

ベクトル計算機による 高速論理シミュレーション

石浦 業岐佐 安浦 寛人 河田 哲郎 矢島 脩三

京都大学工学部

1. はじめに

論理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。論理シミュレーションに要する計算時間は大雑把に言って、対象回路の規模と回路への入力となるテストバタンの長さに比例する。近年の半導体技術の進歩に伴う回路の大規模化は、必然的にその検証に必要となるテスト入力の長さをも増大させ、シミュレーションの計算時間は急速に増大している。回路設計の高信頼化、設計期間の短縮という要求から、論理シミュレーションの高速化は一つの重要な課題となっており、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)汎用計算機の上でソフトウェア・シミュレーションの性能を向上させるものと、2)シミュレーションに適した専用のハードウェアを開発するものに大分される。前者の例としては、コンカレント・シミュレーション[1]、時間優先評価法(Tアルゴリズム)[2]のようなアルゴリズム的改良の他、(通常の汎用計算機上で)ベクトル的コーディングを行うことによりシミュレーション速度を向上させるといったコーディング上の技法[3]がある。後者の例としては、IBMのYSE[4]、日本電気のHAL[5]、商用機のZYCAD-LE[6]等が挙げられる。シミュレーションの速度に関して言えば、ハードウェア・シミュレータはソフトウェアの数千倍から数万倍の性能を持っている。しかし、シミュレータの柔軟性、モデルの拡張性等の観点から、ソフトウェアによるシミュレーションは今後も重要な位置を占めるものと考えられる。

ベクトル計算機は演算パイプライン方式のスーパー・コンピュータであり、最近各社から500MFLOPS (Mega Floating-point Operation Per Second)以上の高性能な機種が発表されている[7][8]。ベクトル計算機は大規模数値計算を指向するものであり、主として浮動小数点演算の処理に重点をおいて開発されてきたが、ベクトル処理の範囲を広げる為にその他の機能も強化されており、数値計算以外にも広範な応用が考えられる。しかし、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。超大型汎用機の数十倍に及ぶその最大性能を引き出す為には、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。この為、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為いずれもコンパイル法[9]を採用しているが、前者は時間優先評価(Tアルゴリズム)[2]を基本とし、後者は空間優先評価(Sアルゴリズム)[2]の立場をとり、根本的に異なる計算手法を採用している。

組合せ回路のシミュレーションに関しては、コンパイル法と時間優先評価法、パラレル法[9]を併用する手法を開発した。計算のベクトル化は、各ゲートにおいて多くのバタンをまとめて評価することにより実現されている。順序回路のシミュレータに関しては、時間優先の評価が困難な為、回路内の同一種類のゲートをまとめて評価するグループ化手法の導入によりベクトル化を図った。

上記シミュレーション手法をベクトル計算機FACOM VP-100(266MFLOPS)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4.0G gate/sec (組合せ回路シミュレータ)、0.7G gate/sec (順序回路シミュレータ)であり、ソフトウェアながら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章では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ー・コンピュータであり、流体、気象、エネルギー

ギー開発等の分野に於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。その最大性能は数百MFLOPSであり、超大型汎用機の数十倍にも達する。

しかしながらこの性能は、プログラム中のほとんどすべての演算がベクトル命令で実行された場合に得られるものであり、ベクトル命令の比率(ベクトル化率)の低下とともに処理速度は急激に低下する。例えばベクトル化率が50%の場合、ベクトル命令がいかに高速に実行されようとも、全体の処理速度はスカラ計算機の2倍を超えることはできず、ベクトル化率90%としてもせいぜい数倍の性能改善しかなされない。またベクトル命令は立ち上がりのオーバーヘッドが大きく、ベクトル長が十分でないとパイプライン処理の効果が得られないことになる。このようにベクトル計算機の計算力を十分に引き出す為には、プログラムが極度にベクトル化されており、かつベクトル長が十分長いことが不可欠となる。さらには命令の種類、メモリ・アクセスの種類、同時に実行できる演算の数等の様々な要素も実効性能に大きく影響する。これらの点を考慮し、場合によってはコーディング技法のみならず、根本的なアルゴリズムまでベクトル計算向きに考え直すことが必要となる。

表1は京都大学大型計算機センターで稼働中のFACOM VP-100の諸元である。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為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128MB(現システムは32MB、うちユーザ領域22MB)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。

表1 FACOM VP-100の諸元

項目	VP-100
最大性能	266MFLOPS
マシン・サイクル	7.5ns
命令数	スカラ195、ベクトル82
レジスタ	汎用16個、浮動小数点8個、 ベクトル32Kバイト、 マスク512バイト
演算パイプライン	加減論理演算、乗算、除算 計6本
主記憶	32Mバイト(最大128M)

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、さらにタイミング(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ボタン)ごとに計算を進めていくこと(空間優先評価)が多いが、ゲート毎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なされることが報告されている[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という2通りの手法が知られている[9]。コンパイル法は入力ボタン毎に回路中の全ゲートを信号値の伝播順に評価するものであり、1)回路の実現する論理を計算する(ホスト計算機の)目的コードを生成し、これを入力ボタン毎に実行するコード生成方式と、2)表を用いて計算を制御する表駆動方式がある。これに対しイベント法は信号値の変化(イベント)に注目し、各入力ボタンに対し、入力にイベントが発生したゲートだけを追跡して評価する方法であり、コンパイル法に比べてゲートの評価回数は少なく済む。しかしながら我々は、

- 1) 遅延を考慮しないシミュレーションではイベントの発生率が高く、イベント法においても全回路の1/4~1/2のゲートを評価しなければならないと考えられる。
- 2) イベント処理の操作は複雑であり、オーバーヘッドが大きい。

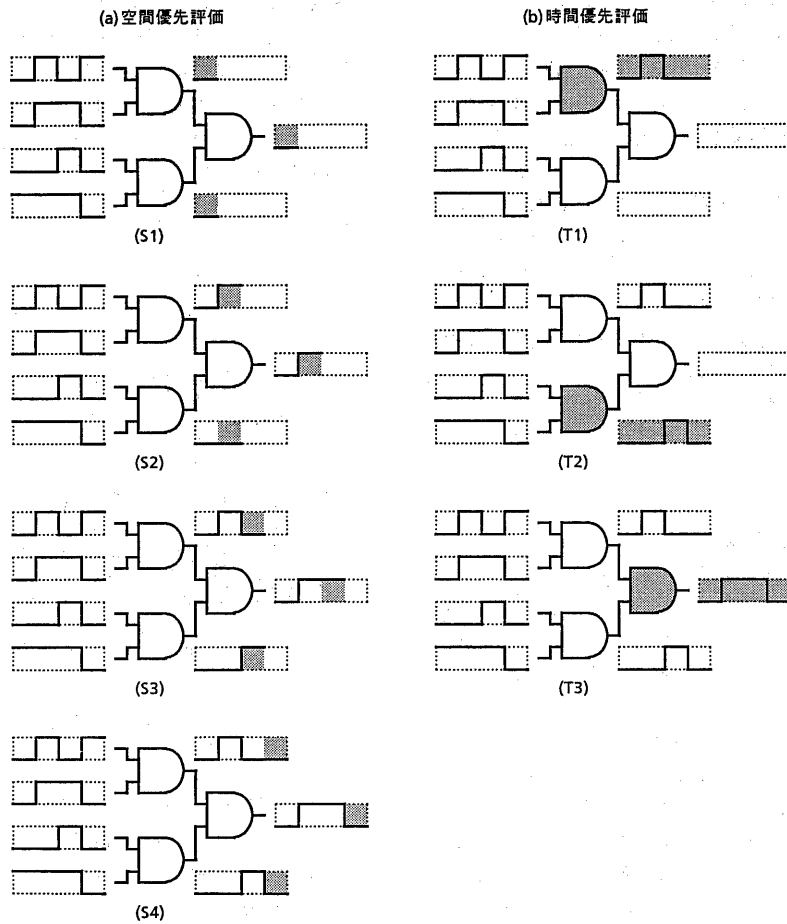


図1 空間優先評価と時間優先評価

3) これに比べコンパイル法の単純な計算過程はベクトル処理しやすいと考えられる。

等の理由から、今回コンパイル法を採用することが有利と判断した。

コンパイル法におけるゲートの処理順序の制御法としては従来からレベル・ソート法が知られている。これは図2に示すように信号値の伝播順にゲートにレベル番号をつけ、A-B-C-D-E-F-Gのようにレベル番号の若いものから順に処理してゆくものである。レベル・ソート法は実現が簡単であり、広く実用されているが、より一般的にはデータ・フロー的な順序制御法が考えられる。すべての入力に入力ボ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制御される。

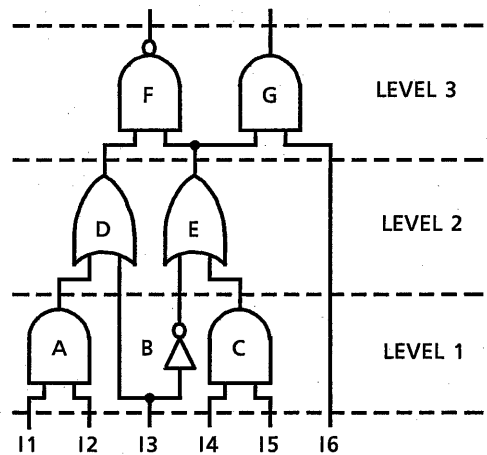


図2 レベル・ソート

Step 0 まず、すべての入力が入力となっているゲートが処理可能である。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 3~4を繰り返す。

Step 2 Sからゲートを1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においてA-D-B-C-E-F-G及びB-C-E-G-A-D-Fはレベル・ソート法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処理順序は、この順序制御法ですべて得ることができ、レベル・ソート法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 シミュレーション・アルゴリズム

組合せ回路のシミュレーションに関してはベクトル計算機向き計算手法として、コンパイル法により時間優先に評価を行う方式を採る。即ち、ゲートの出力評価を多数のボタンについてまとめて行うことにより計算のベクトル化を行う。更に、ビットワイズの論理演算を利用するパラレル法を併用することにより、高速化と記憶量の削減を図る。長さpの入力ボタンはp/wワード(wはワード長)のベクトルとして扱われ、ゲートの評価はこれらのベクトルに対するベクトル論理演算より高速に実行される。

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為には、処理ベクトル長を長くとり、多くのボタンをまとめて処理することが必要となる。この為、大規模な回路のシミュレーションでは、1ボタン毎に回路の評価を行う従来の計算法に比べ、信号線の信号値を記憶する領域の大きさが無視できなくなる。信号値ボタンを記憶するベクトル(以下ボタン・ベクトル)の数について考えると、シミュレーションの全実行過程を通じて全信号線と同じだけのボタン・ベクトルを保持しておく必要はなく、シミュレーションの各時点で必要となる最小限のものだけを記憶しておけばよい。即ち、シミュレーションの始めには外部入力の分だけを記憶しておく。ゲート評価の結果新しいボタン・ベクトルが生成されるが、一方では以後のゲート評価にはもはや必要でないボタン・ベ

クトルもできる。このボタン・ベクトルの占める記憶領域は解放でき、他の新しいボタン・ベクトルを格納するために再利用することができる。このような記憶制御方式を用いると、シミュレーションに必要な記憶は、各時点で保持すべきボタン・ベクトルの最大数Mvの分で済み、全信号線の数に比べると大幅に減少する。

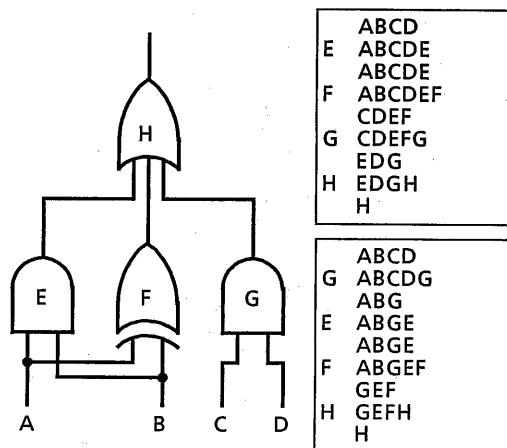


図3 Mv数の処理順序依存性

ところで、このMvは、同一の回路においてもゲートの処理順序によって変化する。例えば図3においてE-F-G-Hの順では6必要なのがG-E-F-Hの順では5で済む。限られた記憶の中でできるだけベクトル長を長くにとってシミュレーションの効率を上げるという観点からも、Mvを小さくするようにゲートの処理順序を決定することが重要になってくる。この際、多くの処理順序の中からMvができるだけ小さなものを選択できるという点で、拘束の強いレベル・ソート法よりもデータフロー制御法のほうが、有利になる。データフロー制御法は、正しいシミュレーション結果を保証するすべての処理順序を与えることができるため、必要記憶量を最小にする解を求めることも可能である。しかし、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)と考えられる為、ここでは解放が可能なボタン・ベクトルの数に注目するヒューリスティックを導入する。

解放可能なボタン・ベクトル数Ndは、シミュレーションの各ゲート評価の段階において、評価可能なゲートに対して定義され、そのゲートを評価した後に解放できるボタン・ベクトルの数を表す。例えば、図4においてゲートBを評価すると、Bの

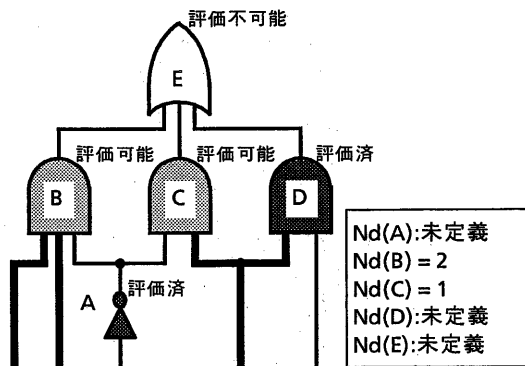


図4 Nd数

入力線のうち太線で示す2本の入力に対するパタン・ベクトルは解放できる。よってゲートBのNdは2となっている。シミュレーションにおいて保持すべきパタン・ベクトルの数は、始めは外部入力の数に等しく、最後には0になる(但し、シミュレーション結果としてトレースすべき信号線のパタン・ベクトルは、求まり次第外部記憶に書き出されるものとしている)。従って、図5からも明らかなように、Ndの大きなゲートを優先して処理するほうが、必要なベクトルの数を小さくできる。

パタン・ベクトル数

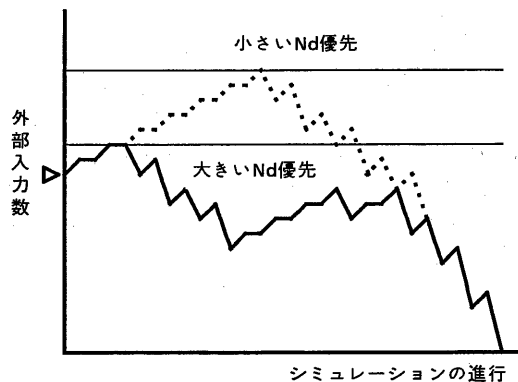


図5 パタン・ベクトル数の変化

表2はいくつかの回路について順序決定アルゴリズムによるMv数の比較を行ったものであるが、本ヒューリスティックによるデータフロー制御法は単純なレベル・ソート法に比べてMv数を7割から2割と大幅に削減している。Ndの計算はレベル・ソートのレベル番号づけと同様、回路データの表検索によ

表2 Mv数の比較

ゲート数 (段数)	レベル ソート	本論文の計 算法	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
1240 (48)	142	63	44.3
5296 (76)	542	119	22.0

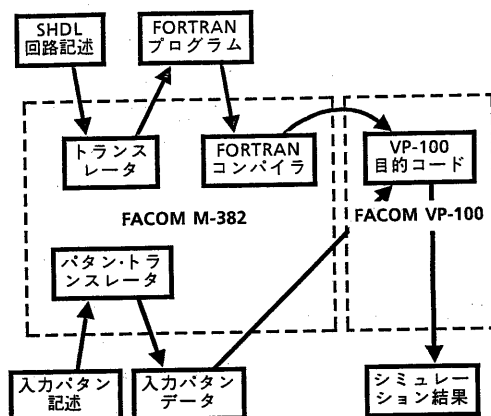


図6 システム構成

り簡単に計算でき、本シミュレーション手法のゲート処理順序制御法として有効であると考えられる。尚、このゲート処理順序は、回路の構造のみから静的に決定でき、前処理時に計算することによってシミュレーションの実行時のオーバーヘッドを避けることができる。

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FACOM VP-100上を実現し、その性能評価を行った。図6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)[11]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7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7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、

```

000001  READ(4,400)((BUFF(J,I),I=1,LEN),J=1,LEN)
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1~16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

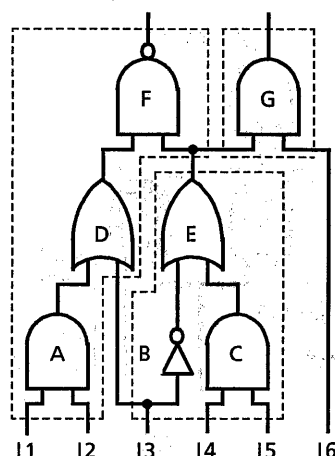
図7 FORTRANプログラム

IEOR、NOT等FORTRANの組み込み関数を用いてコーディングされる。このFORTRANプログラムはFORTRAN77/VPコンパイラによってVP-100のマシン・コードにコンパイルされ、VP-100上でこれを実行することによりシミュレーションが行われる。

FORTRANプログラム中のボタン・ベクトルに対する論理演算は、ベクトル論理演算命令で実行される為、このプログラムは100%ベクトル化される。また、ボタン・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(ボタン長/32)に依存する。表3は実験結果得られたシミュレーション速度である。ベクトル長を長くすれば4.0G 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。スカラー計算機(FACOM M-382: VP-100のスカラー演算能力と同等と発表されている)との比は40倍から60倍にも達する。

表3 処理速度

ベクトル長	速度(Ggate/sec)
5	0.55
25	2.18
100	3.67
500	3.86
1000	3.91
5000	3.89
20000	4.00



```

E = IOR(NOT(I3), IAND(I4, I5))
F = NOT(IAND(E, IOR(I3, IAND(I1, I2))))
G = IAND(E, I6)

```

図8 高速化処理

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)ボタン・ベクトルの領域がある。前節の議論の通り、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為、単純に見積ることはできないが、ユーザ領域22MBの制限の下で1Mゲート程度の回路は扱えるものと考えられる。

3.4 高速化手法

現在のシステムでは1つのゲートを1つのFORTRAN文で表現しているが、図8に示すように適当にゲートを1まとめて1つのFORTRAN文で複数ゲート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為、

- 1) ベクトルのロード/ストアの回数が減少しシミュレーションが更に高速になる。
- 2) 中間結果として保持すべきボタン・ベクトルの数が減少する。

等、処理速度・容量の両面においてメリットがもたらされる。処理速度に関しては、簡単な実験の結果では、更に20%程度の高速化が達成されている。

4. 同期式順序回路の高速論理シミュレーション

4.1 シミュレーション・アルゴリズム

同期式順序回路は一般に図9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

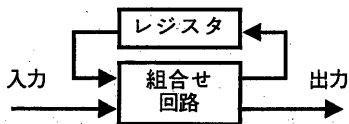


図9 同期式順序回路

レーションはこの組合せ回路部分を1クロック毎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為に、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。勿論多数のケースを同時にシミュレーションすることも考えられるが、1)ベクトル処理の効果が十分に発揮されるには数千、数万というケース数が必要である、2)大きなメモリーをもつ同期回路をシミュレーションする場合、各ケースに対応するメモリーの内容を記憶しなければならない、等の問題があり非現実的である。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。即ち、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10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、クロック周期の最初又は最後(順序回路の形式による)に処理する。

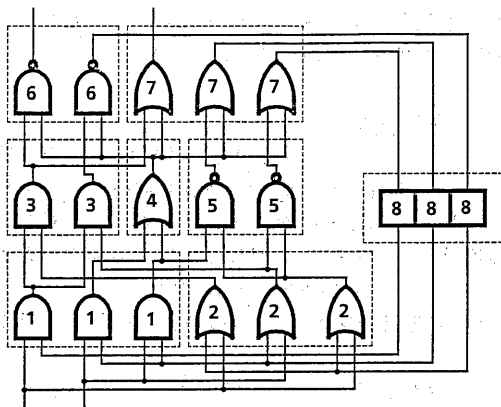


図10 グループ化の例(1)

4.2 グループ化のアルゴリズム

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10のグループ化はレベル・ソート

法に基いて決定されている。即ち、同一の関数と同一のレベル番号を持つゲートが1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・サイズを得る為に、データフロー制御法を提案する。データフロー制御法に基づくグループ化は以下の手続きにより決定される。

- Step 0** 全入力から外部入力から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** 1つの関数を選び、Sからその関数を持つゲートを全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。

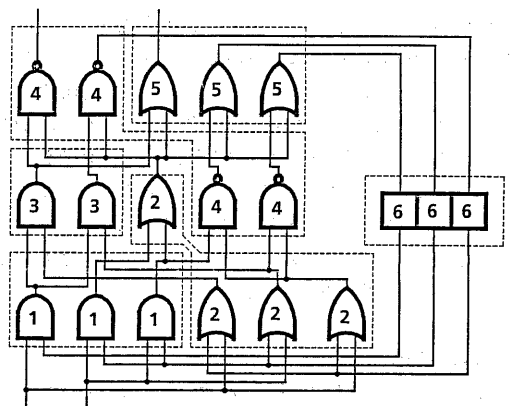


図11 グループ化の例(2)

図10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11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が、レベル・ソート法よりも良い解を保証するヒューリスティックが存在する。表4は7000ゲート(125段、4ゲート種)の回路に対して、いくつかのアルゴリズムでグループ化を行った結果である。

表4 グループ化アルゴリズムの比較

アルゴリズム	グループ数	平均サイズ
レベル・ソート	279	25.0
本稿Heuristic 1	219	31.9
本稿Heuristic 2	188	37.1

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)
```

但し、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。このループでは配列Lに関して見かけ上再帰的な定義・参照が行われているが、実際に再帰性は無く、このことをコンパイラに指示することによりベクトル化が可能となる。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。尚、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。

4.4 性能評価

シミュレータの速度性能は、およそ0.7G gate/sec(32ケース・パラレル、平均グループ・サイズ1000の時)であり、M-382と比較すると20倍以上高速である。また、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為数分の一の性能となっている。

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程高くなる。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い、
- 2)回路の論理段数が小さい、

程大きくなる。近年のデジタル回路は、その大規模化はもちろんのこと、故障検査の容易化、マシン・サイクルの短縮化の為、スキャン・バス・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。

シミュレーションに必要な記憶には、1)ゲートに関するもの、2)グループに関するものがある。2)に関しては1グループ20バイト程度であるが、大規模な回路ではグループ数はゲート数に比べて小さくなる為問題にはならない。1)に関しては、a)ゲートの出力信号値、b)ゲートの入力信号値へのインデックスが必要である。a)は1ゲートあたり4Bである。b)に関しては、先に述べたIXのような固定配列を用いると最大ファンイン数に対する記憶を用意しなければならない。しかし実際には、ほとんど同じ処理効率で、ゲート毎に必要な最小限の記憶で済ませる方法をインプリメントしており、入力線の総数に対して4iバイトで十分である。

従って、例えば2入力ゲートなら必要記憶量は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9のようなものばかりではなく、実際にはクロック・ディストリビューション・ロジックや、レジスタのリセット/プリセット等が用いられる。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、このような実的な回路にも適用することが可能である(図12参照)。さらに、レジスタを1単位時刻の遅延と見なして各論理ゲートに付加することにより、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。

4.6 実行制御方式

シミュレーションの制御方式及び前処理の計算量について検討する目的で、組合せ回路シミュレータに関してはFORTRANコードを生成する方式を、順序回路シミュレータに関しては表で駆動する方式を採った。コードを生成する方式は、

- 1) コード生成のレベルで3.4節のような最適化ができる。
- 2) 表を解釈実行するオーバー・ベッドが不要である。ベクトル長が十分長い場合には余り問題にならないが、短い場合には相当の差が現れる。

等の実行時の長所がある反面、

- 1) FORTRANコードの生成は、文字列処理を必要とし、時間がかかる。
- 2) FORTRANコードのコパイルが必要である。VP用のコンパイラはベクトル化や特別な最適化を行う為、特に時間がかかる。

等、前処理の計算時間に問題点を含んでいる。以上から、小規模なシミュレーションにはコード生成方式が、大規模なシミュレーションには表駆動方式が適していると考えられる。

5. まとめ

組合せ回路及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について述べた。これらの計算手法は、それぞれ時間方向、空間方向に計算をまとめることによってほぼ100%のベクトル化率を実現しており、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。また、シミュレーションの処理効率や記憶効率を向上させる為には、前処理のアルゴリズムも重要であり、レベルソート法の一般化であるデータ・フロー制御法に基づくいくつかのアルゴリズムを示した。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、我々の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。

ベクトル計算機は数値計算のみならず、VLSIのCADのような組合せ問題に対しても大きな潜在力を持っていると考えられる。我々は更に、故障シミュレーションやイベント法によるタイミング・シミュレーションに関しても、ベクトル計算機に適した高速な計算手法を開発中である。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られる。

謝辞

御討論頂いた本学平石裕実博士、高木直史助手をはじめ、矢島研究室の諸氏に感謝します。尚、本研究は一部文部省科学研究費による。

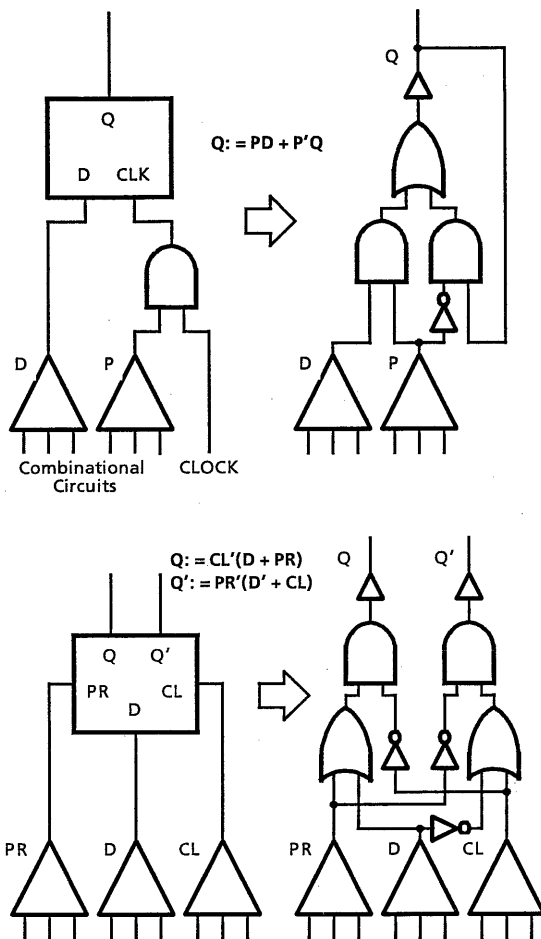


図12 複雑なレジスタ

表5 性能の比較[6]
(ゲートは2入力1出力に換算してある)

Simulator	Drive Algorithm	Modelling Level	Gate Capacity	Gate Evaluation /sec	Active gate Evaluation /sec
Tegas Accelerator	Event	RTL/gate	2.5M	8M	1M
Zycad Logic Evaluator	Event	gate	3.8M		60M
NEC HAL	Event	PLA	3M	1.2G	360M
IBM YSE	Compiler	gate	4M	6.4G	960M
VP-100(組合せ回路)	Compiler	gate	0.1M	4.0G	600M
VP-100(順序回路)	Compiler	gate	1.8M	0.7G	100M

参考文献

- [1] Ernst Ulrich: "A Design Verification Methodology Based on Concurrent Simulation and Clock Suppression", Proc. 20th DAC., pp.709~712, (1983).
- [2] N. Ishiura, H. Yasuura and S. Yajima: "Time First Evaluation Algorithm for High-Speed Logic Simulation", Proc. ICCAD-84, (1984).
- [3] H. E. Krohn: "Vector Coding Techniques for High Speed digital Simulation", Proc. 18th DAC., pp.525~528, (1981).
- [4] M. Denneau, E. Kronstadt and G. Pfister: "Design and Implementatiion of a Software Simulation Engine, CAD Vol.15, No.3, pp.123~130, (1983).
- [5] T. Sasaki, N. Koike, K. Ohmori and K. Tomita: "HAL: A Block Level Hardware Logic Simulator", Proc. 20th DAC., pp.150~156, (1983).
- [6] Tom Blank: "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 4-11 (No.314), pp.105~184.
- [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 11-19 (No.356), pp.237~271.
- [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] M. R. Garey and D. S. Johnson: "Computers and Intractability - A Guide to the Theory of NP-Completeness", W.H. Freeman and Company, (1979).
- [11] 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).