配列間接アクセスを用いないコード生成法を用いた電子回路シミュレーション手法の性能評価

黒田　亮[†]　木村啓二[†]　笠原博徳[†]

本稿では、ランダムスパースマトリクス処理に伴う配列間接アクセスを除去したループフリーコード生成による電子回路シミュレーションの高速化手法を WS 及び PC 上で評価した結果について報告する。広く用いられている電子回路シミュレータ SPICE では直接法を用いたスパースマトリクス求解の高速化のために、コード生成法により非零要素のみの計算を列挙したループフリーコードを生成している。しかし、その際のスパースマトリクス処理のための配列間接アクセスが処理高速化を阻害する要因の一つになっている。そこで本論文では間接配列アクセスを用いないループフリーコードを生成する電子回路シミュレーション手法を間接法による求解が難しい BJT を含む回路に対して適用し、単一プロセッサの WS 及び PC 上で性能評価を行なった。その結果、過渡解析を SPICE3f5 より 2 倍から 110 倍高速に行なえることが確認され、さらにこの高速化はメモリアクセスの大幅な削減によることが確認された。

Performance Evaluation of Electronic Circuit Simulation Using Code Generation Method without Array Indirect Access

Akira Kuroda[†], Keiji Kimura[†] and Hironori Kasahara [†]

This paper evaluates performance of a fast sequential circuit simulation scheme using the loop free code without the array indirect accesses. This scheme allows us to get several tens of times higher processing performance than SPICE version 3f5 on a WS and a PC. The array indirect accesses for the sparse matrix solution in SPICE have been one of the factors that prevents from efficient processing. This paper describes the circuit simulation scheme using loop free code without any array indirect accesses and its performance evaluation shows the scheme gives us 2 to 110 times better performance than SPICE3f5 on a WS and a PC. The performance by reducing the memory accesses overhead significantly.

1　はじめに

近年の半導体技術の進歩により VLSI の集積度は上昇し、回路の設計と検証に多大な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 [6)~18)] 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。

電子回路シミュレーションの過渡解析では大きく分けて直接法 [7),8),13),18)] と間接法 [10)~12)] の 2 つの方法が使用される。ウェブフォーム、ノンリニアリラキゼーション等の間接法は CMOS 回路以外では収束性が悪く適用可能回路が限定されるという問題点がある。これに対し、直接法を用いた手法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ことは可能であるが [12)]、大きな記憶領域を必要とするという [13),14)] 問題点がある。また、直接法の電子回路シミュレータとして広く用いられている SPICE[13)] ではこのシミュレーション時間短縮のためスパースマトリクスの直接法 [18)] 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る。しかしながら、スパースマトリクスの格納のために用いられている配列間接アクセスが、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。

本論文では、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、電子回路シミュレーションを高速化する手法を述べる [5)]。さらに、本手法をＷＳやＰＣ上で評価した結果について報告する。

† 早稲田大学 理工学部 コンピュータネットワーク工学科, Dept. of Computer Science,School of Electronics and Engineering, Waseda Univ.

以下、第 2 章で直接法を用いた電子回路シミュレーションの高速化手法について、第 3 章で単一プロセッサ上での BJT(Bipolar Junction Transister) を含む回路を用いた電子回路シミュレーションの性能評価について述べる。

2　直接法を用いた電子回路シミュレーションの高速化手法

本章は、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。

2.1　直接法を用いた回路シミュレーションの求解手法

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

$$\mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, t) = 0, \quad \mathbf{x}(t=0) = \mathbf{x}_0 - (1)$$

と表せる。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、非線形方程式を解くための Newton-Raphson 法、線形方程式の求解が必要となる。

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、本シミュレータ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 (Backward Differential Formula) 法及び SPICE でも用いられている Trapezoidal 法、Gear 法も実装している。直

接法を用いたマトリクス求解にはSPICEと同様にクラウト法を用い、非零要素に関する計算だけを列挙し高速化を図るループフリーコードを生成する。

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、生成されるループフリーコード[7),8),13),18)]、そして本手法を実現した電子回路シミュレータ構成について述べる。

2.2 ループフリーコード

本節では、本電子回路シミュレータが生成するループフリーコードについて述べる。

このランダムスパースマトリクスを係数マトリクスに持つ線形連立方程式の求解コードは、コード生成法[18)]が逐次計算において最高速であることが知られている。例えば，ループフリーコードは通常のFortran言語で記述されたクラウト法による求解よりも数十倍も高速であることが報告されている[7)]。生成されるループフリーコードの例として、図1にSPICE3f5付属のサンプル回路cascaded rtl invertersにおける係数行列のLU分解部分を示す。この回路の過渡解析で生成される全体のコードサイズは約900行である。

```
          :                          T81= v43 * c46
C     #  LU decomposition            T82= c48 - T81
      T63= v206 + v207 + v210        v44= T82 + c94
      v34= T63 + S9                  v50= S2 * ( c2 / v44 )
      T66= v207 + v210               T89= v236 + v237
      v35= T66 * c28                 v47= T89 + S12
      T68= v209 - v207               v48= v238 - v236
      v37= T68 * ( c2 / v34 )        T93= v236 + v239
      T69= v37 * v35                 T94= T93 * c28
      T70= T69 * c28                 v51= T94 * ( c2 / v47 )
      T71= v207 + v208               T96= v50 * S2
      T72= T70 + c96                 T97= v51 * v48
      v38= T71 + T72                 T98= T97 + T96
      v41= S5 * ( c2 / v38 )         T99= T98 * c28
      T76= v41 * S5                  T100= v235 + v236 + v239
      T77= T76 * c28                 T101= T99 + c94
      T79= T77 + c96                 v52= T100 + T101
      v42= c45 + T79                        :
      v43= c47 * ( c2 / v42 )               :
```

図 1: 生成コード例

図1では、例えば係数行列の要素はスカラ変数v34等として表されている。同様に各識別子の頭文字がc及びSの場合は定数、v及びTの場合は変数を表している。ここでvに続く数字はランダムスパースマトリクスの非零要素に対して1対1対応になるように順に割り振られる。また、1つ1つのステートメントは算術代入文からなるスカラ計算で記述される。

2.3 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シミュレーション手法について述べる。

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。そのため、SPICEに代表される直接法回路シミュレーションは非零要素の保持のために配列間接アクセスを用いて係数行列の演算を行う。しかしながら、この配列間接アクセスにより非零要素を参照する際の主メモリアクセス回数 (すなわち、キャッシュミス回数) 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生成の際、全ての非零要素に対して配列の間接参照を一切用いずスカラ変数のみでコードを生成する。本コード生成法によりデータ記憶領域も最小限に抑えることができ、かつ、配列の添字計算、間接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、単一プロセッサ上での処理の高速化を実現できる。また、今後のチップマルチプロセッサ[4),22)]上での並列処理において、ステートメント間のデータ依存解析を容易にし、並列化を行ないやすいという効果もある。

2.4 定数伝搬

また、本手法ではループフリー求解コードの生成時に過渡解析実行時の計算量を減らすために可能な限り定数伝搬を行い、コード内で定数化可能な変数を定数化する。従来のコード生成法では配列内に存在した定数化可能な変数あるいは不変式もそのまま最内側のNewton-Raphsonループで繰り返し計算されていたのに対し、本手法では、定数伝搬によりこれらのループ不変式をNewton-Raphsonループだけでなく、最外側の時間発展ループの外側へ移動することが可能となる[5)]。またコード生成システム中で、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先行評価 (事前計算) を行っている。

さらに係数行列のLU分解では、計算を左上の要素から右下の要素へと行うので、ある要素の計算を行う際は、その要素の同一列上部及び同一行左部の要素が計算に利用される。そこで係数行列の定数の部分を行列の左上、変数を右下に配置するようにMarkowitz法を併用してリオーダリングを施している。これにより、計算を行うべき要素が定数になる確率が高くなり、その結果、定数伝搬の対象領域を広げることができる。

2.5 電子回路シミュレータの構成

本手法を実現する電子回路シミュレータ構成を図2に示す。本電子回路シミュレータはFortranコンパイラのプリプロセッサとして実装されている。ユーザがSPICE形式の回路リストを入力すると、SPICEの高速化を難しくしていた配列間接アクセスを排除し、さらに定数伝搬などの最適化を行ったFortran言語で記述されたコードが生成される。生成されたコードはWSやPCなどの単一プロセッサマシンのネイティブFortranコンパイラや、マルチグレイン

並列処理を行う OSCAR マルチグレイン自動並列化コンパイラ [1),21)] を用いてコンパイルし、WS や PC、OSCAR 型チップマルチプロセッサ [4)] で実行することが可能である。また今後、大規模回路の回路分割を行ない部分回路間の並列性を OpenMP で記述し SMP サーバ上で実行したり、さらに部分回路内を直接法で求解しそれを近細粒度で階層的にマルチグレイン並列処理を行なう機能もインプリメント中である。

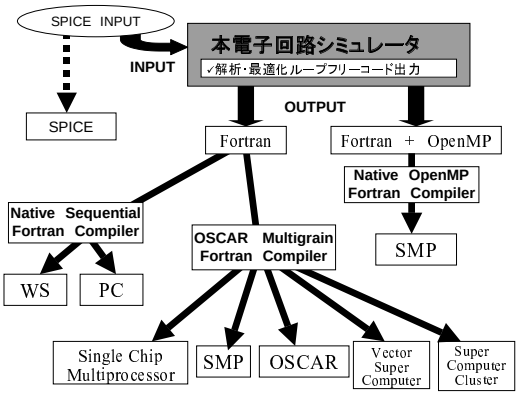

図 2: 本電子回路シミュレーションの構成

3 電子回路シミュレーションの性能評価

ここでは、本電子回路シミュレーションの性能を単一プロセッサの UltraSPARC-II ベースの WS 及び Pentium4 ベースの PC 上で評価した結果について述べる。

3.1 評価環境

本論文では SPICE3f5 付属の BJT を含むサンプル回路及び幾つかの小規模回路を用いたことにより、単一プロセッサでの過渡解析に要する時間の測定により評価を行なった。評価マシンの詳細については、表 1 に示す。WS の構成は CPU UltraSPARC-II 450MHz、L1 I-Cache 16KB、L1 D-Cache 16KB、L2-Cache 4MB、OS Solaris 9、コンパイル Sun ONE Studio 7 Enterprise Edition for Solaris である。また、PC の構成は CPU Pentium4 3.0Hz、L1 I-Cache 12 μ op、L1 D-Cache 16KB、L2-Cache 512KB、OS Linux、コンパイル GNU Fortran77 version 2.95 .4 である。

評価の比較対象として代表的な回路シミュレータである SPICE3f5[2)] を使用する。本評価では、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の実行部分と、SPICE3f5 の過渡現象シミュレーション部分の処理時間を比較した。また、インプリシット積分法を共に Trapezoidal 法および Gear 法へと変更した場合の性能評価を行う。

表 1: 評価 WS、PC の詳細

	WS	PC
CPU	UltraSPARC-II	Pentium4
Clock	450MHz	3.0GHz
L1 I-Cache	16KB	12 μ op
L1 D-Cache	16KB	16KB
L2-Cache	4MB	512KB
Memory Size	512MB	1GB
OS	Solaris 9	Linux Kernel2.4.26
Compile	Sun ONE Studio 7 Enterprise Edition for Solaris	GNU Fortran77 version 2.95.4

3.2 評価結果

図 3 は各評価回路を WS 上において SPICE3f5 及び本電子回路シミュレータの Trapezoidal 法を用いた過渡解析実行時間を棒グラフで示し、SPICE3f5 に対する本電子回路シミュレータの速度向上率を折れ線グラフで示す。図 4 は各評価回路を WS 上において SPICE3f5 及び本電子回路シミュレータの GEAR 法を用いた過渡解析実行時間を棒グラフで示し、SPICE3f5 に対する本電子回路シミュレータの速度向上率を折れ線グラフで示す。図 5 は各評価回路を PC 上において SPICE3f5 及び本電子回路シミュレータの Trapezoidal 法を用いた過渡解析実行時間を棒グラフで示し、SPICE3f5 に対する本電子回路シミュレータの速度向上率を折れ線グラフで示す。図 6 は各評価回路を PC 上において SPICE3f5 及び本電子回路シミュレータの GEAR 法を用いた過渡解析実行時間を棒グラフで示し、SPICE3f5 に対する本電子回路シミュレータの速度向上率を折れ線グラフで示す。

各評価回路は、左から latch、2bit 乗算器、3bit 乗算器、4bit 加算器、8bit 加算器、4bit nand 回路示す。

生成された回路行列サイズ、BJT 数及びステートメント数は表 2 に示す。表 2 は、評価回路名、係数行列サイズ、BJT 数および Trapezoidal 法におけるステートメント数を示す。例えば、latch において係数行列サイズ 65X65、BJT 数 14、ステートメント数 6557 行である。

WS 上では、図 3 に示すように、Trapezoidal 法を用いた場合の SPICE3f5 と本電子回路シミュレータは最大で 4bit nand 回路において過渡解析時間が SPICE3f5 では 1100ms、本電子回路シミュレータでは 10ms であり、本電子回路シミュレータは 110 倍処理を高速化できることが確認された。3bit 乗算器は、過渡解析時間が SPICE3f5 では 1200ms、本電子回路シミュレータでは 469ms であり、本電子回路シミュレータは 2.6 倍処理を高速化できることが確認

表 2: 評価回路の詳細データ

評価回路名	係数行列サイズ	BJT 数	ステートメント数 (Trapezoidal 法)
latch	65×65	14	6557
2bit 乗算器	114×114	10	5745
3bit 乗算器	450×450	42	26250
4bit 加算器	378×378	36	17570
8bit 加算器	754×754	72	34823
4bit nand 回路	754×754	72	33203

された。8bit 加算器は、過渡解析時間が SPICE3f5 では 1080ms、本電子回路シミュレータでは 210ms であり、本電子回路シミュレータは 4.9 倍処理を高速化できることが確認された。また、6 例の評価回路は平均で約 22 倍の高速化が確認された。

さらに、図 4 に示すように、Gear 法を用いた場合の SPICE3f5 と本電子回路シミュレータは最大で 4bit nand 回路において過渡解析時間が SPICE3f5 では 800ms、本電子回路シミュレータでは 20ms であり、本電子回路シミュレータは 40 倍処理を高速化できることが確認された。3bit 乗算器は、過渡解析時間が SPICE3f5 では 1880ms、本電子回路シミュレータでは 430ms であり、本電子回路シミュレータは 4.4 倍処理を高速化できることが確認された。8bit 加算器は、過渡解析時間が SPICE3f5 では 1140ms、本電子回路シミュレータでは 290ms であり、本電子回路シミュレータは 3.9 倍処理を高速化できることが確認された。また、6 例の評価回路は平均で約 11 倍の高速化が確認された。

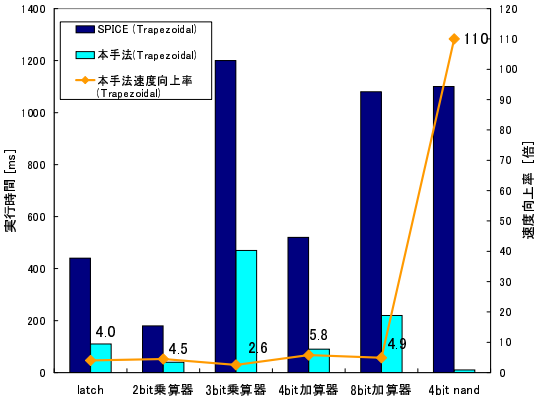

図 3: WS 上での SPICE3f5 に対する Trapezoidal 法の速度向上率

PC 上では、図 5 に示すように、Trapezoidal 法を用いた場合の SPICE3f5 と本電子回路シミュレータは最大で 4bit nand 回路において過渡解析時間が SPICE3f5 では 160ms、本電子回路シミュレータでは 10ms であり、本電子回路シミュレータは 16 倍処理を高速化できることが確認された。3bit 乗算器は、過渡解析時間が SPICE3f5 では 180ms、本電子回路シミュレータでは 109ms であり、本電子回路シミュレータは 1.6 倍処理を高速化できることが確認

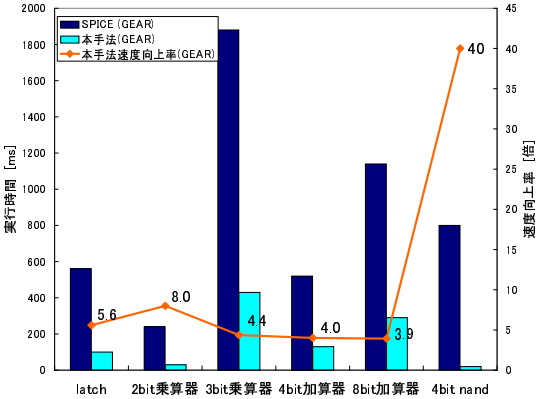

図 4: WS 上での SPICE3f5 に対する Gear 法の速度向上率

された。8bit 加算器は、過渡解析時間が SPICE3f5 では 180ms、本電子回路シミュレータでは 90ms であり、本電子回路シミュレータは 2 倍処理を高速化できることが確認された。また、6 例の評価回路は平均で約 4.9 倍の高速化が確認された。

さらに、図 6 に示すように、Gear 法を用いた場合の SPICE3f5 と本電子回路シミュレータは最大で 4bit nand 回路において過渡解析時間が SPICE3f5 では 120ms、本電子回路シミュレータでは 10ms であり、本電子回路シミュレータは 12 倍処理を高速化できることが確認された。3bit 乗算器は、過渡解析時間が SPICE3f5 では 280ms、本電子回路シミュレータでは 100ms であり、本電子回路シミュレータは 2.8 倍処理を高速化できることが確認された。8bit 加算器は、過渡解析時間が SPICE3f5 では 180ms、本電子回路シミュレータでは 100ms であり、本電子回路シミュレータは 1.8 倍処理を高速化できることが確認された。また、6 例の評価回路は平均で約 5.1 倍の高速化が確認された。

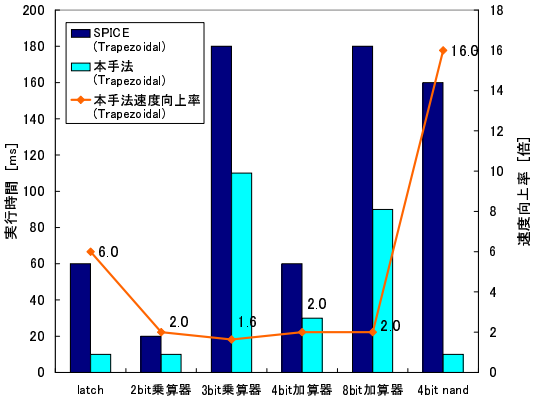

図 5: PC 上での実行時間及び SPICE3f5 に対する Trapezoidal 法の速度向上率

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

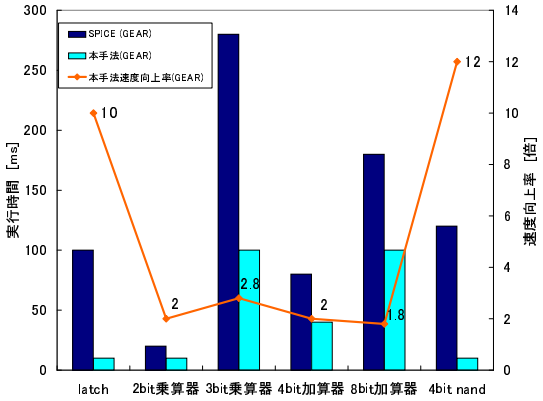

図 6: PC 上での実行時間及び SPICE3f5 に対する Gear 法の速度向上率

また、WS 上と PC 上での速度向上率は同じ傾向を示していることから、ここでは WS 上での速度向上率に着目して配列間接アクセスの除去の効果について述べる。

3.3 間接参照除去によるメモリアクセスオーバーヘッド削減効果

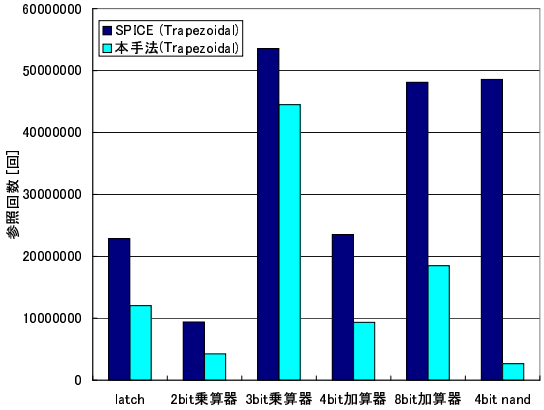

図 7: L1 D-Cache の参照回数 (Trapezidal 法)

図 7 は、SPICE3f5 の Trapezoidal 法、本手法の Trapezoidal 法についての L1 D-Cache の参照回数を示している。縦軸は参照回数、棒グラフは、左から SPICE3f5、本電子回路シミュレータの結果を示す。横軸は各評価回路を示し、左から latch、2bit 乗算器、3bit 乗算器、4bit 加算器、8bit 加算器、4bit nand 回路の結果を示す。

図 7 により、本電子回路シミュレータは SPICE3f5 に対して L1 D-Cache の参照回数を削減できていることを確認できる。4bit nand 回路は、SPICE3f5 の L1 D-Cache の参照回数が 4857 万回であり、本電子回路シミュレータの L1 D-Cache の参照回数が 165 万回である。これにより、L1 D-Cache 参照回数は、配列間接アクセスの除去を行なったことにより削減

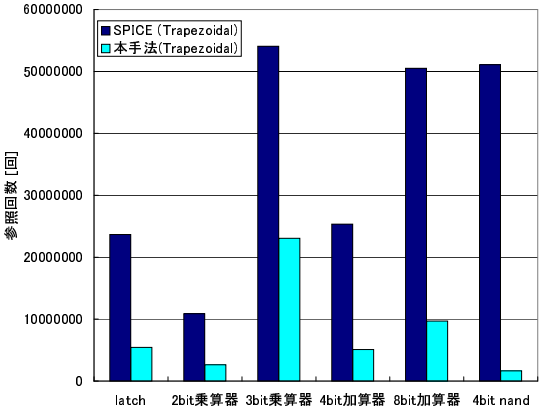

図 8: L1 I-Cache の参照回数 (Trapezoidal 法)

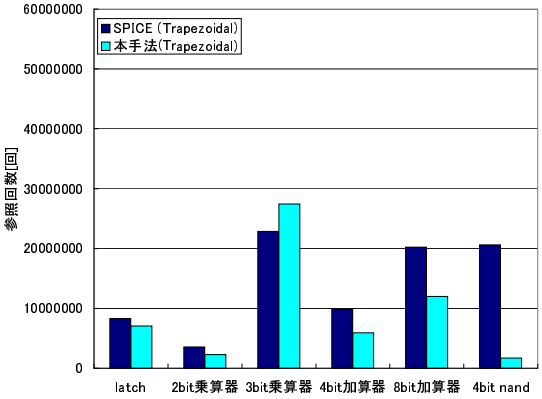

図 9: L2-unified Cache の参照回数 (Trapezoidal 法)

されていることがわかる。

図 8 は、SPICE3f5 の Trapezoidal 法、本電子回路シミュレータの Trapezoidal 法についての L1 I-Cache の参照回数を示している。縦軸は参照回数、棒グラフは、左から SPICE3f5、本電子回路シミュレータの結果を示す。横軸は各評価回路を示し、左から latch、2bit 乗算器、3bit 乗算器、4bit 加算器、8bit 加算器、4bit nand 回路の結果を示す。

図 8 により、本電子回路シミュレータは SPICE3f5 に対して L1 I-Cache の参照回数を削減できていることを確認できる。4bit nand 回路は、SPICE3f5 の L1 I-Cache の参照回数が 5108 万回であり、本電子回路シミュレータの L1 I-Cache の参照回数が 165 万回である。これにより、L1 I-Cache 参照回数は、定数伝搬及び Newton-Raphson 内のループ不変式を最外側の時間発展ループ外側へ移動するなどのコードリストラクチャリングを行なった結果として削減されていることがわかる。

図 9 は、SPICE3f5 の Trapezoidal 法、本電子回路シミュレータの Trapezoidal 法についての L2-unified Cache の参照回数を示している。縦軸は参照回数、棒グラフは、左から SPICE3f5、本電子回路シミュレータの結果を示す。横軸は各評価回路を示し、左から latch、2bit 乗算器、3bit 乗算器、4bit 加算器、8bit 加算器、4bit nand 回路の結果を示す。

図 9 により、本電子回路シミュレータは SPICE3f5

に対して L2-unified Cache の参照回数を削減できていることを確認できる。4bit nand 回路は、SPICE3f5 の L2-unified Cache の参照回数が 2062 万回であり、本電子回路シミュレータの L2-unified Cache の参照回数が 169 万回である。L2-unified Cache 参照回数は、配列間接アクセスの除去を行なったことにより削減されていることがわかる。

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 WS 及び PC 上で単一プロセッサを用いて実行した際に SPICE3f5 と比較して、シミュレーション時間は最大で 4bit nand 回路において

WS 上の Trapezoidal 法で 110 倍、GEAR 法で 40 倍、PC 上の Trapezoidal 法で 16 倍、GEAR 法で、12 倍高速化できる。

平均では、WS 上の Trapezoidal 法で 22 倍、GEAR 法で 11 倍、PC 上の Trapezoidal 法で 4.9 倍、GEAR 法で 5.1 倍高速化できる事を確認できた。

また，この高速化の効果が提案している間接参照を伴わないループフリーコード生成と定数伝搬及びコードモーションの効果であることがメモリアクセス回数の削減する等を通して論説できた。

今後、BSIM3(Berkeley Short-channel IGFET Model Version3) への本手法適用を行なっていく予定である。また、分割回路を行なった後の各回路間の粗粒度並列性やステートメントレベル間の近細粒度並列性など、様々な並列性を階層的に組み合わせて利用するマルチグレイン並列処理 [1),3)] の適用などを行なっていく予定である。

参考文献

[1] 笠原, “並列処理技術”, コロナ社 (1991).

[2] http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/

[3] 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.

[4] 早稲田大学, “シングルチップマルチプロセッサ”, 平成 11 年 特許出願第 363702 号, 平成 11 年 12 月 22 日.

[5] 早稲田大学, “電子回路シミュレータ”, 2000 年 特許出願第 055562 号, 平成 12 年 3 月 1 日.

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

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

[8] Yamamoto, F. and Takahashi, S.: Vectorized LU Decomposition Algorithms for Large-Scale Circuit Simulation, IEEE Trans. Computer Aided Design of Integrated Circuits and Systems, Vol.CAD-4, No.3, pp.232-239 (1985).

[9] Lynn Pointer: PERFECT REPORT:1, CSRD Rpt. No.896 (1989).

[10] White, J. and Sangiovanni-Vincentelli, A.: RELAX2: A Modified Waveform Relaxation Approach to the Simulation of MOS Digital Circuits, Proc. ISCAS*83, pp.756-759 (1983).

[11] Saleh, R. A., Newton, A. R.: Iterated Timing Analysis in SPLICE1, Proc. ICCAD*83, pp.139-140 (1983).

[12] 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

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

[14] Newton, A. R.: The Simulation of Large Scale Integrated Circuits, IEEE Trans. Circuits and Systems, Vol.CAS-26, pp.741-749 (1979).

[15] 鹿毛, “回路シミュレーション技術の動向”, 電子情報通信学会技術研究報告, VLD90-44(1990-9)

[16] L.Dagum and R.Menon, “OpenMP: An Industry-Standard API for Shared-Memory Programming”, IEEE Computatinal Science and Engineering., Vol.6, Num.9, pp.943-962, Jan 1998

[17] 鹿毛, “VLSI 回路シミュレーション”, 電気学会論文誌 C 分冊, No.6,1987

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

[19] I.N.Hajj, “Sparcity Considerations in Network Solution by Tearing”, IEEE Trans. Circuits and Syst., CAS-27, 5, pp.357-366, May 1980

[20] O.Tanabe.: LU Decompoaition on Distributed Memory Machines. IPSJ SIG Notes,pp.55-60 (1995).

[21] 笠原博徳:“最先端の自動並列化コンパイラ技術”, 情報処理学会誌, Vol.44, No.4, pp.384–392, Apr 2003

[22] 木村 啓二, 尾形 航, 岡本 雅巳, 笠原 博徳 : “シングルチップマルチプロセッサ上での近細粒度並列処理”, 情報処理学会論文誌, Vol.40, No.5, pp. 1924–1934, May 1999.