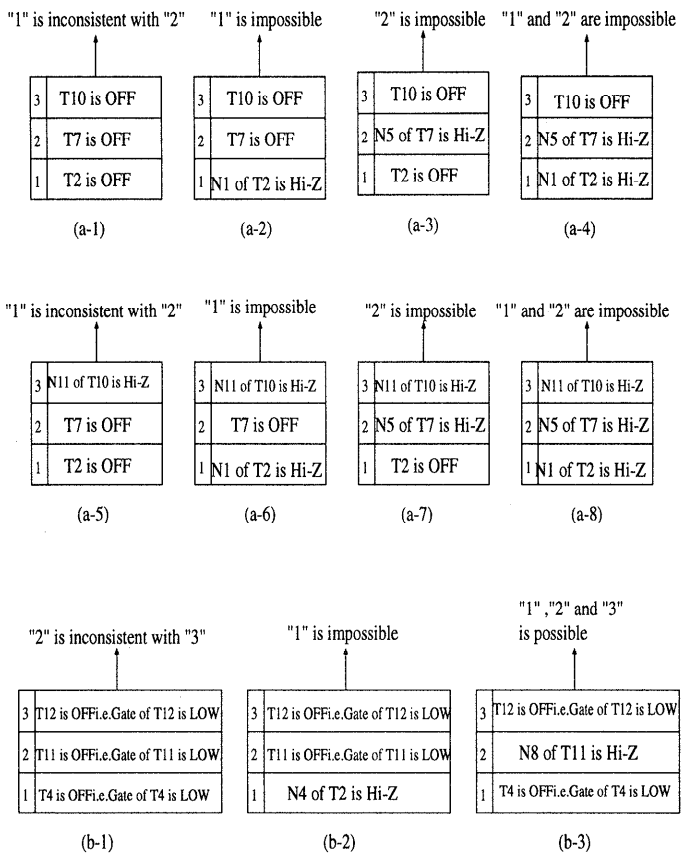


図 11 配列 Q の操作
Fig. 11 Operation of array Q.

5. 計算機実験の結果

信号の流れる方向を決定するアルゴリズムをスタティックタイミング解析プログラム STANT に実装した．32 MIPS の RISC ワークステーションを用いて，性能を評価した．ワークステーションは，192 メガバイトの主メモリと 400 メガバイトのスワップを持っている．

評価のポイントは，以下の 3 点である．Non-Z 手法にかかわるパラメータ Max_level は，1 に設定して実行した．

- トランジスタの信号の流れる方向の自動的決定率（%）
- 処理時間
- 出力された最長 10 本のクリティカルパスにフォールスパスが存在するかどうか

評価に用いたデータは，トロンアーキテクチャーのマイクロプロセッサーの barrel shifter と，8 ビットのリップル桁上げ加算器（adder8）と，RISC マイクロプロセッサ R10000 の部分回路（CMP, CSA, CSV）と約 5000 トランジスタの回路と約 10000 トランジスタの回路である．

表 1 は，Non-Z 手法がトランジスタの信号の流れる方向を決定した結果を示す．(S1) and (S2) の欄は，3 章で述べた信号の流れる方向を決定する処理のうち最初の 2 つ，すなわち，単純ゲートの認識の利用 (S1) と基本的なルールの適用 (S2) で決まったトランジスタ数を示す．Non-Z の欄は，Non-Z 手法が決定したトランジスタ数を示す．

Non-Z 手法は，barrel shifter の 2222 トランジスタのうち 334 個のトランジスタの信号の流れる方向を決定したことを示す．これは，単純ゲートの認識の利用と基本的なルールで決まった 1888 個のトランジスタと合わせると 100% の信号の流れを決定したことになる．また，Non-Z 手法は，データ I2ADD では 94 個，データ 5K-Tr. では 254 個，データ 10K-Tr. では 514 個のトランジスタの信号の流れる方向を決定している．

表1 Non-Z procedureが信号の流れる方向を決定したトランジスタ数
Table 1 Numbers of transistors which signal flow direction were determined by Non-Z procedure.

Circuit	#tr.	(S1) and (S2)	Non-Z
barrel shifter	2222	1888	334
I2ADD	1938	1844	94
adder8	270	270	0
CMP	58	58	0
CSA	168	160	4
CSV	26	25	0
5K-Tr.	5370	5116	254
10K-Tr.	10830	10316	514

表2 信号の流れる方向を決定した率（市販ツールとの比較）
Table 2 Ratios of transistors which signal flow direction were determined (Comparison with a commercial tool).

Circuit	# Tr.	STANT (%)	commercial (%)
barrel shifter	2222	100.00	100.00
adder8	270	100.00	100.00
CMP	58	96.55	86.21
CSA	168	97.62	77.38
CSV	26	100.00	84.62
I2ADD	1938	96.70	93.09
5K-Tr.	5370	100.00	90.35
10K-Tr.	10830	100.00	90.42

表2 は，信号の流れる方向が自動決定されたトランジスタ数の比率を，市販ツールと比較した結果を示す．使用した市販ツールは，マイクロプロセッサーのタイミング検証に広く使われいるものである．このツールは，トランジスタレベルで解析することもできるし，キャッシュなどのアナログ回路を含むブロックに対してはブラックボックス化して解析することもできる．

STANTは，ほぼ100%トランジスタの信号の流れる方向を決定している．一方，市販ツールはbarrel shifterを自動でタイミング解析することができなかった．その理由は，トランジスタの信号の流れる方向を決定できないためである．なお，市販ツールには，信号の流れる方向を人手で指定する機能を持っている．市販ツールもこの機能を使えばbarrel shifterを解析ができた．

表3 は，信号の流れる方向を自動決定できなかったトランジスタ数を，文献11)と比較した結果を示す．文献11)は，トランジスタの信号の流れる方向を2段階の処理で決定している．まず，structure-basedと呼ばれる処理では，グラフ理論に基づくアルゴリズムで決定する．次に，もしstructure-basedで方向が決定できないトランジスタがあれば，rule-basedと呼ばれる処理で決定される．表3において，文献11)の欄は，structure-basedの処理で未決定となったトランジスタ数と，未決定となったトランジスタに対してrule-basedが呼ばれて決定できなかったトランジスタ数を示す．

表3 信号の流れる方向を決定できなかったトランジスタ数（文献11)との比較）
Table 3 Numbers of transistors which signal flow direction were not determined (Comparison with Ref. 11)).

Circuit	# Tr.	STANT	11)
CVSL	29	2	$11 \rightarrow 0$
FF	10	0	$2 \rightarrow 0$
SHIFT	56	0	$16 \rightarrow 0$

表4 CPU時間の比較（STANTと市販ツール）
Table 4 Comparison of CPU Time with STANT and Commercial tool.

Circuit	#transistor	STANT (sec)	commercial (sec)
adder8	270	0.4	14.4
CMP	58	0.2	10.2
CSA	168	0.4	12.8
CSV	26	0.1	10.8
I2ADD	1938	15.4	33.9
5K-Tr.	5370	22.8	287.7
10K-Tr.	10830	87.8	801.1

評価に用いた回路は，文献10)に掲載されているものである．CVSLは，Cascade voltage switch logic（ページ312）である．FFは，CMOS positive-level-sensitive D latch（ページ19）である．SHIFTは，Array shifter using transmission gates（ページ561）である．

STANTは，回路CVSLで2個のトランジスタの信号の流れる方向を決定することができなかった．回路FFと回路SHIFTは，すべてのトランジスタの信号の流れる方向を決定できた．一方，文献11)のstructure-based処理で決定できなかったトランジスタは，CVSLが11個，FFが2個，SHIFTが16個であった．しかし，その後に続くrule-based処理ではすべてのトランジスタの信号の流れる方向を決定することができている．

表4 は，STANTと市販ツールのCPU時間の比較を示す．CPU時間は，SPICEネットリストを読みこんで，トランジスタの信号の流れる方向を決定して，クリティカルパスを探索して，その結果をレポートするのにかかった時間である．STANTは，市販ツールよりも1桁高速である．このことは，提案する信号の流れる方向を決定する方法が効果的であることを示

表5 CPU時間の比較（STANTと文献11)）
Table 5 Comparison of CPU time with STANT and Ref. 11).

Circuit	#Tr.	STANT (sec)	reference 11) (sec)
CVSL	29	0.13	0.050 + 0.050
FF	10	0.09	0.033 + 0.033
SHIFT	56	0.29	0.033 + 0.067

している．信号の流れる方向をより多く決定できているために，タイミング解析全体としての処理が高速である．

表5は，STANTと文献11)とのCPU時間の比較を示す．STANTの値は，SPICEネットリストを読みこんで，トランジスタの信号の流れる方向を決定して，クリティカルパスを探索して，その結果をレポートするのにかかった時間である．一方，文献11)の値は，トランジスタの信号の流れる方向を決定するだけにかかった時間である．この結果より，STANTの信号の流れる方向の決定方法は，高速であることが分かる．また，STANTは，トランジスタの方向決定手法のみならず，タイミング解析ツールとして見た場合にも高速であることが分かる．

回路の設計者が，STANTがレポートしたクリティカルパスと，市販ツールがレポートしたクリティカルパスの両方をチェックした．両ツールがレポートした最長10本のパスがフォールスパスであるかどうかをチェックした．STANTは，どのデータでも1本もフォールスパスはレポートしていなかった．このことは，STANTのフォールスパスを排除するアルゴリズムが効果的であることを示す．また，信号の流れる方向を決定する手法が正しく機能していることも示す．しかしながら，市販のツールがレポートしたパスには，数本のフォールスパスが存在した．データadder8では，最長10本のパスのうち，1，2，5，6番めのパスがフォールスパスであった．データI2ADDでは，1，4，5，10番目のパスがフォールスパスであった．

6. まとめ

本論文では，トランジスタレベルのスタテックタイミング解析における信号の流る方向を決定する新しい効果的な手法を述べた．

トランジスタレベルのフォールスパスの問題は，信号の流れる方向を決定する問題と密に関係している．提案した方法は，製品となる実際の回路で，パストランジスタの信号の流れる方向を正確に決定するのにおおいに貢献している．

計算機実験の結果によれば，ベンチマークしたほとんどのデータで100%近くのトランジスタの信号の流れる方向を決定した．また，市販ツールでは，人手の介入なしには解析できなかったバレルシフターも解析することができた．信号の流れる方向の決定率が高くて正確であることは，フォールスパスを削減するのみならず，処理時間の短縮にもつながる．

この優れた能力は，人手での指定が非常に難しいレイアウトから抽出されたデータの場合には，非常に効果的である．

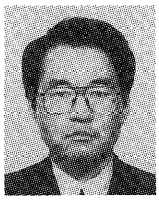
STANTは，高速なタイミング解析ツールである．計算機実験によれば，STANTは市販ツールより1桁高速である．このことは，数百トランジスタからなる今日のマイクロプロセッサーを実用的な時間でタイミング解析することができることを意味している．

参考文献

1) McWilliams, T.M.: Verification of Timing Constraints on Large Digital MOS VLSI, *Proc. 17th Design Automation Conference*, pp.139–147 (1980).
2) Hitchcock, R.B., Smith, G.L. and Cheng, D.D.: Timing Analysis of computer hardware, *IBM J. Res. Develop.*, Vol.26, No.1, pp.100–105 (Jan. 1982).
3) Ousterhout, J.K.: Crystal: A Timing Analyzer for nMOS VLSI Circuits, *Proc. 3rd CalTech VLSI Conference*, pp.57–69 (1983).
4) Jouppi, N.P.: TV: An nMOS Timing Analyzer, *Proc. 3rd Caltech VLSI Conference*, pp.71–85, (1983).
5) Ousterhout, J.K.: A Switch-Level Timing Verifier for Digital MOS VLSI, *IEEE Trans. Computer-Aided Design*, Vol.CAD-4, 3, pp.336–349 (July 1985).
6) Szymanski, T.G.: LEADOUT: A Static Timing Analyzer for MOS Circuits, *Proc. ICCAD-86*, pp.130–133 (1986).
7) Jouppi, N.P.: Derivation of Signal Flow Direction in MOS VLSI, *IEEE Trans. Computer-Aided Design*, Vol.CAD-6, 3, pp.480–490 (May 1987).
8) Cherry, J.J.: Pearl: A CMOS Timing Analyzer, *Proc. 25th Design Automation Conference*, pp.148–153 (1988).
9) Pan, J., et al.: Timing Verification on a 1.2M-Device Full-Custom CMOS Design, *Proc. 28th Design Automation Conference*, pp.551–554 (1991).
10) West, N. and Eshraghian, K.: *Principles of CMOS VLSI design, a system perspective*, Second Edition, Addison Wesley (1993).

11) Lee, K.J., et al.: An Integrated System for Assigning Signal Flow Direction to CMOS Transistors, *IEEE Trans. Computer-Aided Design*, Vol.14, No.12, pp.1445–1458 (1995).

(平成10年9月17日受付)
(平成11年2月 8 日採録)



栗林 元隆（正会員）

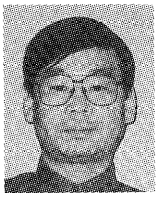
昭和34年生．昭和60年京都大学大学院工学研究科数理工学専攻修士課程修了．同年（株）東芝入社．主に，半導体集積回路のレイアウトのCAD，タイミング検証のCADの研究・開発に従事．IEEE，ACM，電子情報通信学会各会員．

山田 正昭（正会員）

昭和56年京都大学大学院工学研究科電気工学第二専攻修士課程修了．同年,(株) 東芝入社．半導体集積回路の自動設計，特にレイアウト検証・自動レイアウトの研究開発に従事．昭和61年から63年にかけてイリノイ大学客員研究員．電子情報通信学会会員．

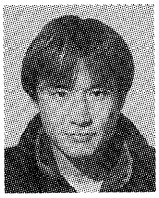
竹内 秀輝

昭和39年生．昭和63年鹿児島大学工学部電子工学科卒業．同年（株）東芝入社．主に，スタティックRAMの回路設計，半導体集積回路のタイミング検証CAD，モジュールジェネレータの研究・開発に従事．



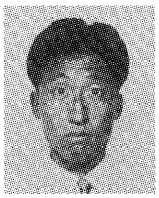
辻本 順一

昭和28年5月生．昭和57年大阪大学大学院理学研究科数学専攻博士課程修了．昭和58年湯川奨学生．昭和58年（株）東芝入社．現在は東芝デバイス技術研究所で高速SRAM設計グループのマネージャーを務める．今までに，LSI設計，検証用CADの研究，開発およびEEPROM，SRAM設計開発に従事．



外岡 康宏

1991年東京大学農学部農業工学科卒業．現在,(株) 数理システムに勤務．半導体集積回路検証のためのタイミング解析やネットリスト圧縮等のCAD，および，ネットワーク関連の各種システム開発に従事．



黒岩健太郎

1984年大阪大学工学部造船学科卒業．現在,(株) 数理システムに勤務．半導体集積回路検証のためのタイミング解析やネットリスト圧縮等のCAD，および，数理計画法による金融システム開発に従事．