5H-08　配列間接アクセスを用いないコード生成法による電子回路シミュレーションの高速化

間中 邦之 †, 刑部 亮 †, 前川 仁孝 ‡, 笠原 博徳 †
早稲田大学理工学部電気電子情報工学科 †, 千葉工業大学情報工学科 ‡

1　はじめに

近年の半導体技術の進歩と共にVLSIの集積度は上昇し、回路の設計と検証に多くの時間とコストが必要になっている。中でも電子回路のシミュレーション [1] に要する時間の短縮はチップ開発期間短縮のための重要な課題の一つである。従来より広く用いられている電子回路シミュレータSPICE[2] ではこのシミュレータ時間短縮のためスパース行列の直接法を用いた求解に、コード生成法を用いループフリーコードを生成していた。しかし、スパース行列の格納のために用いている配列間接アクセスが、並列化を含めそれ以上の処理高速化を阻害する要因の一つになっている。本稿では、この配列間接アクセスを用いずループフリーコードを生成することにより、WS(WorkStation) やPC(PersonalComputer) などの単一プロセッサシステム上で、回路シミュレーションをSPICEより数十倍以上高速に行なうことを可能としたので、その結果について報告する。

2　直接法を用いた回路シミュレーションの高速化手法

本節では、直接法を用いた電子回路シミュレーションの求解手法について述べる。

一般的な修正節点解析法を用いて回路方程式のモデル化 [3] を行うと、電子回路の動特性は非線形連立微分方程式

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, t) = 0, \quad \mathbf{x}(t=0) = \mathbf{x}_0$$

で表せる。ここで、$\mathbf{x}$ は節点電圧などに使われる解ベクトルである。直接法を用いて上式を解く場合には、非線形連立微分方程式を解くためのインプリシット積分法、及び、非線形方程式を解くためのNewton-Raphson法、連立方程式を解くための直接行列求解を組み合わせた手法がよく用いられる。

電子回路の非線形連立微分方程式は特性が急に変化するスティフな系となる事から、本研究ではインプリシット積分法として、スティフな系に強い可変ステップ可変次数のBDF (BackwardDifferentialFormula) 法 [4] を用いる。更にSPICEの解析手法である台形積分法とGear法を利用することも可能である。直接行列求解にはクラウト法を用い、求解コードをループフリーコードで生成する。また、本回路シミュレーションは、ユーザがSPICE形式の回路リストを入力することにより、自動的に配列間接アクセスの排除や定数伝搬などの最適化を行ったFortran言語によるソースコードを生成する。ユーザはこの生成コードを各使用するマシンのFortranコンパイラでコンパイルし、回路シミュレーションを行う。

筆者等はループフリーコードの並列化を行い、OSCAR用自動並列化電子回路専用コンパイラを開発しているが [5]、本研究では生成コードを逐次Fortran言語とすることでWSやPC上でも高速シミュレーションを可能とする。

以下では、本手法の特徴である配列間接アクセス、定数伝搬、生成コードについて述べる。

Evaluation of Electronic Circuit Simulation which generate code without array indirect access
Kuniyuki MANAKA, Ryo OSAKABE, Yoshitaka MAEKAWA, Hironori KASAHARA
† Dept. of Electrical, Electronics and Computer Engineering, School of Science and Engineering, Waseda Univ.
‡Dept. of Computer Science, Chiba Institute of Technology.

2.1　配列間接アクセス

本節では、配列間接アクセスを用いない手法について述べる。

SPICEに代表される直接法回路シミュレーションは、回路の種類や構造、また回路の動作速度によらずどのような回路も解析する事が可能である。これは全ての変数を連立させて有限の手順で計算機精度内で精確な解を与える直接法行列求解の基本的な性質によるものである [6]。この直接法行列求解の回路方程式の係数行列はランダムスパース性が高いため、配列間接アクセスを用いて係数行列の演算を行う。このため、配列を参照する時に、メモリアクセス回数が多くなり、これが処理速度向上を阻む原因の一つとなっている。

そこで本手法では、ループフリーコードの生成の際、全ての変数、定数、一時変数全てに対して配列を一切用いないコードを生成することにより、記憶領域も最小限の大きさで良く、配列へのアクセスを無くし値を参照する時にメモリを直接参照することで、従来の間接参照を行うコードに比べ、単一プロセッサ上での処理速度を大幅に向上させる。

2.2　定数伝搬

本手法では求解コードの生成時に計算量を減らすために、コード内で定数と判断される変数については、定数伝搬を行い定数として定義し、計算可能なステートメント部分についてはループフリーコード生成時に内部で計算を行うようにする。また、Newton-Raphson法による収束ループや、時間発展ループなどの、ループ内では常に変化しない変数や、常に計算結果が同じになる変数を演算するステートメントに対してループ外へと追い出す搬定数伝搬も同時に行い、さらなる処理の高速化を行っている。

2.3　ループフリーコード

本節では、本電子回路シミュレーションが生成するコードについて述べる。生成したループフリーコードの例を図1に示す。生成する線形連立方程式の求解コードは、逐次計算において最高速であることが知られているコード生成法 [7] を用いる。これは通常のFortran言語で記述されたクラウト法による求解より数十倍も高速であることが知られている [8]。生成コードは一般的なFortran言語のみで書かれているため、使用マシンのネイティブFortranコンパイラでコンパイルし、WSやPCで実行する事が可能である。

```
      PARAMETER(c1=0.000000e+00)
      ⋮
100 IF(c11.GE.v2)THEN
C     # Newton-Raphson
      ⋮
C     #  make b matrix
      T24 =(v125 + v126)* v120
      T25 =(v126 + v127)* v116
      T26 = T25 - T24
      v30 = T26 - v130
      ⋮
C     #  LU decomposition
      T93 = v236 + v239
      T94 = T93 * c28
      v51 = T94 *( c2 / v47 )
      ⋮
C     #  forward
      v91 = v30 - S5
      T49 = v31 * v91
      ⋮
C     #  backward
      T51 = v101 *(c2 / v32)
      v102= ABS(v97 - T51)
      v97 = T51
      ⋮
      IF(v6.GE.c2)GOTO 200
      WRITE (*,1000) 'V(1)=',v48
      ⋮
      GOTO 100
      END IF
      END
```

図 1: 生成コード例

3 回路シミュレーションの性能評価

ここでは、本手法の回路シミュレーションの性能を単一プロセッサの WS 上で評価した結果について述べる。

3.1 評価環境

本評価は SPICE3f.4 付属のサンプル回路及び幾つかの小規模回路の過渡解析を例として、単一プロセッサでの過渡解析に要する時間の測定を行い比較した。評価マシンは Sun Microsystems,Inc.Ultra 60 で、CPU は UltraSPARC-II 360MHz、オンチップ L1Cache は I-Cache・D-Cache 共に 16KB、L2Cache は 4MB、Memory Size は 512MB、OS は Solaris 2.6 である。評価の比較対象として代表的な回路シミュレーションである SPICE3f.4 (Univ. of California, Berkeley) を使用する。この比較では、本提案手法で生成された Fortran コードをコンパイルした実行形式ファイルを実行させた時間と、SPICE3f.4 の ICL(Interactive Command Language) を使用した時の実行時間を用いた。Fortran コンパイラは Sun の Fortran77 SC4.2 を用い、コンパイラオプションはマシン最適化、浮動小数点演算最適化、最適化数学ライブラリのインライン展開等を行う '-fast' を使用する。また、インプリシット積分法を共に台形積分法および Gear 法へと変更した場合の性能評価を行う。

3.2 評価結果

図 2は SPICE の台形積分法と本手法の BDF 法、台形積分法について、図 3は SPICE の Gear 法と本手法の Gear 法について、各評価回路での過渡解析に要する時間を評価し、SPICE を 1 とした場合の速度向上率をグラフとして示したものである。グラフ中、左から 8bit 加算器、3bit nand 回路、2bit 乗算器、ecl compatible schmitt trigger(schmitt ckt)、cascaded rtl inverters(rtlinv ckt) の評価結果を示す。生成された回路行列サイズは順に 754,378,114,19,12 である。

台形積分公式を用いた SPICE と本手法の場合は (図 2)、平均で約 16 倍、最大で 3bit nand 回路の 1070ms に対して 27.5ms と約 38.8 倍処理を高速化できることが確認された。また、台形積分公式を用いた SPICE に対する BDF 法を用いた本手法を比較した場合 (図 2) には、平均で約 13 倍、最大で同じく 3bit nand 回路の 1070ms に対して 37.0ms と約 29 倍処理を高速化できることが確認された。さらに、共に Gear 法を用いた場合 (図 3) でも SPICE に対し本手法は平均で約 14.6 倍、最大で 3bit nand 回路の 910ms に対して 30.6ms と約 29.7 倍処理を高速化できることが確認された。次にインプリシット積分法を変更した場合で比較してみると、例えば 8bit 加算器の場合には、BDF 法が 634ms に対して台形積分法が 422ms、Gear 法が 434ms、2bit 乗算器の場合には、BDF 法が 80.6ms に対して台形積分法が 44.1ms、Gear 法が 40.3ms と、全体的に BDF 法より台形積分法及び Gear 法の方が実行時間が短くなる。これは BDF 法の可変ステップによるタイムステップ制御、タイムステップが可変であるため、素子数が増加することによって、正確な電圧等の値を求めようとタイムステップを小さくし、常にタイムステップが小さい状態で解析を行なう場合がある事が原因と考えられる。

以上の結果より、本手法により回路シミュレーションが単一 WS 上で大幅に高速化できる事が確認できた。

4 まとめ

本稿では、直接法を用いた回路シミュレーションの高速化のため配列間接アクセスを用いないループフリーコードを生成する手法の提案を行なった。この手法により単一プロセッサシステムである WS 上で SPICE3f.4 と比較して、シミュレーション時間を 2 倍から 39 倍高速化できる事を確認できた。

今後は、使用する変数を事前にキャッシュに置くプレロードの適用、より大規模な回路に適用するため、回路分割を利用した粗粒度並列処理やステートメントレベルの近細粒度並列処理等を組み合わせた並列化手法を提案していく予定である。

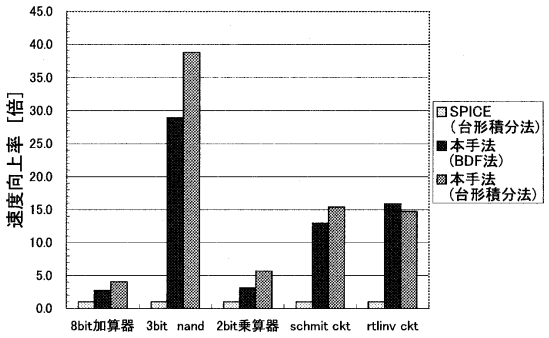

図 2: SPICE に対する速度向上率 (BDF 法、台形積分法)

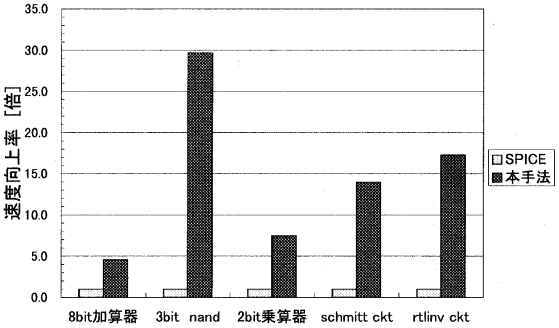

図 3: SPICE に対する速度向上率 (Gear 法)

参考文献

[1] Sadayappan, P. and Visvanatan, V.: Circuit Simulation Shared Memory Multiprocessors,IEEE Trans. Computers, Vol.C-37 ,No.12,pp.1634-1642 (1988).

[2] Cohen, E.: Program Reference for SPICE2, Electronics Res. Lab., Mem. No.ERL-M592, Univ. of California, Berkeley(1976).

[3] Ho, C. W., Ruehli, A. E. and Brennan, P. A.: The Modified Nodal Approach to Network Analysis, IEEE Trans. Circuits and Systems, Vol.CAS-22, No.6, pp.504-509 (1975).

[4] Brayton, B. K., Gustavson F. G. and Hachtel G. D.: A New Efficient Algorithm for Solving Differential Algebraic Systems Using Implicit Backward Differential Formulas, Proc. IEEE, Vol.60, No.1, pp.98-108 (1978).

[5] 前川 仁孝, 高井 峰生, 伊藤 泰樹, 西川 健, 笠原 博徳,“スタティックスケジューリングを用いた電子回路シミュレーションの粗粒度/近細粒度階層型並列処理手法”, 情報処理学会論文誌, Vol.37, No.10, Oct.1996.

[6] 西原明法, 鹿毛哲朗, 奥村万規子, 山村清隆, “ポスト SPICE 回路シミュレータ”, 電子情報通信学会誌, Vol82, No.1, pp47-54, 1999

[7] Duff, I. S., Erisman, A. M., Reid, J. K.: Direct Method for Sparse Matrices, Oxford Univ. Press (1986).

[8] Fukui, Y., Yoshida, H. and Higono, S.: Supercomputing of Circuit Simulation, Proc. Supercomputing'89, pp.81-85 (1989).