ハイパフォーマンス
コンピューティング 48－10
（１９９３．８．２０）

前処理つき Jacobi 法による並列回路 simulation

須田 礼仁
東京大学 理学部 情報科学科

計算機の進歩に伴い大規模回路 simulation の必要性は高まる一方であるが，回路 simulation は問題の構造が複雑で，また連立一次方程式の並列化が難しく，並列化に向かないと言われてきた．これに対し前処理つき緩和法が大規模論理回路の過渡解析において（並列環境でなくても）既存の方法に優る性能を出し得ることがわかってきた．今回はこれらのうち並列性の高い前処理つき Jacobi 法を AP1000 に実装し，並列化の効果と通信 overhead の大きさを評価する．また，task 割り付け，前処理，過渡解析のすべてを並列化した“全並列”回路 simulator の構想にもふれたい．

Parallel Circuit Simulation with the Preconditioned Jacobi Method

Reiji Suda
Department of Information Science, Faculty of Science, the University of Tokyo

The needs for large scale circuit simulation grow as the progress of computers. However circuit simulation has been thought as hard to parallelize because of the complexity of the problem structure and serialism of the existent linear solver. Recently the preconditioned relaxation methods are found to be effective in transient analysis of large scale circuits. The author implements the preconditioned Jacobi method, which has the highest parallelism, on AP1000, and evaluate its parallel performance and communication overheads. The plan of a “totally parallel” circuit simulator in which all of task allocation, preconditoining, and transient analysis are parallelized, is also discussed.

1. 前処理つき緩和法による回路 simulation

最近の計算機の進歩のともなって，より高速により大規模な回路の過渡解析を行なう必要性が増大してきている．そのため回路 simulation の並列化の要求も高まっているはずであるが，問題の構造が複雑であるのと，連立一次方程式の既存の解法が並列性が低いのとが障害となって，あまり効率的な並列化は実現されていない．

並列化のみならず sparsity の有効利用という点でも連立方程式の解法は反復法の方が望ましいのであるが，回路 simulation に出てくる連立一次方程式は random sparse, non-synmetric, nondefinite で，一般には反復解法は適用しにくい．特に小規模な回路では直接法に対し Markowitz-Tewarson の最適化が有効に働いて fill-in を減らし，その確実性にも伴われて直接法が広く用いられている．しかし大規模な回路になると単純な greedy な方法である Markowiz 法の性能は次第に悪くなり，simulation に時間がかかるようになる [3]．これに対し行列の sparsity をよりよく利用することのできる反復解法の回路 simulation への適用が研究されてきた [4][3][1]．これらの解法は残念ながらどのような回路，どのような精度の simulation でも確実に収束し直接法よりも速いとは言えないが，安定な論理回路の simulation にはほぼ適用可能で，大規模回路に対しては直接法に優る速度を提供しうる．

回路 simulation に適用された最初の反復解法は Gauss-Seidel 法であった [4]．Gauss-Seidel 法は係数が下三角行列に近いと速く収束するが，一定の条件を満たす CMOS 回路ではこれが完全な下三角行列になる．このような有利な状況においては Gauss-Seidel 法は非常に高速であるが，残念ながら適用範囲が非常に狭いのが欠点であった．つぎに [3] において完全 LU 分解前処理つき共役残差法が提案された．この方法はある時点での完全な LU 分解の結果を共役残差法の前処理に用いたもので，LU 分解が前処理として有効でなくなればその時点であらためて LU 分解をおこなうため，任意の回路に対して確実に解が得られることが特長であった．しかし非対称行列用の共役勾配法系の algorithm はいずれも一反復にかかる計算が多く，直接法をしのぐ速度的性能は観察されなかった．しかしまだ [3] に述べられている algorithm に対していくつかの改良が可能であり，今後の研究の進展が望まれる．最後に再び Gauss-Seidel 法が，今度は前処理を伴って提案された [1]．Gauss-Seidel などの古典的緩和法に前処理を適用するには前処理行列を陽な形で求めなければならない．この陽な形の前処理行列を求めるのに相当の時間がかかるため，緩和法には前処理はほとんどもちいてこられなかったのである．しかし過渡解析の前に一度だけ前処理行列をもとめてそれをずっと使うことにすれば，一度だけの前処理に必要な時間は過渡解析全体の時間に比べれば十分小さく抑えることができる．このような方法により前処理つき Gauss-Seidel および Jacobi 法が大規模回路の過渡解析において直接法をしのぐ性能を発揮することが確かめられた．しかし前処理を一度しか行なわないという制約のために非線形性が高く不安定な [1]回路においては収束しないこともありうるという欠点は避けられない．この点については研究は始まったばかりであり，どういう回路には単一の前処理行列が使えるのか，どの程度の頻度まで前処理の再計算が許されるのか，など研究課題は残っている．

1.1. 前処理つき緩和法の algorithm

緩和法は連立一次方程式 $Ax = b$ に対して一般に

$$x = x + \tilde{A}(b - Ax) \tag{1}$$

[1] 前処理行列は係数行列の逆行列に近いほど良い．一般的には回路が不安定であれば係数行列の条件数が大きくなり，係数行列の変化に対して逆行列が大きく変化することになる．このため不安定な回路においては固定された行列がよい前処理行列になりにくい．

のように表現することができ，Gauss-Seidel 法では $\tilde{A} = L^{-1}$ (L は A の下三角成分)，Jacobi 法では $\tilde{A} = D^{-1}$ (D は A の対角成分)，Richardson 法では $\tilde{A} = \lambda I$ (λ は定数) となる．これらの緩和法は係数行列 A が単位行列 I に近いほど速く収束する．

前処理つき緩和法は連立一次方程式 $Ax = b$ に対し前処理行列 P, Q をもちいて

$$PAQy = Pb \tag{2}$$

という問題を構成し，これを緩和法で解くものである．解 x は $x = Qy$ として求められる．実効的な係数行列は PAQ であるので $P = A^{-1}, Q = I$, $P = I, Q = A^{-1}$, $P = L^{-1}, Q = U^{-1}$ (ただし $A = LU$) などの組合せにおいては緩和法は一反復で収束し，またこれに近ければ非常に速く収束することになる．

前処理つき緩和法はたとえ係数行列 A が疎であっても前処理行列 P, Q が密になると計算量が多くなってしまう．しかし (このことは数値計算の世界でもあまり知られていないが) $P = L^{-1}, Q = U^{-1}$ ととると P, Q は A の sparsity をある程度継承して疎行列となり，計算量を抑えることができる．

反復法は要求される精度が低ければ非常に少ない反復回数で解を得ることができる．回路 simulation のなかの連立一次方程式は Newton 法のなかで用いられるが，Newton 法は反復のはじめの方では高い精度の計算は必要なく，収束直前になって急激に要求精度が上がるという性質がある．そこで Newton 法が要求する精度をうまく評価して連立一次方程式の収束条件を最適に設定することは非常に効果的である．

1.2. 前処理つき緩和法の serial machine 上の性能

[1] で報告されている前処理つき緩和法の serial machine 上での性能を引用する．図 1, 2 は回路の規模に対し

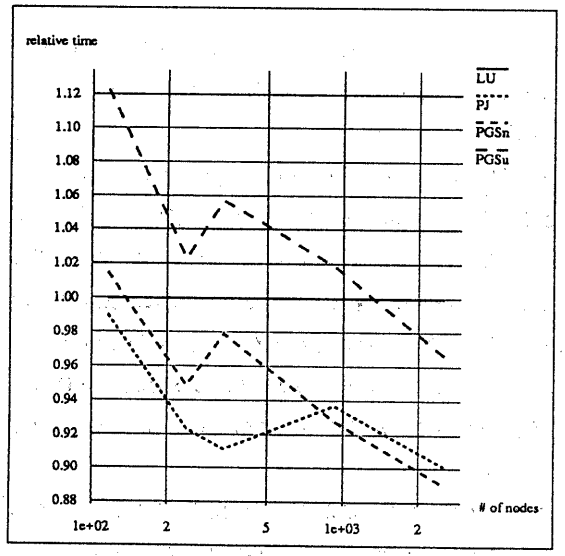

図 1. 前処理つき緩和法の性能: $\epsilon = 10^{-2}$

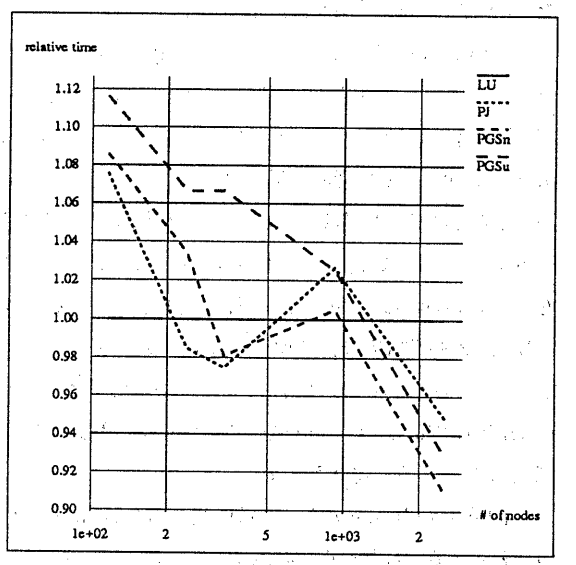

図 2. 前処理つき緩和法の性能: $\epsilon = 10^{-3}$

て simulation の所要時間を，直接法での所要時間との比の形で plot したものである．対象とした回路は QFP [5] の half adder, full adder, 2 bit adder, 4 bit adder の 4 つであり，simulation の精度を図 1 では 10^{-2}, 図 2 で

は 10^{-3} と設定してある [2]. これらの結果から (1) 大規模な回路ほど反復法に有利である, (2) 要求精度が低いほど反復法に有利である, (3) 直接法と反復法の速度の差は, この範囲では 10% 程度である, ということがわかる. このうち (1) の大規模な回路ほど反復法に有利であることは [3] でも報告されており, Markowiz 法の限界を示すものとして理解されている. また (2) は反復解法の性質から当然の結果といえる. (3) については緩和法の平均反復回数が 2-3 回であることから予想される結果と良く合っている.

このように前処理つき緩和法は serial machine 上でも直接法に匹敵する速度を出し得ることがわかっている. しかも前処理つき Jacobi 法は本質的に serial な直接法と大きく異なり高い自然な並列性を持っている. このことから大規模回路の並列 simulation に前処理つき Jacobi 法が有効であることは想像に難くない. [2] では model 化された parallel machine 上での性能が前評価されているが, 今回の研究では AP1000 に実装して評価をおこなった.

2. 前処理つき Jacobi 法の並列化

回路の過渡解析の基礎方程式は連立非線形常微分方程式であり, 一般に 3 重の algorithm で実現される. まず, 常微分方程式を後退 Euler 法や Gear 法を用いて時間刻みごとの非線形連立方程式に帰着させる. つぎに, 非線形方程式を Newton 法やその改良をもちいて反復的に解く. その際, Jacobian を係数行列とする連立一次方程式を繰り返し解く. この 3 重の algorithm のうち Euler 法や Newton 法は owner-computes rule をもちいれば自然に並列化でき, また連立一次方程式も前処理つき Jacobi 法により同様な並列化が可能となる. しかしながら前処理つき Jacobi 法は Euler 法や Newton 法に比べ通信の要求が大きく, 並列効果は相対的に低い. しかしこのことはおのおのの node の変数 (電圧など) の変化をほかの node に情報として伝えているのは連立一次方程式の求解の step にほかならないことから, どのような algorithm をもちいても基本的にはかわらないのである (ただし, どれほど効率的に情報が伝わるかは algorithm によるが).

回路 simulation においては code generation という技法が広くもちいられている. 通常の回路の model 化では simulation 中に回路の構造や parameter が変化することはなく, 変数の値や反復回数を除いて「同一」の計算が繰り返し実行される. そこで回路の構造や parameter, LU 分解の手順や非零 patern を組み込んでしまった program を生成して過渡解析の際の index などの計算を極力排除しようとする方法が code generation である. このような方法は整数計算能力に比べて浮動小数点数の計算能力が強い machine では特におおきな効果を示す. しかし workstation など浮動小数点数計算が弱い machine ではそれほどの効果はなく, むしろ instruction の locality を下げて cache の mishit を呼び, 性能を下げることもありうる. 今回の実装対象とした AP1000 では浮動小数点数計算能力が整数計算能力に比べて低く, code generation の効果は高くないと期待されるが, 一応 code generation をもちい, 各 processor に別々の task を割り付けることにした ([6] 参照). しかし AP1000 は本質的に SPMD machine であり, system 全体でわずか 31 種類の task しか生成することができい. このため processor 台数は 30 以下にせざるを得なかった.

回路 simulation においては回路の構造が simulation の途中で変わることはなく, どの task がどの data をいつ必要とするかは前処理の時点で決定される (code generation をもちいれば, compile 時に決まるといってよい).

---

[2] Model そのものの誤差や製造に伴う parameter の誤差を考えればこれらの精度は論理回路の動作の検証としては十分な精度として認め得るものである. しかし, switching time の限界や設計 margin など device の理論的特性を詳細に調べるには不十分であろう.

このため通信 pattern は複雑であるものの simulation 中に request などを必要とせず, data が生成されしだい必要となる processor に送りつけてしまうことが可能であって, 通信の latency をかなり隠すことができる. 特に行列と vector との積などにおいて, 自分がもっている data をつかって計算できる部分と他の processor から送られてくる data を必要とする部分とを分離して, 通信を行なっている間に自分が持っている data でできる範囲の計算をおこなうことは, 通信の遅延の影響を隠蔽するのに有効であった.

なお, task allocation は task graph (無向グラフ) をある方法で再帰的に 2 分割してゆく algorithm である. 計算量は $O(n^2)$ で, processor 台数に依存しない. また, かなり高い並列性をもっており, 並列実装も検討中である. この algorithm は simulated annealing などにくらべればそれほどよい結果を出すとはかぎらないが, 次節で data を示すように一応使える結果を出してくれる. また, mesh task を mesh processor に割り付ける場合には, 「自明な」最適解が存在するという条件のもとでその最適解をあたえるという特長がある. 計算量の $O(n^2)$ は task allocation の algorithm としてはかなり少ない方であるが, n が万の order になることをかんがえれば実用上は限界といえよう. Algorithm の詳細についてはここでは省略する.

3. 実験結果と考察

前節で説明した方法で前処理つき Jacobi 法をもちいた回路の過渡解析 program を並列計算機 AP1000 上に実装した. Sample とした回路は QFP による half adder, full adder, 2 bit adder, 4 bit adder の 4 種類である. Carry look ahead をもちいているため gate 間の結合はやや複雑である. まず, 所要時間の graph を図 3 に示す. 水平な線は single processor においてもっとも速かった algorithm (前処理つき Jacobi 法ではない !!) の所要時間である. まず, single processor の性能を追い抜くには 2-3 台必要で, 台数効果はあまり高くないことがわかる. このことは 1 節で述べたこと (前処理つき Jacobi 法は直接法に匹敵する性能をしめす) と矛盾している訳だが, これは single processor 用の前処理つき Jacobi 法と並列用のそれとは同一ではないことによっている. 実際まだ program に改良の余地はあり, 今後の工夫次第でこのことは改善されるであろう.

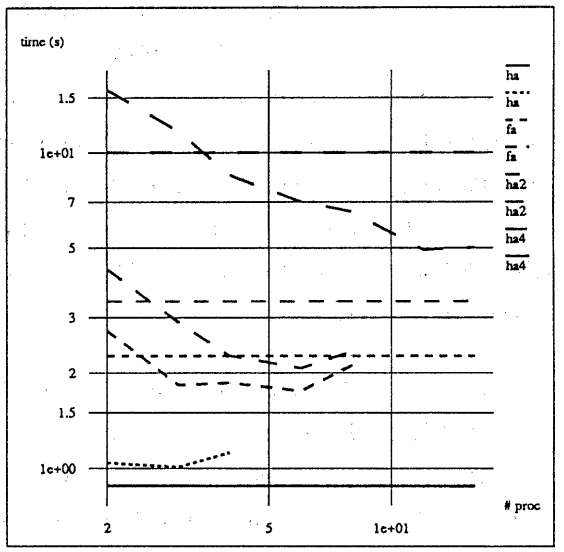

図 3. Simulation 時間

一方並列性能同士の比較をすれば, processor 台数がある値になると台数を増やしても所要時間が増大するようになる. これは一定の並列性をもつ問題を processor 台数を変えて解く時に必ずあらわれる現象で, それがいつおこるかは問題自体の他, task allocation の方法や machine の通信性能に依存する. 今回の条件ではほぼ 1 processor あたり 40-75 node (実際には 80% は線形 node であるため静的に解かれており, 実際に simulation 中に解く必要がある node 数は 8-15 node 程度である) のところで所要時間が逆転したが, task allocation の方法や通信性能が改善されればもっと高い並列度が利用でき, 通信性能がもっと悪い環境ではもっと低い並列度で我慢しなければならないことになる.

このことをもう少し定量的に論じるために, 図4 に processor あたりの node 数に対して通信の overhead 率 (通信時間 / task 時間) を log-log で plot する. 傾きはおよそ −1.6 であり, 偏微分方程式の red-black Gauss-Seidel 法による求解 (2 次元で −1/2, 3 次元で −2/3) などよりもはるかに急である. このことは processor 台数を増やすと急激に通信の overhead が増えることを意味し, 前処理つき Jacobi 法をもちいても回路 simulation の問題が並列化しにくいことを明らかにしている. しかし逆に見ると, processor 数を少し減らせば通信の overhead はぐんと減ることになり, 少々通信性能が悪い machine でもそれなりに並列 simulation できることがわかる. たとえば通信性能が AP1000 よりも倍悪い machine でも 1 processor あたり 60-110 node をわりあてればよい. 逆に AP1000 よりも通信が倍速い machine でも並列度はせいぜい 1.5 倍になるだけである.

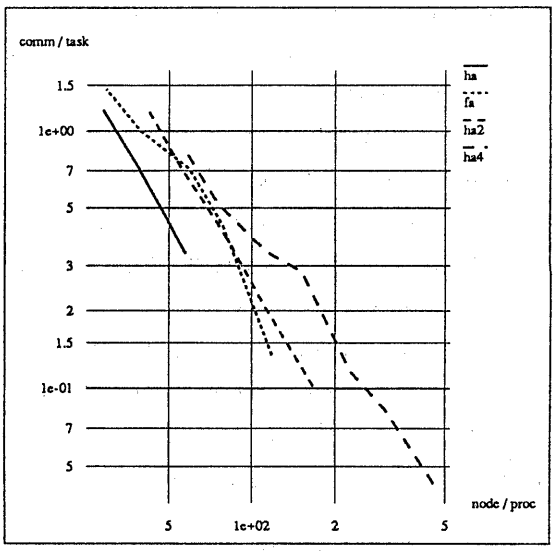

図 4. 通信 overhead

もっともこれらの評価値は現在の前処理法と task allocation に依存した値である. 特に前処理行列は並列化を考慮にいれずに作っているのでなんらかの工夫を加えることにより並列性が伸びる可能性は十分にある. また, 現在は絶対速度を得るために計算量を減らすさまざまな工夫を加えている. これらを除いて program を改悪することにより通信の overhead を相対的に縮小して「並列効果」をのばすことも可能であるが, それは決して全体の所要時間を減らすことにはならない. 著者は並列性が低くなることを承知の上で計算量を減らす工夫を適用し, 絶対時間を縮小している.

4. “全並列” 回路 simulator, sparta の構想

今回の研究では task allocation と前処理を single processor 上で行なって各 processor のための program を生成したが, 実験の所要時間のほとんどは compile 時間で, それに task allocation, 前処理が続き, 実際に問題となるはずの simulation の所要時間は (反復回数を少なく設定したこともあるが) 全体のごくわずかでしかなかった. 今回の実験は前処理つき Jacobi 法の有効性を確かめる目的であったからそれでもよかったが, このままでは実用にはとても使えない. そこで, task allocation, preconditioning, transient analysis のすべてを並列におこなう回路 simulator, sparta を計画している. この節では予定している sparta の概要を述べたい.

まず task allocation には今回もちいたものと基本的には同じ algorithm をもちいる. この algorithm は graph の分割のある条件を満たす候補を数え挙げてそのうちで最適なものを選ぶという方法であるため, 数え上げを並列化すればほとんど通信を必要とせず, 高い並列性を持ち, 実行に先立つ task allocation を必要としない (random でよい). 2 節でふれたような mesh-to-mesh の場合の最適性を保持するには若干の serialism を残す必要があるが, (もともとほとんどない) 通信の overhead を除けば並列化効率は 50% を下ることはないことが理論的にわかっている. ただ現在の algorithm は graph が小さくなると次第にうまく分割できなくなってくるので algorithm 自体の改良が必要かも知れない.

次に行列の LU 分解をおこなわなければならない．密行列の LU 分解の並列化は話が簡単であるが，疎行列ではまったく状況が異なる．ひとつひとつの掃き出しの計算量がきわめて少なく，そもそもまったく関係しない processor がかなりある．そこでひとつひとつの掃き出しを並列化するよりも，いくつもの掃き出しを並列におこなうことが必要になってくる．さらに processor 間で独立に行なうことのできる掃き出しが尽きてもいきなり全 processor level で掃き出すのではなく，processor を cluster にして cluster 間で並列に掃き出し，徐々に cluster を融合してゆくことで最大限の並列性をひきださなければならない．また，過渡解析の際の通信量は LU 分解の pivoting に大きく依存する．結果的に simulation 中の通信量が減るような工夫を組み入れる予定である．

次に前処理行列をもとめる．これは逆行列を行ごとに並列にもとめることができるので並列化自体は容易であるが，もとの行列を各 processor に送らなければならないので通信量がかかる．この通信をなんらかの工夫で減らせるかどうかが効率の鍵になる．前処理行列がもとまったら通信 pattern の決定など若干の準備をしてから過渡解析をおこなう．過渡解析の結果は simulation と同時に表示して，並列化の速度的効果を視覚的に理解できるようにする予定である．また，余裕があれば直接法による解法も同時に実装して前処理つき Jacobi 法と性能を比較したい．

5. まとめ

前処理つき Jacobi 法を code generation の手法をもちいて並列計算機 AP1000 上に実装し，その並列性を検証した．その結果 single processor 上での最適 algorithm の所要時間を上回る性能を 2-3 台から示した．また 1 processor あたりおよそ 40-75 node (dynamic には 8-15 node) まで速度の向上が得られた．通信 overhead は processor あたりの node 数のおよそ -1.5 乗で増加し，偏微分方程式などの簡単な問題にくらべはるかに並列化が難しいことが明らかとなった．これらの値は今後 task allocation や前処理法の改良によって改善されるだろう．

Task allocation, preconditioning, transient analysis のすべてを並列に実行できる "全並列" 回路 simulator, sparta の構想も述べた．sparta においては現在の algorithm の並列化のみならず，algorithm 自体の改良も考えている．また，結果の実時間表示と直接法との比較も行ないたい．

6. 参考文献

[1] 須田礼仁, "前処理つき Gauss-Seidel 反復法による Josephson 論理 gate の回路 simulation," 第 21 回数値解析シンポジウム予稿集, 1992, pp. 1–4.

[2] 須田礼仁, 小柳義夫, "前処理つき Gauss-Seidel 法による Josephson junction 回路 simulation," 情報処理学会研究報告 92-NA-42, Aug. 1992, pp. 1–8.

[3] 山本富士男, 梅谷征雄, 高橋栄, "大規模回路シミュレーションに対する完全 LU 分解付き共役残差法とその適用性評価," 情報処理学会論文誌, Vol. 27, No. 8, Aug. 1986, pp. 774–782.

[4] Newton, A. R., and A. Sangiovanni-Vincentelli, "Relaxation-based Electrical Simulation," *IEEE Trans. on CAD*, Vol. CAD-3, No. 4, Apr. 1984, pp. 308–331.

[5] W. Hioe and E. Goto, *Quantum Flux Parametron*, Singapore: World Scientific, 1991.

[6] 前川仁孝 他, "近細粒度タスクを用いた電子回路シミュレーションの並列処理," JSPP '92 論文集, Jun. 1992, pp. 453–460.