数 値 解 析　16－3
（1986. 3. 20）

ベクトル計算機による回路行列の高速求解法

下郡　慎太郎　　鹿毛　哲郎
（株）富士通研究所

1．序　論

回路解析プログラムは，信頼性の高い集積回路を効率良く設計するために広く使用されている。従来の回路解析プログラムは，一般に汎用計算機で実行されてきたが，回路の大規模化に伴い，膨大な計算時間を必要とするようになってきた。この問題を解決するために，ベクトル演算機能をそなえたスーパーコンピュータを用いた回路解析プログラムの研究が行われている（【2】～【8】）。

回路解析の解析時間の内訳をみてみると，(1)非線形素子の値を計算するためのモデル関数の計算，(2)非線形依存電源の線形化・リアクタンス素子の離散化等に伴う行列要素の更新，および(3)連立一次方程式を解くための行列求解に大きく分れる。回路解析で扱う行列は，ランダムスパース行列（非零要素がまばらで不規則な行列）であり，しかも，同じ形の行列を数百～数千回繰り返し解く。そのため，これら3段階が実行時間の大半を占めている。

筆者等が開発を行ってきた汎用回路解析プログラムＦＩＮＡＰ－ＶＰでは，スーパーコンピュータにおいて，従来困難とされていたモデル計算・行列更新の効率的なベクトル化に成功し，スカラ演算に比べ１０～３０倍の高速化を達成した【11】。この結果，相対的に行列求解の比重が高くなって，解析時間の６～９割以上を占めるようになり，行列求解の高速化をはかる必要がでてきた。

回路解析では，回路行列のスパース性に着目して，行列求解を高速化しようとする試みが幾つかなされてきた。比較的古いものとしては，行列求解手順を機械コードとして生成するコード生成法がある。これは，スカラ演算としては非常に効率的な方法である。ランダムスパース行列の演算では，要素を探すためのアドレス計算，ループを回る時のオーバーヘッド等の時間が大きく，数値的な演算そのものの比率は小さい。コード生成法では，その行列を解くための最適な機械語を生成するため，これらの余分な時間がなくなり，7～10倍の高速化を実現する。

また，ベクトルプロセッサを用いての高速化の研究が活発に行われている（【2】～【8】）。パイプライン型のベクトルプロセッサで高速化するためには，プログラムのベクトル化を促進するとともに，ベクトル長（並列計算可能な要素の数）を長くする必要がある。

ところが，行列求解に用いられるＬＵ分解法やガウスの消去法は，スパース行列に対しては，非常に並列度の低いアルゴリズムで，そのままではほとんど高速化できない。このため，行列を幾つかのブロックに

分けてそのブロック内を並列化する，部分的に稠密行列として扱う【７】，部分回路記述を利用する【２】等の手法が用いられた。

最近，ＬＵ分解の最大限の並列性を引き出すアルゴリズムとして，演算を参照順序に従ってレベル付けする方法が示された【８】。ところが，このベクトル化手法でも，スパース行列に対して，行列求解の全体を効率良くベクトル化しているとは言えない。すなわち，直接解法で最も効率の良いとされるＬＵ分解法において，稠密行列では行列次元ｎの３乗に比例して演算が増えるＬＵ分解の過程のみが注目される。しかし，回路行列のようなスパース行列では，通常は無視される前進代入・後退代入の過程もかなりの比重を占めている。このため，ベクトル計算機によって，行列求解を高速化するためには，これらの過程も含めた行列求解全体をベクトル化しなければならない。本稿では，ＬＵ分解法の全段階をベクトル化する手法を示し，実行テストを行った結果を示す。さらに，そのための前処理時間・使用メモリ量についても言及する。

２．ＬＵ分解法

２．１　用語の定義

連立一次方程式の直接求解方法としては，ガウスの消去法が良く知られている。これは，前進消去により上三角行列をつくり，それを後退代入により解くというものである。また，別の方法として，ＬＵ分解法と呼ばれる方法がある。これは，行列を一旦，下三角行列（Ｌ）と上三角行列（Ｕ）の積の形に分解し，その後２回の代入で解を求めるものである。

ガウスの消去法とＬＵ分解法とは，実は，数学的には全く等価な方法であり，演算量も等しいが，計算機で行う場合には，計算順序および演算以外の処理に違いが出てくるため，明確に区別しておく必要がある。また，ＬＵ分解法において，行列を分解する過程には幾つかのバリエーションが考えられ，これも数値計算では微妙な差がでてくるにもかかわらず，明確に区別されていない場合がある。

本稿では，混乱を避けるために，行列を一旦下三角行列と上三角行列の積の形に分解した後，代入により解を求める方法を総称してＬＵ分解法と呼ぶ。したがって，ガウス消去法はＬＵ分解法とは区別される。また，行列を分解する過程で，内積計算により分解後の要素を一度で計算する方法をクラウト法，ガウス消去法と同様の計算で少しづつ計算する方法をガウス法と呼ぶことにする。

回路解析における行列求解には，一般にＬＵ分解法が用いられる。ＬＵ分解法は，

① ＬＵ分解（クラウト法，ガウス法等による）
② 前進代入
③ 後退代入

の３段階に大きく分かれる。ＬＵ分解は，行列を下三角行列と上三角行列の積に分解する過程，前進代入は下三角行列による代入，後退代入は上三角行列による代入である。これらの名称も文献によって違ってい

る場合があるが，本稿ではこのように呼ぶことにする。先にあげたガウスの消去法では，ＬＵ分解と前進代入とが前進消去として同時に行われる。

２．２　ＭＶＡ（Maximal Vectorization Algorithm）

ＬＵ分解の計算は，

$$a = a - b \times c, \quad a = a / b$$

の２種の基本演算のみで計算できる。この特徴に注目して，文献【８】では，両演算を要素の参照関係に基づいてレベル付けし，同じレベルの演算を並列実行することにより，ベクトル長の十分に長いＬＵ分解が可能となることを示している。以下に，このアルゴリズムの概要を述べる。

解くべき行列を　Ａｘ＝ｂ　とする。行列Ａの要素と一対一に対応した，各要素の演算レベルを格納するための配列を用意し，その初期値を０とする。

次に，ＬＵ分解の過程をシミュレートする。このとき，ａ＝ａ－ｂ×ｃタイプの演算についてはａ，ｂ，ｃのレベルのうち最大のものに１を足した値をａの新しいレベルとする。すなわち，

```
LEVEL(a) = MAX(LEVEL(a), LEVEL(b),
                         LEVEL(c)) +1
```

とする。ｂ，ｃのレベルは変えない。同時に，ａ，ｂ，ｃの要素ポインタの値をａの新しいレベル（これがこの演算のレベルとなる）にしたがってテーブルに格納する。ａ＝ａ／ｂのタイプについても同様に，ａとｂのレベルの大きい方に１を足した値をａの新しいレベルとし，ａ，ｂの要素ポインタをテーブルに格納する。以上の処理が，前処理として一度だけ行われる。

実際に行列求解を行うには，レベルの小さな演算から順に，格納された要素ポインタを使って計算していけば良い。このとき，レベルが同じ演算は互いに参照関係がないため，すべて並列実行できる。

ＬＵ分解法による稠密行列の求解では，ＬＵ分解と前進代入・後退代入の実行時間は，それぞれ行列次元の３乗，２乗に比例するため，行列次元の大きな行列では，ＬＵ分解が大部分の時間を占めるようになる。

しかしながら，回路解析では，非常にスパースな行列（行列次元にかかわらず，１行当り３～６要素程度）を扱うために，稠密行列のようにはならない。筆者等が開発を行ってきた回路解析プログラムＦＩＮＡＰでは，スカラでの高速解析を行うために，コード生成法による行列求解を行っている。コード生成法で生成されるコードにはループがないため，コードの量とその実行時間はほぼ比例すると考えても良い。実際の回路で，ＬＵ分解・前進代入・後退代入のコード生成量を調べたところ，約７：３：２の割合であり，これは回路規模に関係しなかった。したがって，行列求解全体を高速化するためには，ＬＵ分解のみではなく，前進代入・後退代入も同程度に速くしなければならない。

3．拡張レベル付けベクトル化ＬＵ分解法

3．1　ＬＵ分解法における計算

ＬＵ分解（クラウト分解による場合）・前進代入・後退代入の各段階では，実際には図3.1のような計算を行う。概念的に式で書けば，

$$\left.\begin{aligned} l &= a - \Sigma\, l\, u \\ u &= (a - \Sigma\, l\, u) \,/\, l \end{aligned}\right\}(\text{ＬＵ分解})$$

$$y = (b - \Sigma\, l\, y) \,/\, l \quad (\text{前進代入})$$

$$x = b - \Sigma\, u\, x \quad (\text{後退代入})$$

というような計算である。これらの式では，l，u，x，yを全て別に書いているが，実際には，メモリ節約のため，行列ＡとＬ，Ｕ，ベクトルｂとｘ，ｙとは同一メモリに割り当てられる。

ここで，上式を良く見ると，ＬＵ分解のみならず，前進代入・後退代入もａ＝ａ－ｂ×ｃ，ａ＝ａ／ｂという基本演算のみで計算できることがわかる。このため，ＬＵ分解・前進代入・後退代入の各過程にまたがってレベル付けすることが可能である。このアルゴリズムを拡張レベル付けベクトル化ＬＵ分解法と呼ぶことにする。

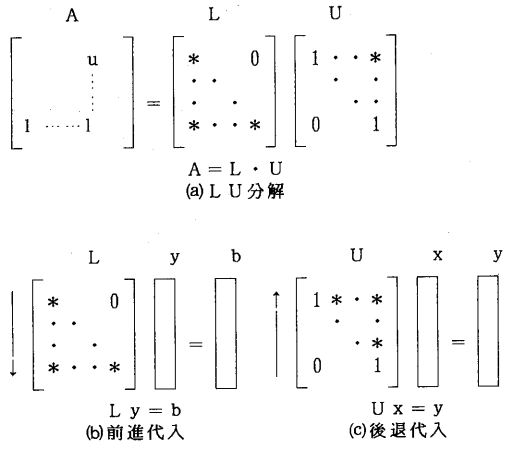


図3.1　ＬＵ分解法における計算

ＬＵ分解はＭＶＡと同様にレベル付けする。計算方法はクラウト法でもガウス法でも効率は同じである。コード生成法を行う場合には，レジスタ上での内積累和のできるクラウト法のほうが有利であるが，ＭＶＡでは，内積累和ができないからである。前進代入・後退代入は，ＬＵ分解に引き続きレベル付けする。要素ポインタもＬＵ分解の要素ポインタと同じ配列に格納する。これにより，実際の実行時には，ＬＵ分解・前進代入・後退代入といった過程を全く意識することなく同等に取り扱うことができる。

3．2　前進代入過程のレベル付け

前進代入のレベル付けをする時には，Ａの要素のレベルを記憶するための配列と別にｂの要素のレベルを記憶するための配列が必要となる。

さらに注意すべきことは，後退代入に追い越されないように，行列ｂの要素のレベルを更新しなければならない，ということである。このため，一般的には第一行，第二行，…　と行ごとに計算するが，本アルゴリズムでは，図3.2のように第一列，第二列，…　と列ごとに計算する。各要素を更新するときに，更新される要素のレベルを変えるのはもちろんであるが，第ｉ列目の前進代入が終ったときに，ｉ列での最大のレベルをｂｉの新しいレベルとする。

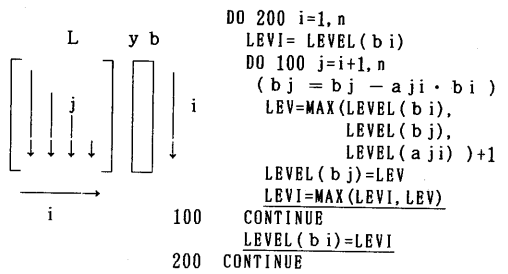
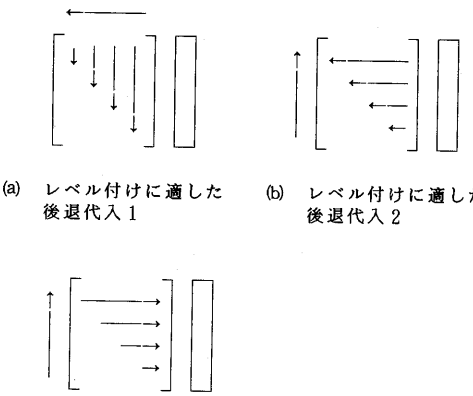


図3.2　レベル付けに適した前進代入



(a)　レベル付けに適した後退代入1

(b)　レベル付けに適した後退代入2

(c)　レベル付けに適さない後退代入

図3.4　各種方法による後退代入

3．3　後退代入過程のレベル付け

後退代入時にも，前進代入と同様にレベル付けの効率を良くするために，図3.4(a)または，(b)のような方法をとったほうが良い。両方法とも等価であるが，行列の要素を示すためのポインタは，一般に行番号・列番号が増える方向に持たせることが多く，この場合(b)の方法ではポインタを追う回数が増えるため，(a)のほうが効率的である。なお，(c)のようにすると（これは一般的に行われる後退代入の方法である），レベル数が増大し，効率が悪くなる。

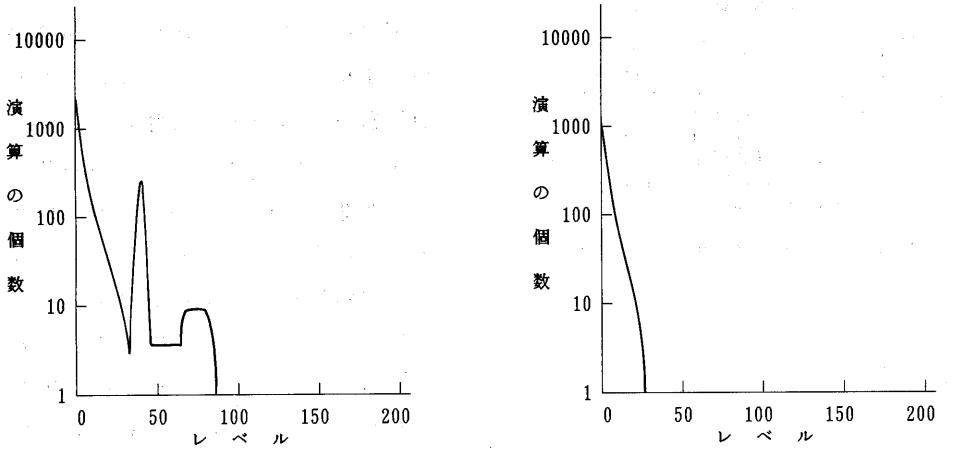
4．演算並列度の分析

実際の回路行列における，演算の分布を図4.1に示す。これは，16ビットｎＭＯＳ加算器（ＡＤＤ１６と呼ぶ）の例である。

ａ＝ａ－ｂ×ｃタイプについてみると，ピークが３か所ある。最初のピークはレベル 1～10くらいのもので，ベクトル長が長く，回路にもよるが数千程度になる。これはＬＵ分解と前進代入によるものである。第２のピークと第３のピークは後退代入によるものである。ＬＵ分解と前進代入の影響をあまり受けないものはピーク２に，影響を受けるものはピーク３に集中するものと考えられる。大部分の要素のＬＵ分解と前進代入は，第１のピークで終っているので，後に残るのは特定のノードに多数の素子が接続されたもの（これは，行列中では多数の要素を含んだ行および列として現れる）である。このノードに関係しない要素の後退代入は，第１のピークが終った直後に実行可能であり，これが第２のピークを形成する。

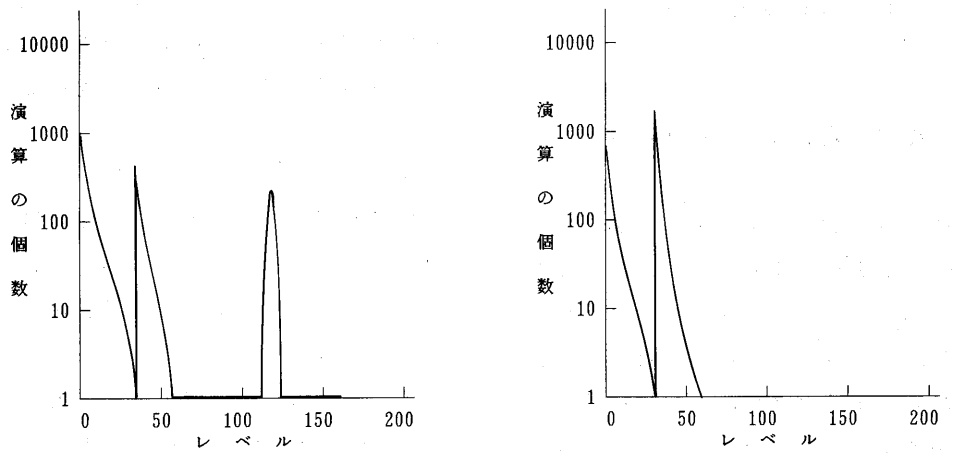
ａ＝ａ／ｂタイプについてみると，演算の数は最初多くて，あとは急激に減少している。ピークは１個しかない。これは，ａ＝ａ／ｂタイプの演算がＬＵ分解と前進代入にしかないためである。

図4.2にはＬＵ分解・前進代入・後退代入の３段階を完全に分離したときのレベルの分布を示す。３段階を分離せずにレベル付けしたほう（図4.1）が，効率が良いのが一目瞭然である。



(a) ADD16 a=a-b×c　　(b) ADD16 a=a/b

図4.1　演算の分布



(a) ADD16 a=a-b×c　　(b) ADD16 a=a/b

図4.2　各過程を分離したときの演算の分布

5．行列求解の実行

今まで述べた方法を，筆者らが開発した，ベクトル処理向き汎用回路解析プログラムＦＩＮＡＰ－ＶＰに組み込み，当社のスーパーコンピュータＦＡＣＯＭ　ＶＰ－４００で実行した。

5．1　ＦＯＲＴＲＡＮによる実行

作成したインデックステーブルに従って演算を実行するためのプログラムは，ＦＯＲＴＲＡＮで容易に記述することができる。実行するときには，同じレベルの演算は並列実行可能なのでベクトル計算をする。このとき，ベクトルパイプライン起動時のオーバーヘッドにより，ベクトル長が非常に短いときには，スカラ命令のほうが速く計算できる。そこで，あらかじめスレッショルドをきめておき，並列化可能な演算数がそれよりも多ければベクトル演算を，少なければスカラ演算をさせるようにした。

5．2　ベクトルコード生成法による実行

拡張レベル付けによるインデックスを使った計算は，ベクトル化したＦＯＲＴＲＡＮでも十分高速であるが，それでも，インデックスを取り出すときのアドレス計算，あるいはループを回るときのオーバーヘッドなど，無駄な部分がまだある。これらを取り除いて可能な限りの高速化をはかるために，ベクトルコード生成を行った。

回路解析では，同じ形の行列を多数回解くので，あらかじめ行列を解く過程をシミュレートして，その行列を解くための手順を機械コードとして生成しておくことにより，行列求解を高速に実行できるようになる。この手法をコード生成法と呼ぶ。

今回は，このコード生成法において，ベクトルコードとスカラコードとを混在して生成するようにした。コード生成時には，レベル付けにより作成されたインデックスを調べて，ベクトル長の長いものに対してはベクトルコードを生成し，短いものに対してはスカラコードを生成する，というようにした。

6．　実行テスト

表6.1のような；いくつかの規模の異なる回路に対して実行した結果を，図6.1に示す。ＥＣＬｍは，ＥＣＬインバータをｍ段接続した回路，ＡＤＤｍはｍビットｎＭＯＳ加算器である。

図6.1で，スカラFORTRAN とあるのは，拡張レベル付けＬＵ分解法により作成したインデックスをベクトル化しない FORTRAN プログラムで解いた場合である。したがって，一般の FORTRANによる行列求解よりもアドレス計算が少なく，高速である。また，スカラコード生成では，要素のタイプも考慮した最適化を行っている。ベクトルモードでの解析では，行列求解以外の部分もベクトル化している。

行列求解のみについてみると，ベクトル FORTRAN はスカラFORTRAN に比べ，４～７倍高速化している。ベクトルコード生成で

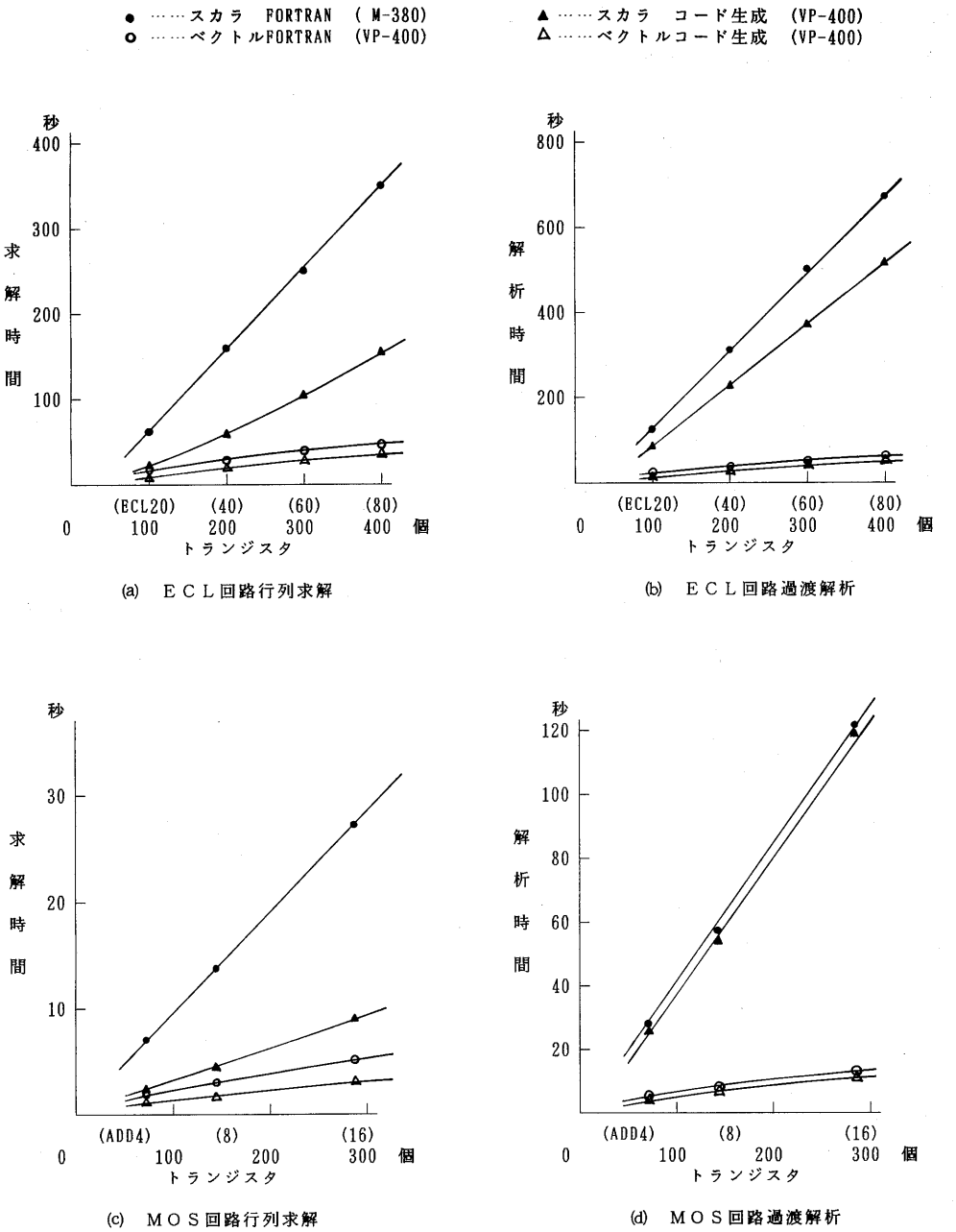


図6.1 各回路の行列求解時間および過渡解析時間

は，７～１０倍程度高速になっており，ベクトルFORTRAN に比べても 1.3～1.8 倍高速である。

解析全体の実行時間では，行列求解のみの高速化に比べ，さらに高速化している。これは，行列求解以外の処理（モデル関数計算・行列更新）のベクトル化率が非常に高く，高速になっているためである。

拡張レベル付けＬＵ分解法では，回路の規模が大きくなるにつれて，スカラに対する高速率が高くなっている。実行時間は飽和していく傾向にあり，今回の方法が規模の大きな回路に対して特に有効であることを示している。

次に，各方法で前処理にどの程度の時間がかかるのかを調べてみる。表6.2に前処理に要する時間を示す。どれも，行列求解時間に比べて十分に小さく，２%以下である。なお，スカラコード生成法の前処理時間が長いのは，生成されるコードの最適化を行うのに時間がかかっているからである。

最後に，メモリ量について検討してみる。表6.3に各方法で行列求解に必要とするメモリ量を示す。単位はキロバイトである。概して，必要なメモリ量は同程度ではあるが，スカラコード生成法では，より多くのメモリを必要とする。ベクトルコード生成法では，FORTRAN による場合よりも５～１０%程度増えているだけである。これは，ベクトル命令では，数百個分の計算を数十バイトで記述できるために，スカラ命令のように多くのメモリを必要とはしないからである。

表6.1　測定用回路の概要

回路名（タイプ）	デバイス数	行列次元	非零要素数	スパース率
ECL20 (Bip.)	140	867	3375	99.6 %
ECL40 (Bip.)	280	1731	6743	99.8
ECL60 (Bip.)	420	2588	10111	99.9
ECL80 (Bip.)	560	3459	13479	99.9
ADD4 (MOS)	72	198	828	97.9
ADD8 (MOS)	144	378	1504	98.9
ADD16 (MOS)	288	738	2856	99.5

表6.2　各方法の前処理に要する時間（秒）

回路名	スカラ FORTRAN	スカラコード生成	ベクトル FORTRAN	ベクトルコード
ECL20	0.10	0.29	0.10	0.11
ECL40	0.21	0.58	0.21	0.22
ECL60	0.31	0.90	0.31	0.33
ECL80	0.41	1.21	0.41	0.43
ADD4	0.02	0.03	0.02	0.02
ADD8	0.03	0.07	0.03	0.03
ADD16	0.05	0.14	0.05	0.05

表6.3　行列求解に必要とするメモリ(k Byte)

（　）内はFORTRAN に対する比率

回路名	FORTRAN	スカラコード生成	ベクトルコード生成
ECL20	136 (1.00)	152 (1.11)	152 (1.12)
ECL40	280 (1.00)	324 (1.16)	304 (1.09)
ECL60	420 (1.00)	500 (1.19)	440 (1.06)
ECL80	556 (1.00)	688 (1.24)	580 (1.04)
ADD4	24 (1.00)	28 (1.20)	28 (1.13)
ADD8	44 (1.00)	52 (1.22)	48 (1.10)
ADD16	84 (1.00)	112 (1.34)	92 (1.09)

7．結　論

ＬＵ分解法に基づく連立一次方程式の求解が全て２種類の基本的な演算のみから行われることに着目し，全演算をレベル付けすることで，ベクトル長の十分に長いベクトル化を実現した。また，この方法に最適なＬＵ分解・前進代入・後退代入の方法を検討した。

さらに，このアルゴリズムを汎用回路解析プログラムＦＩＮＡＰ－ＶＰに組み込んで，実行した。その結果，

(1) 行列求解では，同じ方法をスカラで実行した場合の４～７倍高速，
(2) ベクトルコード生成法にすれば，さらに 1.3～1.8 倍高速になる，
(3) 大きな回路ほど高速化できる，
(4) レベル付けの時間は解析時間にくらべて十分に小さい，
(5) 使用メモリ量はスカラコード生成法よりも少ない，

ということが，わかった。

なお，本手法は，回路解析のみならず，スパース行列の多数回の求解を含む問題には，共通して有効であると考えられる。

謝　辞

本研究について，日頃から御指導していただいている，㈱富士通研究所システム研究部上原部長，石井部長代理および白石主任研究員に深く感謝の意を表します。

【参考文献】

〔1〕D.A.Calahan and W.G.Ames,"Vector Processors : Models and Applications", IEEE Trans. Circuit Syst., Vol.CAS-26, Sept. 1979.

〔2〕D.A.Calahan,"Multi-Level Vectorized Sparse Solution of LSI Circuits", Proc., IEEE Int. Conf. Circuits Comp., Oct. 1980.

〔3〕A.Vladimirescu and D.O.Pederson,"A Computer Program for the Simulation of Large-Scale-Integrated Circuits", Proc., IEEE Int. Symp. Circuits Syst., Apr. 1981.

〔4〕A.Vladimirescu and D.O.Pederson,"Performance Limits of the CLASSIE Circuit Simulation Program", Proc., IEEE Int. Symp. Circuits Syst., May 1982.

〔5〕A.Vladimirescu and D.O.Pederson,"Circuit Simulation on Vector Processors", Proc., IEEE Int. Conf. Circuits Comp., Sept. 1982.

〔6〕S.D.Hamm and S.R.Beckerich,"VAMOS : Circuit Simulation Program for a Vector Computer", Proc., IEEE Int. Conf. Computer-Aided Design, Nov. 1983.

〔7〕F.Yamamoto, K.Kobayashi, and Y.Umetani, "A Comparative Study of Two Vectorized LU Decomposition Algorithm for Large Scale Unstructured Sparse Matrix", Proc., IEEE Int. Symp. Circuits Syst., June 1985.

〔8〕F.Yamamoto and S.Takahashi,"Vectorized LU Decomposition Algorithms for Large-Scale Circuit Simulation", IEEE Trans. Computer-Aided Design, Vol.CAD-4, No.3, July 1985.

〔9〕T.Kage, Y.Oishi, and M.Ishii,"A Circuit Analysis Program Using an Attached Array Processor - FINAP-AP -", Proc., IEE European Conf. Electronic Design Automation No.232, Mar. 1984.

〔10〕W.Nagel,"SPICE2, A Computer Program to Simulate Semiconductor Circuits", Univ. of California, Berkeley, Memo No. ERL-M520, May 1975.

〔11〕鹿毛，下郡「ベクトル計算機による回路解析プログラム（FINAP-VP）の高速化の検討」，信学技報 Vol.85,No.279, CAS85-168, 1986年 1月.