ベクトル計算機による高速論理シミュレーション†

石浦菜岐佐†† 安浦寛人†† 矢島脩三††

論理シミュレーションの高速化の新しいアプローチとして，ベクトル計算機を利用する手法を提案する．ゲート・レベルの零遅延シミュレーションについて，それぞれ組合せ回路，同期式順序回路を対象とする2通りの計算手法を開発した．いずれの手法も遅延を考慮する必要がないことから，コンパイル法，パラレル法を基本としているが，処理をベクトル化するために新たな計算手法を導入している．組合せ回路のシミュレーションに関しては，Tアルゴリズムを基本とし，多数のパタンを時間方向にまとめて処理するベクトル・パラレル法（VP 法）によりベクトル化を図った．順序回路についてはSアルゴリズムの立場をとり，同種類のゲートをまとめて評価するゲート・グルーピング法（GG 法）によりベクトル化を実現している．これらのシミュレーション手法をベクトル計算機 FACOM VP-100 (266 MFLOPS) 上に実現した結果，いずれもほぼ100%ベクトル化され，大規模なシミュレーションにおいて高い性能を示すことが確認された．その最大性能はおよそ4.0 G gate/sec（組合せ回路シミュレータ），0.9 G gate/sec（順序回路シミュレータ）であり，論理シミュレーション専用に開発されたハードウェアに匹敵するものである．

1. はじめに

論文シミュレーションはディジタル回路の設計検証を目的として，その論理的動作を計算機上で模擬するものであり，計算機をはじめ種々の論理システムのCAD において不可欠なものとなっている．近年の半導体技術の進歩に伴う回路の大規模化は，シミュレーションの計算時間を急速に増大させており，これに対処すべく様々な研究がなされている．これらの研究のアプローチは，

1）機能レベル・シミュレーションなど回路モデルの工夫によるもの，

2）コンカレント・シミュレーション[1]，Tアルゴリズム[2]など，アルゴリズム的改良によるもの，

3）Krohn のベクトル・コーディング[3]（ただし，通常の汎用計算機におけるものである）にみられるコーディング上の工夫，

4）IBM の YSE[4]，日本電気の HAL[5]，ZYCAD の LE[6]など，専用のハードウェアを開発するもの，

に大別されよう．これに対し本論文では第5の新しいアプローチとしてベクトル計算機を利用する手法を提案する．

ベクトル計算機は演算パイプライン方式のスーパ・コンピュータであり，最近500 MFLOPS 以上の高性能な機種が相次いで発表されている[7),8)]．ベクトル計算機は大規模数値計算を主目的として開発されてきたものであるが，浮動小数点演算以外の処理能力も強化されており，非数値計算にも広範な応用が考えられる．

ベクトル計算機の性能は，いかなる計算に対しても発揮されるというわけではない．最大性能を引き出すためには，演算の大部分がベクトル処理され，しかも各ベクトルが十分な長さを持っていることが不可欠である．このため，問題によってはコーディング技法に留まらず，根本的なアルゴリズムにまで立ち返って，計算のベクトル化を考えることが必要となる．

本論文では以上の点をふまえ，ベクトル計算機に適した二つのシミュレーション手法を提案する．両者とも零遅延シミュレーションを対象としているが，一方は組合せ回路専用であり，他方は同期式順序回路も扱えるものである．遅延を考慮する必要がないため，いずれもコンパイル法，パラレル法[9]を基本としているが，処理をベクトル化するために新たな計算手法を導入している．すなわち，組合せ回路のシミュレーションに関しては，時間優先評価（Tアルゴリズム）を基本とし，多数のパタンを時間方向にまとめて処理するベクトル・パラレル法（VP 法）によりベクトル化を図った．また，順序回路については空間優先評価（Sアルゴリズム）の立場をとり，同種類のゲートをまとめて評価するゲート・グルーピング法（GG 法）によりベクトル化を実現している．

これらのシミュレーション手法をベクトル計算機 FACOM VP-100 (266 MFLOPS) 上に実現した結果，いずれもほぼ100% ベクトル化され，大規模なシミュレーションにおいて高い性能を示すことが確認された．その最大性能はおよそ4.0 G gate/sec（組合せ

† High-Speed Logic Simulation Using a Vector Processor by NAGISA ISHIURA, HIROTO YASUURA and SHUZO YAJIMA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室

回路シミュレータ），0.9 G gate/sec（順序回路シミュレータ）であり，前述のハードウェア・シミュレータに匹敵するものである．

本論文では，2章でベクトル計算機と論理シミュレーションに関する基本的事項について述べた後，3，4章でそれぞれ組合せ回路，順序回路のシミュレーションに関して述べる．

2. 準　　備

2.1 ベクトル計算機

ベクトル計算機は演算パイプライン方式のスーパ・コンピュータであり，流体，気象，エネルギー開発などの分野における大規模な科学技術計算の要求を満たすべく開発されたものである．計算の高速化は，数値計算に頻繁に現れるベクトル・データ（配列データ）に対する同一の繰り返し処理を，演算パイプラインによって高速に実行することで実現されている．

本報告で使用したベクトル計算機 FACOM VP-100 は広範な応用を指向して，ベクトル化の範囲を広げるために様々な処理機能を備えている．データ型に関しては，浮動小数点ばかりでなく，整数，論理データもベクトル演算の対象となっており，32 ビット 1 ワードのデータに対する加減乗算，ビットごとの論理演算がパイプラインによって高速に処理される．主記憶は最大 128 MB（現システムは 32 MB，うちユーザ領域 22 MB）であり，ベクトル・アクセスは，1) 連続アクセス，2) 等間隔アクセス，3) リスト・ベクトル・アクセスの 3 種が可能である．リスト・ベクトル・アクセスとは，

```
      DO 10 I=1,N
10    A(I)=B(L(I))
```

のように，整数配列を介して間接アクセスを行うものである．アクセスの速度はメモリのバンク競合の関係で，一般に 1), 2), 3) の順となる．

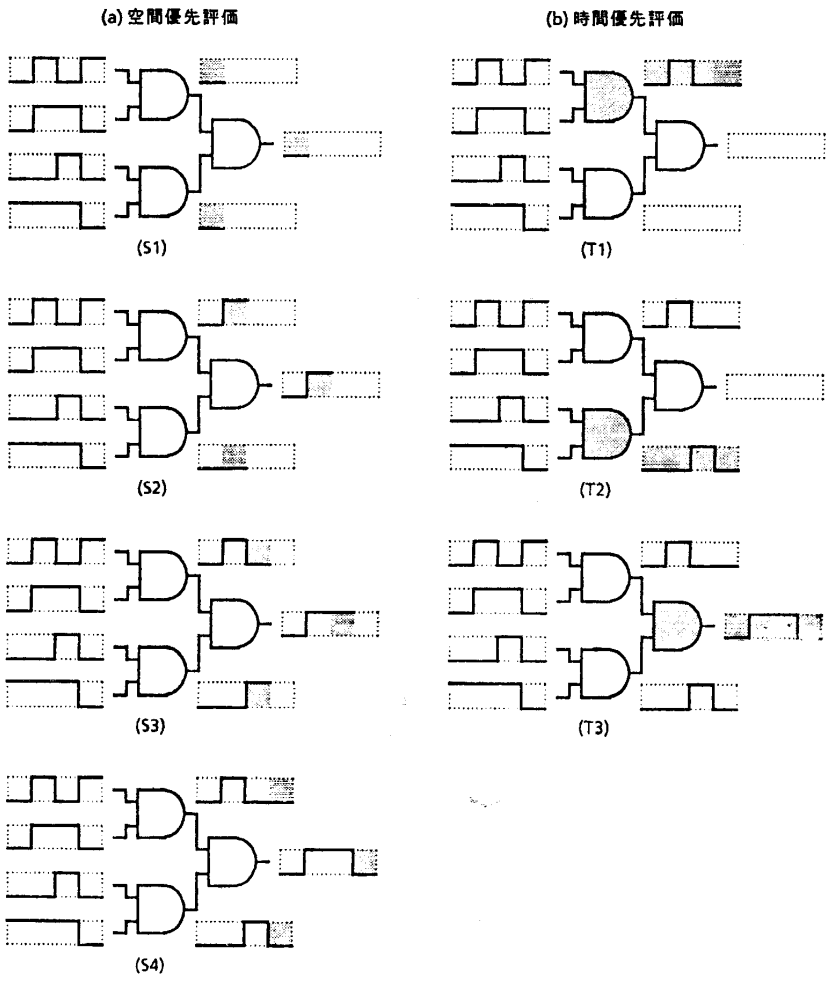

図 1　空間優先評価と時間優先評価
Fig. 1　Space first evaluation and time first evaluation.

2.2 論理シミュレーション

論理シミュレーションは，シミュレーションの単位によりゲート・レベルと機能レベルに分類され，さらに遅延を扱うものと扱わないものがあるが，ここではゲート・レベルの遅延を考慮しないものについて議論する．

シミュレーションは通常図 1 (a)のように 1 時刻（1 パタン）ごとに計算を進めていく空間優先の評価がよく用いられるが，ゲートごとに可能な信号値計算を行う時間優先の評価法も考えられる(図 1 (b)参照)．ループのない組合せ回路に対しては，時間方向に計算をまとめて行うことが効率上有利であり，タイミング・シミュレーションに関しては空間優先に比べて数倍の高速化が達成できる[2]．

シミュレーションの実行制御方式としてはコンパイル法とイベント法がある[9]．イベント法は信号値の変化（イベント）に注目して計算を行うために，一般にコンパイル法に比べゲートの評価回数は

少ないが，

1）零遅延シミュレーションにおいてはイベント発生率が高く，イベント法でも多くのゲート評価が必要になる．

2）イベント法におけるイベント・スケジューリングの処理はオーバヘッドが大きく，しかもベクトル化が困難である．

3）これに対し，コンパイル法の単純な処理過程はベクトル化しやすいと考えられる．

等の点を考慮し，今回コンパイル法を採用した．

コンパイル法におけるゲートの処理順序の決定法としては，従来からレベル・ソートが広く用いられているが，ここではより一般的なデータ・フロー・ソート（以下 DF ソート）を考える．すべての入力に入力パタンが準備されたゲートを評価可能と呼ぶことにすると，ゲートの評価順序は次のように決定される．

アルゴリズム DF ソート

Step 0 まず，すべての入力が外部入力となっているゲートが評価可能である．これらのゲートを集合 S の要素とする．

Step 1 S が空になるまで Step 2～3 を繰り返す．

Step 2 S からゲートを一つ取り出して評価する．

Step 3 Step 2 の結果新たに評価可能になるゲートがあれば S に加える． □

図 2 の組合せ回路において A-D-B-C-E-F-G 及び B-C-E-G-A-D-F は，DF ソートにより得られるがレベル・ソートでは得られない処理順序である．正しいシミュレーション結果を与えるゲート評価順序は，DF ソートですべて得ることができ，レベル・ソートより処理順序の選択に自由度が大きいと言える．本論文のシミュレーション手法ではこの自由度を，シミュレータの速度性能，処理容量を向上させることに活用している．

3. 組合せ回路の高速論理シミュレーション

3.1 ベクトル・パラレル法

組合せ回路のシミュレーションにおけるコンパイル法のベクトル化手法としては，Tアルゴリズムに基づくベクトル・パラレル法を提案する．これは，信号値パタンの評価を時間方向にまとめて行うものであり，信号線のとるパタン系列をベクトルとして，ゲートの評価をこのベクトルに関するベクトル論理演算で高速に行う手法である．さらに，このベクトル論理演算がビットワイズに定義されていれば通常のパラレル法[9]を併用することができる．すなわち，長さ p の入力パタンを $\lceil p/w \rceil$ ワード（w はワード長，$\lceil x \rceil$ は x より小さくない最大の整数を表す）のベクトルとして扱い，処理の高速化と使用記憶量の削減を図る．

3.2 記憶域の制御とゲートの処理順序

ベクトル計算機の性能を引き出すためには，処理ベクトル長を長くとり，多くのパタンをまとめて処理することが必要となる．このため，大規模な回路のシミュレーションでは，1パタンごとに回路の評価を行う従来の計算法に比べ，信号値パタンを記憶するベクトル（以下パタン・ベクトル）のための領域が無視できなくなる．そこで，ここではこの使用記憶量を，

1）シミュレーションの各時点で必要なパタン・ベクトルのみを保持し，

2）パタン・ベクトルの領域を再利用する，

ことにより削減している．

図 3 はシミュレーションの各時点で記憶すべきパタンを，二つのゲート処理順序について示したものであ

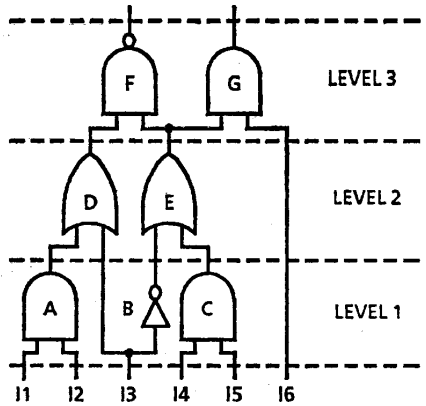

図 2 ゲートの評価順序
Fig. 2 Ordering of gate evaluation.

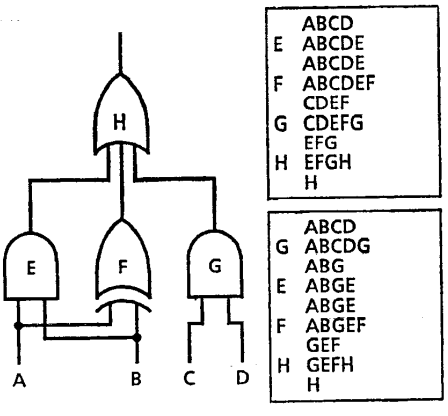

図 3 記憶すべきパタン・ベクトル
Fig. 3 Pattern vectors to be memorized.

表 1 *Mv* の比較
Table 1 Comparison of *Mv*.

ゲート数(段数)	レベルソート	DF ソート	比率(%)
85 (13)	19	14	73.6
168 (25)	31	22	70.9
270 (22)	38	26	68.4
346 (48)	55	38	69.0
1,240 (48)	142	63	44.3
5,296 (76)	542	119	22.0

る．この図より記憶すべきパタンの最大数（以下 *Mv* という）は同一の回路においてもゲートの処理順序によって異なることがわかる．ここではさらにこの *Mv* をできるだけ小さくする処理順序を求めることを考えるが，その際，多くのゲート処理順序の中から *Mv* が小さなものを選択できるという点で，レベル・ソートよりも自由度の高い DF ソートのほうが有利になる．必要記憶量を最小にする解を求めることも可能ではあるが，この最適化問題を完全に解くことは難しい（レジスタ充足問題[10]の変形で NP 困難）と考えられるため，ここでは，再利用が可能となるパタン・ベクトルの数に注目するヒューリスティックを導入している[11]．

表 1 はいくつかの回路について単純なレベル・ソートと上記ヒューリスティック付き DF ソートによる *Mv* の比較を行ったものであるが，

1）いずれの場合も *Mv* はゲート数に比べて小さい．
2）DF ソートはレベル・ソートに比べて *Mv* を 7 割から 2 割と大幅に削減している．

ことがわかる．

3.3 シミュレータの実現と性能評価

我々はこのシミュレーション手法を京都大学大型計算機センターの FACOM VP-100 上に 2 値論理シミュレータとして実現し，その性能評価を行った．**図 4** はシステム構成を示している．トランスレータは SHDL (Structured Hardware Design Language)[12] による回路記述を読み込み，回路の時間優先評価を行う FORTRAN プログラムを生成する．**図 5** は図 2 の回路に対する FORTRAN プログラムである．前節で述べたゲート評価の順序はこの段階で決定されており，作業配列（図 5 の BUFF）におけるパタン・ベクトルの格納位置も FORTRAN プログラム中に指定されている．ゲートの評価は IAND, IOR などの組み込み関数を用いてコーディングされる．このプログラムは VP-100 の機械コードにコンパイルされ，このコードの実行によりシミュレーションが行われる．

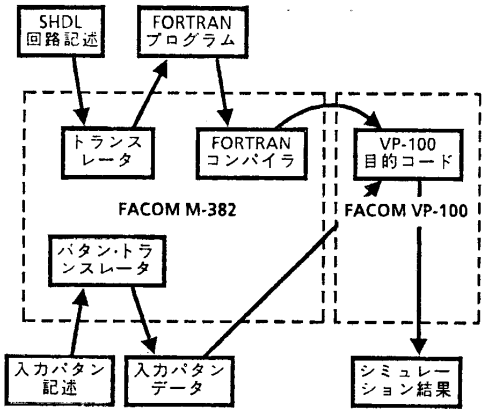

図 4 システム構成
Fig. 4 System configuration.

```
000001      READ(4,400)((BUFF(J,I),J = 1,LEN),I = 1,6)
000002      DO 10 J = 1,LEN
000003      BUFF(J,7) = IAND(BUFF(J,1),BUFF(J,2))
000004      BUFF(J,1) = IAND(BUFF(J,4),BUFF(J,5))
000005      BUFF(J,2) = IOR(BUFF(J,7),BUFF(J,3))
000006      BUFF(J,4) = NOT(BUFF(J,3))
000007      BUFF(J,3) = IOR(BUFF(J,4),BUFF(J,1))
000008      BUFF(J,1) = NOT(IAND(BUFF(J,2),BUFF(J,3)))
000009   10 BUFF(J,2) = IAND(BUFF(J,3),BUFF(J,6))

      1行目  外部入力I1~I6の読み込み
      3行目  ゲートAの評価
      4行目  ゲートCの評価
      5行目  ゲートDの評価
      6行目  ゲートBの評価
      7行目  ゲートEの評価
      8行目  ゲートFの評価
      9行目  ゲートGの評価
```

図 5 FORTRAN プログラム
Fig. 5 FORTRAN program.

FORTRAN プログラム中のパタン・ベクトルに対する論理演算は，ベクトル論理演算命令（32 ビット 1 ワードに対してビットワイズ）で実行されるため，100%ベクトル化される．また，パタン・ベクトルへのアクセスは連続アクセスであり，最も高速に実行される．シミュレーション速度は主としてベクトル長（パタン長/32）に依存し，ゲート数には依存しない．**表 2** は実験の結果得られたシミュレーション速度である．ベクトル長を長くとれば 4.0 G gate/sec に及ぶ処理速度が得られるが，100 程度のベクトル長でも十分な性能が得られる．スカラ計算機（FACOM M-382：VP-100 のスカラ演算能力と同等と発表されている）との比は 20 倍にも達する．

現在のシステムでは一つのゲートを一つの FORTRAN 文で表現しているが，一つの文で複数ゲート

表 2 処理速度（組合せ回路）
Table 2 Simulation speed (for combinational circuit).

ベクトル長	速度 (G gate/sec)
8	0.62
32	2.07
128	3.52
512	3.80
2,048	3.87
8,192	3.87
20,000	4.00

の論理を表現することが考えられる．この結果，演算の数そのものに変化はないが，処理がベクトル・レジスタ上で行われて主記憶への格納が省略されるため，処理速度が向上する．十数ゲートの回路に対して簡単な実験を行った結果では，20%の性能向上が測定された．

シミュレーションに必要な記憶には，1) プログラムの領域と，2) パタン・ベクトルの領域がある．前節の議論のとおり，2) に必要となる記憶量はベクトル長と Mv の積に比例する．Mv は回路のゲート数だけでなく構造にも依存するため，単純に見積ることはできないが，ユーザ領域22 MB の制限の下で1 Mゲート程度の回路は扱えるものと考えられる．

4. 同期式順序回路の高速論理シミュレーション

4.1 ゲート・グルーピング法

同期式順序回路は一般に**図6**に示すように，組合せ回路とレジスタ（記憶）により構成される．シミュレーションはこの組合せ回路部分を1クロックごとに評価することにより進められるが，レジスタからの帰還のために，3章で用いた時間方向に信号値をまとめて計算する方法は適用することができない．そこで本論文では空間方向に計算をまとめて行う方法を考える．すなわち，組合せ回路部分の評価の際に，同じ関数を持つゲートをひとまとめにし（以下グループ化と言う），これらを一度に処理することにより計算のベクトル化を行う．無論，信号伝播の順序を無視してはならず，グループ化は**図7**の例のように，シミュレーション結果に矛盾が生じないように決定されなければならない．レジスタは単純な遅延素子と見なし，クロック周期の最後（Mealy 型の場合は最初でもかまわない）に処理する．

図 6 同期式順序回路
Fig. 6 Synchronous sequential circuit.

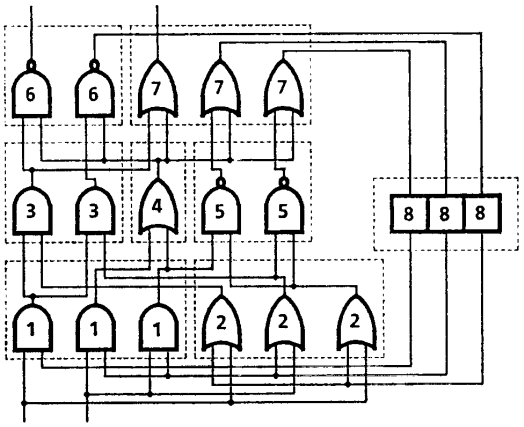

図 7 グループ化の例（1）
Fig. 7 Example of grouping (1).

4.2 DF ソートによるゲート・グルーピング

ベクトル演算の効率化という観点から，グループに含まれるゲートの数の平均は大きいことが望ましい．これは言い換えれば，回路が与えられたときに，それをできるだけ少ないグループに分ける問題となる．図7のグループ化はレベル・ソートに基づいて決定されている．すなわち，同一の関数と同一のレベル番号を持つゲートが一つのグループにまとめられている．ここでは，さらに大きな平均グループ・サイズを得るために，DF ソートに基づいてグループ化を決定する方法を提案する．その手続きは以下のとおりである．

アルゴリズム DF ソートによるゲート・グルーピング

Step 0 全入力が外部入力かレジスタの出力につながっているゲートを集合 S の要素とする．

Step 1 Step 2〜3 を S が空になるまで繰り返す．

Step 2 一つの関数を選び，S からその関数を持つゲートをすべて取り出しグループとする．

Step 3 Step 2 で選んだゲートを評価した結果，新たに評価可能となるゲートができれば，それを S に加える． □

図7と同じ回路にこの手法を適用して得られるグループ化の例を**図8**に示す．この手続きの結果得られるグループの平均サイズは，Step 2 における関数の選択基準に依存する．最適解を求めることはやはり難しい問題と考えられるため，我々はいくつかのヒューリスティックを開発している．**表3**は7,484ゲート

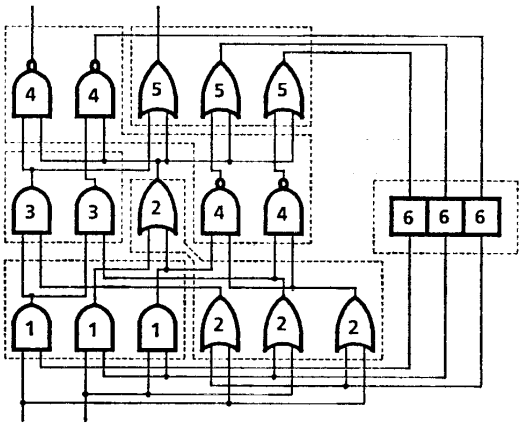

図 8 グループ化の例 (2)
Fig. 8 Example of grouping (2).

表 3 グループ化アルゴリズムの比較
Table 3 Comparison of grouping algorithms.

アルゴリズム	グループ数	平均サイズ	処理時間(秒)
レベル・ソート	355	21.1	0.75
本稿 Heuristic 1	304	24.6	1.13
本稿 Heuristic 2	235	31.9	2.10

(130段, 4ゲート種) の演算回路 (積和を計算する回路) に対して, グループ化を行った結果である. 表中, Heuristic 1 はレベル・ソートより良い解を, Heuristic 2 はそれよりもさらに良い解を保証するものである.

4.3 シミュレータの実現

順序回路シミュレータに関しては, コード生成方式でなく表駆動方式により実現した. ゲートの関数評価は次のようなプログラムにより計算される (2入力 AND ゲートの例).

```
      DO 20 J=NX+1, NX+NG(I)
   20 L(J)=IAND(L(IX(J,1)), L(IX(J,2)))
      NX=NX+NG(I)
```

ただし, NG(I) はグループに含まれるゲートの数, L(g) はゲート g の信号値, IX(g, i) はゲート g の i 番目の入力に信号値を供給するゲートの番号を表す. 入力信号値のフェッチはリスト・ベクトル・アクセスを用いてベクトル化される. 同一グループ内のゲートには連続したゲート番号を割り付けることにより, 配列Lへのストアは連続アクセスで処理される. IAND, IOR 等の組み込み関数はビット・ワイズの論理演算を実現し, 2値論理シミュレーションでは32ケースをパラレルにシミュレーションすることができる. なお, メモリー素子は多入力・多出力で記憶を持つゲートと見なし, 通常の論理ゲートと同等に扱っている.

表 4 処理速度 (順序回路)
Table 4 Simulation speed (for sequential circuits).

ゲート数	グループサイズ	速度 (M gate/sec)
40	8	46.0
80	16	88.9
160	32	164.0
320	64	287.2
640	128	450.8
1,280	256	650.8
2,560	512	796.3
5,120	1,024	902.9

4.4 性能評価

単純な繰り返し構造を持つ回路 (AND, OR 等1ベクトル論理演算命令で処理できるゲートで構成) について, グループ・サイズと処理速度の関係を測定する実験を行った. 結果を**表4**に示す. 32ケース・パラレル・シミュレーションによる最大性能は 0.9 G gate/sec であり, M-382 によるものと比較すると 10 倍程度高速である. 組合せ回路シミュレータと比べると, リスト・ベクトル・アクセスを使用しているため数分の一の性能となっている. また, 表3で示した実際の複雑な回路の場合にも, グループ・サイズ 31.9 において 122.6 M gate/sec の速度が測定された. これは表4の結果よりも遅いが, 3入力ゲート, 否定ゲート等の使用により, 1ゲートあたり約 1.8 命令を要していることを考慮すれば, 処理効率はむしろ良くなっている (3.3 節で述べた最適化と同様の効果が表れている).

シミュレーションの速度効率は, 前節でも述べたように, 1グループあたりのゲート数が大きいほど高くなる. 1グループあたりのゲート数は, 1) 回路全体のゲート数が大きく, 2) 回路の論理段数が小さいほど大きい. 近年のディジタル回路は, その大規模化はもちろんのこと, 故障検査の容易化やマシン・サイクルの短縮化のため, スキャン・パス・レジスタを多用して回路の論理段数を小さくする傾向にあり, 本手法が有効になると考えられる. また, 素子の種類は少ないほどグループのサイズは大きくなることから, 多種類のゲートを含む回路の場合には適当に展開を行って (例えば2入力のゲートばかりに展開する等) ゲートの種類を減らすことが有効になると考えられる.

また, シミュレーションに必要な記憶量は1ゲートあたり $4(i+1)$ バイト (ただし i はゲートの入力数) である. したがって, 例えば2入力ゲートなら必要記

億量は12バイトであり，22Mバイトのユーザ領域でおよそ1.8Mゲートまでのシミュレーションが可能である．

4.5 回路のモデル

同期式順序回路は図6のようなものばかりではなく，組合せ回路部分からレジスタのクロックやリセット／プリセット入力に信号が供給される場合がある．本手法は単なる遅延素子として扱われているレジスタに簡単な論理を追加することにより，このような実際的な回路にも適用することが可能である[11]．さらに，レジスタを1単位時刻の遅延と見なすことにより，単位遅延シミュレーションも可能であり，非同期回路も扱うことができる．

5. むすび

組合せ回路および同期式順序回路のベクトル計算機向きシミュレーション手法を提案した．処理のベクトル化は，組合せ回路ではTアルゴリズムに基づくベクトル・パラレル法（VP法）により，順序回路ではSアルゴリズムに基づくゲート・グルーピング法（GG法）により達成している．本論文では2値論理のみ実現したが，信号値の符号化により多値論理を扱うことも可能である（詳細は別紙にゆずる）．シミュレーションの処理効率や記憶効率を向上させるためには，前処理のアルゴリズムも重要であり，レベル・ソートの一般化であるDFソートとそれに基づくいくつかの前処理アルゴリズムを提案した．本論文のシミュレーション手法はほぼ100%のベクトル化率を達成しており，大規模なシミュレーションにおいて非常に高い処理効率を示す．**表5**は本論文の手法に基づくシミュレータと現存するハードウェア・シミュレータの性能を比較したものであるが，本シミュレータはハードウェア・シミュレータに匹敵する性能を持っているといえる．

表5 論理シミュレータの性能比較
Table 5 Performance of logic simulators.

シミュレータ	Drive algorithm	Modeling level	Active gate evaluation /sec
Mega LOGICIAN	Event	RTL/gate	0.1
Realfast	Event	RTL/gate	0.5
Tegas Accelerator	Event	RTL/gate	1.0
Zycad Logic Evaluator	Event	gate	60.0
NEC HAL	Event	PLA	360.0
IBM YSE	Compiler	gate	960.0
VP(Combinational)	Compiler	gate	600.0
VP(Sequential)	Compiler	gate	100.0

ハードウェア・シミュレータのデータは文献6)による（処理速度は2入力1出力ゲート相当に換算）

ベクトル計算機は数値計算のみならず，ここに示したVLSIのCADのような組合せ問題に対しても大きな潜在能力を持っていると考えられる．今後，種々の組合せ問題に対するベクトル計算機向きのアルゴリズムの開発が盛んになると思われるが，ベクトル計算機のアーキテクチャをこれらの組合せ問題向きに改良することも重要であると考える．

謝辞 ご討論いただいた本学平石裕実講師，高木直史氏をはじめ，矢島研究室の諸氏に感謝します．また，本報告の組合せ回路シミュレータのコーディングに尽力された河田哲郎氏（現在，インテル・ジャパン勤務）に深謝します．なお，本研究は一部文部省科学研究費による．

参考文献

1) Ulrich, E.: A Design Verification Methodology Based on Concurrent Simulation and Clock Suppression, *Proc. 20th DAC.*, pp. 709-712 (1983).
2) Ishiura, N., Yasuura, H. and Yajima, S.: Time First Evaluation Algorithm for High-Speed Logic Simulation, *Proc. ICCAD-84*, pp. 197-199 (1984).
3) Krohn, H. E.: Vector Coding Techniques for High Speed Digital Simulation, *Proc. 18th DAC.*, pp. 525-528 (1981).
4) Denneau, M., Kronstadt, E. and Pfister, G.: Design and Implementation of a Software Simulation Engine, *CAD*, Vol. 15, No. 3, pp. 123-130 (1983).
5) Sasaki, T., Koike, N., Ohmori, K. and Tomita, K.: HAL: A Block Level Hardware Logic Simulator, *Proc. 20th DAC.*, pp. 150-156 (1983).
6) Blank, T.: A Survey of Hardware Accelerators Used in Computer-Aided Design, *IEEE Design & Test of Computers*, Vol. 1, No. 3, pp. 21-39 (1984).
7) 速さを競うスーパーコンピュータ，日経エレクトロニクス，1983年4月11日号（No. 314），pp. 105-184 (1983).
8) スーパーコンピュータSXシステム，日経エレクトロニクス，1984年11月19日号（No. 356），pp. 237-271 (1984).
9) 村井真一：ゲート・レベル論理シミュレーション，情報処理，Vol. 22, No. 8, pp. 762-769 (1981).
10) Garey, M. R. and Johnson, D. S.: *Computers and Intractability—A Guide to Theory of NP-*

Completeness, W. H. Freeman and Company (1979).
11) 石浦菜岐佐，安浦寛人，河田哲郎，矢島脩三：ベクトル計算機による高速論理シミュレーション，情報処理学会設計自動化研究会資料，25-2, pp. 1-10 (1985).
12) 安浦寛人，蚊野 浩，大井 康，木村晋二，石浦菜岐佐，矢島脩三：入力制約監視機能を持つ会話型シミュレーション・システム ISS，情報処理学会論文誌，Vol. 25, No. 2, pp. 285-292 (1984).

(昭和60年11月 1 日受付)
(昭和61年 2 月20日採録)

石浦菜岐佐（正会員）

昭和36年生．昭和59年京都大学工学部情報工学科卒業．昭和61年同大学院修士課程修了．現在，同大学院博士課程に在学中．論理回路のCAD/DA の研究に従事．電子通信学会会員．

安浦 寛人（正会員）

昭和28年生．昭和51年京都大学工学部情報工学科卒業．昭和53年同大学院修士課程修了．昭和55年3月より京都大学情報工学教室助手．工学博士．非同期回路，論理関数の複雑さ，VLSI 向きハードウェアアルゴリズム，論理設計の CAD 等の研究に従事．昭和57年電子通信学会学術奨励賞受賞．IEEE，電子通信学会，EATCS，LA シンポジウム各会員．

矢島 脩三（正会員）

昭和8年生．昭和31年京都大学工学部電気工学科卒業．同大学院博士課程修了・工学博士．昭和36年より京大・工学部に勤務，昭和46年情報工学科教授，昭和35年京大第一号計算機 KDC-1 を設計稼動，以来，計算機，論理設計，オートマトン等の研究教育に従事．著書は「電子計算機の機能と構造」(岩波，57年) 等．本学会元常務理事，元会誌編集委員（地方），元 JIP 編集委員．電子通信学会元評議員およびオートマトンと言語研専元委員長，North-Holland 出版 IPL 編集委員，IEEE Senior Member.