電子回路シミュレーションの粗粒度/近細粒度階層的並列手法*

2J－9

伊藤 泰樹, 前川 仁孝, 高井 峰生, 西川 健, 笠原 博徳†

早稲田大学理工学部‡

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に要する時間の短縮は重要な課題の一つである。従来より直接法を用いた電子回路シミュレーションの並列処理では、回路分割により粗粒度タスクを生成し、分割回路をプロセッサに割り当てて粗粒度並列処理を行なう方法が研究されてきた[1, 2]。本稿では、回路の自動分割を行ない、分割回路間での粗粒度並列処理と、分割回路内における直接解法のステートメントレベル近細粒度並列処理を、階層的に組み合わせる並列処理手法を提案する。

2 回路シミュレーションの階層的並列手法

本節では、電子回路シミュレーションの階層的並列処理手法について述べる。

一般に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、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、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クラウト法を組み合わせた手法がよく用いられる。

本研究では、このような直接法を用いた電子回路シミュレーションを一般ユーザで簡単に並列処理を行なえる電子回路専用目的コンパイラの開発を行なった。本専用目的コンパイラはユーザが、SPICE形式の回路リストを入力することにより、自動的に、回路分割、タスク生成、タスクスケジューリングを行ない、並列マシンコードを生成する。以下では、回路分割, タスク生成, タスクスケジューリングの３つの手法について述べる。

2.1 回路の自動分割手法

本節では、回路の粗粒度並列性を利用するための回路の自動分割手法について述べる。

従来より、回路分割の代表的な手法の一つであるKernighanとLinの考案したアルゴリズム(KL法)[3]を基に様々な方法が提案されているが、本コンパイラではその中でも代表的なFudicciaとMattheyesが考案したアルゴリズム(FM法)[4]を採用した。このアルゴリズムはKL法を改良したもので、２組のブロックの間でセルを交換するのではなくセルを１つずつ移動させることにより分割を最適化する。また、移動候補を選ぶ際にはcutsetの増減(gain)を計算してgainでソートしたGain Vectorと呼ばれる配列を用い、KL法に比べ計算量を大幅に削減している。セルの数がnとすると、KL法では候補を選ぶ際の時間計算量が$O(n^2)$となるのに対し、FM法では$O(n)$であるが、収束回数が増えるために$O(n^{1.2})$から$O(n^{1.5})$程度の計算量になる。

2.2 粗粒度及び近細粒度タスクの生成

本節では回路分割を適用した際のシミュレーション計算手法と、粗粒度並列処理の単位となる粗粒度タスクの生成手法について述べる。

以下に、本コンパイラで採用したHaji等によって提案された節点分割を行なった場合の計算手法について説明する。与えられた回路を解くための連立一次方程式を生成する際に節点分割法で回路分割し、分割点を縁側に持っていくことにより図1のような縁付ブロック対角行列になる。

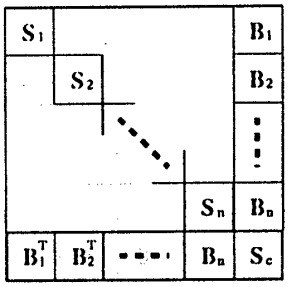

図 1: 縁付ブロック対角(BBD)行列

このような行列をクラウト法を用いて解くことを考えると、$S_k(k=1,2,\cdots,n)$は並列にLU分解でき, またS_kの計算が終了すれば、B_k, $B_k^T(k=1,2,\cdots,n)$も並列にLU分解することができる。また、S_cのLU分解は、S_k, B_k, B_k^TのLU分解が全て終了した段階で、計算可能になる。

また、前進代入部における計算は、S_k, B_k, B_k^Tを一つのグループとする部分行列を考えると、S_cを除いて全て他の部分行列の計算結果を必要としないので、S_c以外は並列に計算できる。S_cについては、自分のブロック以外の全ての解を必要とするので、各部分行列の前進代入の計算が全て終り次第、計算可能となる。

最後に行なわれる後退代入部の計算は、S_cに属する解の計算が終らないと、各部分行列の計算ができない。そのため、始めにS_cに関する計算を行なった後に、この値を他の部分行列に転送することにより、残りの各部分行列を並列に計算できるようになる。

これらの各処理を粗粒度タスクとして定義し、タスク融合を行なうと図2のようなタスクグラフを得る。これにより、S_k, B_k, B_k^TのLU分解と前進代入をn個の粗粒度タスクで並列に処理し、その後S_cの部分のLU分解、前進代入、後退代入を計算し、最後に各部分行列の後退代入をn個の粗粒度タスクで処理する。

図1の中で各部分行列の結合行列であるS_cは、通常の回路行列より要素が密であり(70～80%)[2]、分割数を増やすと粗粒度並列処理できないS_cの部分の計算負荷がかなり大きくなる。そのため従来の回路分割による粗粒度並列処理のみを用いた手法の場合、多数のプロセッサ上で並列処理を行なう場合は

*Hierarchal Parallel Procesing of Electronic Circuit Simulation with Coarse and Near Fine Grain Tasks

†Taiki ITOH, Yoshitaka MAEKAWA, Mineo TAKAI, Takeshi NISHIKAWA, Hironori KASAHARA

‡School of Science and Engineering, Waseda University

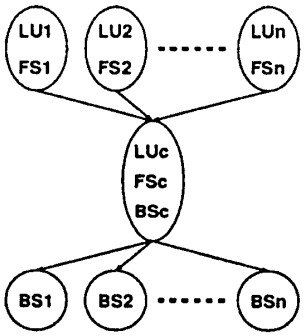

図 2: 粗粒度タスクグラフ

回路の分割数を増やさねばならず、並列効果を得ることが難しかった[2]。そこで本手法では、分割した回路内でステートメントレベルの近細粒度タスクを生成し、分割回路間の粗粒度並列処理と分割回路内の近細粒度並列処理とを階層的に組合せて並列処理する手法を提案する。

2.3 階層的スケジューリング

粗粒度タスク及び近細粒度タスクのプロセッサへの割当と実行順序を決定するスケジューリング問題は、強 NP 困難な問題であることが知られている。本手法では、スケジューリング手法にデータ転送時間を考慮したヒューリスティックスケジューリングアルゴリズムである CP/DT/MISF 法を用いる[5]。

階層的なスケジューリングを行なう際には、まず回路分割より得られる各粗粒度タスク内をステートメントレベルの近細粒度タスクに分割して近細粒度タスクグラフを生成し、CP/DT/MISF 法を用いて近細粒度タスクのスケジューリングを行なう。次にスケジューリングを行なった結果得られる近細粒度タスク集合の推定実行時間を粗粒度タスクの推定実行時間とし、粗粒度タスク間のデータ依存関係より、近細粒度タスクグラフと同様な方法で粗粒度タスクグラフ (マクロタスクグラフ) を生成し、CP/DT/MISF 法を用いて粗粒度タスクレベルでスケジューリングを行なう。最後にこの粗粒度レベルでのスケジューリング結果と各粗粒度タスク内の近細粒度タスクのスケジューリング結果を組み合わせることにより最終的なスケジュールを生成する。

3 電子回路シミュレーションの階層並列化手法の性能評価

本節では、電子回路シミュレーションの階層的並列処理手法の性能を OSCAR シミュレータ上での評価した結果について述べる。

3.1 OSCAR のアーキテクチャ

今回評価に用いたマルチプロセッサシステム OSCAR は、16 台の PE と集中共有メモリ (CSM) が 3 本のバスで接続されている。各 PE は、5MFLOPS の 32bit カスタム RISC processor、ローカルメモリ、分散共有メモリ (DSM) から成っている。この DPM を使ったデータ転送は 1PE 対 1PE、1PE 対全 PE の転送モードがあり、いずれも CM を介するより高速にデータを転送できる。

3.2 性能評価

本節では、小規模加算器回路の過渡解析を例に、階層的並列手法を、粗粒度あるいは近細粒度並列処理のみの場合と比較することにより性能評価を行なう。表 1 は 4bit 加算器の回路について、過渡解析に要する時間を評価したものである。表中、分割数はシミュレーションを行なう際の回路の分割数、PC 台数は使用するプロセッサクラスタ台数、PE 台数はプロセッサクラスタ内で使用するプロセッサ台数を表しており、全体で使用される総プロセッサ台数は、PC 台数×PE 台数となる。

表 1では総プロセッサ台数を 1,2,4,8 台と増やしていった時の実行時間は、粗粒度並列処理 (PE 台数を 1 に固定し、PC 台数を 1,2,4,8 台と増やしていく) では 611.1, 361.7, 200.6, 155.6 ms となり、近細粒度並列処理 (PC 台数を 1、PE 台数を 1,2,4,8 台と増やしていく) では 611.1, 409.7, 244.3, 149.8 ms となっている。このことより 4 分割までは粗粒度並列処理が近細粒度並列処理を上回っているが、8 分割では近細粒度並列性が粗粒度並列性を上回ることにが分かり、これから現在の回路分割技術を用いた粗粒度並列処理では扱う回路に応じ並列性に限界があることが分かる。また、回路を 4 分割して、分割回路内をさらに 2 台のプロセッサを用いて近細粒度並列処理を行なう階層的並列処理を行なった場合、130.2 ms となり同じ PE 台数が 8 台の粗粒度のみ及び近細粒度のみの場合と比較して短い処理時間が得られることが確認できた。この結果より回路を分割による粗粒度並列処理と分割回路内の近細粒度並列処理を階層的に行なう本手法の有効性が確認された。

表 1: Parallel Processing Time of 4bits Adder Circuit

並列化手法	分割数	PC 台数	PE 台数	実行時間 [ms]
—	1	1	1	611.1
近細粒度	1	1	2	409.7
	1	1	4	244.3
	1	1	8	149.8
粗粒度	2	2	1	361.7
	4	4	1	200.6
	8	8	1	155.6
階層	4	4	2	130.2

4 まとめ

本論文では、回路分割を行ない分割回路をプロセッサクラスタに割り当てる粗粒度並列処理と、分割した回路内の計算をステートメントレベルで並列処理する近細粒度並列処理を組み合わせた階層的並列処理手法の提案を行なった。また、提案した手法の有効性を確認するために OSCAR シミュレータ上で評価を行なった。評価の結果、分割した回路に適用した粗粒度並列処理手法と分割回路内に適用した近細粒度並列処理手法を階層的に組合せることにより、従来の粗粒度並列処理のみで並列処理を行なった場合と比較して、より高い並列効果が得られるのが確認できた。

今後は、より大規模な回路に適用し提案する階層的並列化手法の性能を評価していく予定である。

本研究の一部は、文部省科学研究費 (一般研究 (b)05452354 (c)05680284) により行なわれた。

参考文献

[1] 中田ほか, "並列回路シミュレーションマシン Cenju", 情報処理学会誌, Vol31, No.5, pp.593-601, 1990

[2] 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)

[3] B.W.Kernighan, S.Lin, "An Efficient Heuristic Procedure for Partiotioning Graphs", Bell Syst. Tech. J., Vol.49, pp.291-307, Feb.1970

[4] C.M.Fiduccia, R.M.Mattheyses, "A Linear-time Heuristic for Improving Network Partiotions", Proc. 19th Design Automat. Conf., pp.175-181, 1982

[5] H.Kasahara, H.Honda, S.Narita, "Parallel Processing of Near Fine Grain Tasks Using Scheduling on OSCAR(Optimally Scheduled Advanced Multiprocessor)", Proc. IEEE ACM. Conf. Supercomputing'90, pp.856-864, 1990