3D−2

OSCAR上での直接法を用いた回路シミュレーションの並列処理

前川 仁孝　田村 光雄　Wichian Premchaiswadi　笠原 博徳　成田 誠之助

早稲田大学理工学部

1. はじめに

本稿では、直接解法を用いた電子回路シミュレーション[1][2]の並列処理手法を提案する。ここでは、電子回路の非線形連立微分方程式を可変ステップ可変次数のBDF法[6]を用いて求解する。また、BDF法の中で使われるニュートンラフソン法では、スパース係数行列を持つ連立方程式の求解が必用となり、この連立方程式の求解が電子回路シミュレーションの高速化のためのネックとなっていた。本手法では、この連立方程式求解におけるLU分解部分を、コード生成法[2]を用いてループフリーコード[2][3]で表現し、これを近細粒度タスクに分割して並列処理を行っている。また、本手法はマルチプロセッサシステムOSCAR[4]上でインプリメントされており、本稿ではその性能評価についても述べる。

2. 電子回路解析手順とタスク生成

今回の電子回路解析では、電子回路の動特性解析を接点解析法を用いて非線形連立微分方程式

$$f(\dot{x}, x, t) = 0$$
$$x(t=0) = x_0$$

ただし、xは解のn次元ベクトル

を生成し、BDF法を用いて求解するという手順をとる。

以下では、この手順の詳細とタスク生成手法について述べる。

STEP 1　BDF法の予測子の値によって、次に計算する時刻$time_{t+1}$での変数の値を予測する。
予測子x_{t+1}'は次のように求めることができる。

$$x_{t+1}' = \beta_1 x_t + \cdots + \beta_{k+1} x_{t-k}$$

この時、各β_iを求める実行中に変化する積分ステップ内の関数と成っており、その計算式は、コンパイル時にシンボリックに生成する。ここでは、各β_iの計算を1つのタスクと定義する。またn次元ベクタである予測子x_{t+1}'の値の計算では、各要素の計算ごとに1つのタスクと定義する。

STEP 2　BDF法で求められた微分値を用いて、非線形微分方程式を非線形方程式に置き換える。

$$\dot{x}_{t+1} = -(\alpha_0 x_{t+1} + \cdots + \alpha_k x_{t-k+1})/h$$

ここでは係数を求めなければならないが、この計算にはβと同様な手法が適用でき、タスク生成もSTEP 1と同様に行われる。

STEP 3　非線形方程式ををニュートンラフソン近似を用いて、線形方程式に置き換える。また、ニュートンラフソン法の初期値には予測子の値を用いる。

次式のニュートンラフソン法の計算

$$J x^{k+1} = J x^k - f(x^k)$$

Jはヤコビ行列

におけるヤコビ行列の計算では、値が定数になる部分は実行時の計算は必要ないが、変数の部分については、毎回、行列の要素の値を計算する必要がある。ここでは、ヤコビ行列の各要素ごとの計算を1つのタスクと定義する。

STEP 4　線形方程式を解く。このスパース線形方程式の求解ではLU分解、前進代入、後退代入を行う。

本並列処理手法では、この求解のためにコード生成法を用いループフリーコードを生成し、それを並列処理する。このループフリーコードは、従来回路シミュレーションの高速化のためにネックとなっていたスパース行列の直接解法を用いた求解を、通常の Fortranプログラムによる求解より数十倍も高速化することが知られている[3]。ここでは、この非零要素の計算のみを列挙したループフリーコードすなわち条件分岐、繰り返し等の制御構造を持たない算術代入分の集合を、複数浮動小数点演算レベルの近細粒度タスクに分割して並列処理することにより、さらに高速化する。

STEP 5　ニュートンラフソン法が収束すればSTEP 6へ、収束しなければSTEP 3へ。

ここで収束の判断を行うが、まずn次元ベクタである変数x_{t+1}の収束計算を行うのに、各変数の変化分の計算を1つのタスクとして定義する。最後に、収束判定を1つのタスクと定義する。

STEP 6　シミュレーション終了時刻まで計算したら終了。そうでなければ次の積分ステップ幅、次数を計算して、STEP 1へ。

ここで、積分ステップ、次数の計算を行う前に予測子より得られた値(x_{t+1}')と実際の計算結果(x_{t+1})との差を計算する。この各変数に対

Paralell Processing of Circuit Simulation Using a direct method on OSCAR

Yoshitaka MAEKAWA, Mitsuo TAMURA, Wichian PREMCHAISWADI, Hironori KASAHARA, Seinosuke NARITA

WASEDA Univ.

する誤差の評価計算を1つのタスクと定義する。最後に、各変数の誤差の評価結果を用いて、積分ステップ、次数の計算を行う。この計算も1つのタスクとして定義する。

3. タスクグラフ生成

生成されたタスク間には、ループのような制御構造、フロー依存,出力依存,逆依存のようなデータ依存が存在する。ここでは細粒度タスクの並列処理を行うので、データ転送、同期、実行時スケジューリングオーバーヘッドを最小化できるスタティックスケジューリングを用いる。スタティックスケジューリングを使用する場合、タスク間に条件分岐が存在すると効率的なスケジューリングが行いにくいので、本手法では、前述の手順のSTEP 1,2の部分（外側ループである時間発展ループ内）とSTEP 3,4,5（内側ループであるニュートンラフソンループ内）とSTEP 6（時間発展ループ内）の3つの部分に分けてスケジューリングを行う。この3つの部分の内のデータ依存は、タスクグラフ[5]と呼ばれる無サイクル有向グラフで表される。

4. スタティック・スケジューリング・アルゴリズム

これらのタスクをプロセッサにスケジューリングする問題は強NP困難であるので、スケジューリングに要する時間とスケジュール結果の質を考慮にいれ、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CP/DT/MISF[5]を適用する。

5. 最適化並列マシンコード生成

実際のマルチプロセッサシステム上で、効率良い並列処理を行うために、スケジューリング結果を用い、最適化された並列マシンコードを生成する。このマシンコード生成は、使用するマルチプロセッサシステムのアーキテクチャに依存するので、以下では今回使用したOSCAR[4](図1)を例に取り議論を進める。

スタティックスケジューリングの結果より得られる情報は、各PE上で実行すべきタスクの集合、同一PE上で実行されるべきタスクの実行順序、他のPEに割り当てられている先行タスクからデータが送られてくるまでの推定待ち時間、どのタスク間で同期をとるべきかなどである。以上の情報を十分に活用することにより、同期及びデータ転送オーバーヘッドを含めた並列処理時間を最小化にするマシンコードを生成することができる。

また、回路シミュレーションを近細粒度タスクレベルで並列処理をすると、数千から数万以上のタスク数が生成されるため、従来のマルチプロセッサ上での細粒度並列処理のように、データ毎あるいはタスク毎に同期フラグを割り当てると、同期のために非常に大きなメモリが消費される。このため、本手法では、各PE上にPE台数分のみのフラグ領域を各PE上の分散共有メモリに用意し、各PEが何番目のタスクまで実行したかを示すフラグの値を、そのプロセッサからのデータを待っているプロセッサ上のフラグ領域に書き込む。これにより、プロセッサ間のデータ同期制御は、タスク数によらずO(m),(mはプロセッサ台数)のメモリで実現できる。

6. 性能評価

本手法の性能を評価するために、2ビットの加算器をNAND回路を用いて構成した小回路について、回路の過渡解析に要する処理時間を測定する。この結果、PEが1台の時の処理時間が16.2s、4台で6.07s、8台で3.55sと短縮されることが確認された。

7. おわりに

本稿では、直接解法を用いた電子回路シミュレーションの並列処理手法を提案し、その有効性をOSCAR上で評価した。評価の結果、近細粒度タスクの利用により、従来ネックとなっていたスパース行列求解部分も含め、電子回路のシミュレーション時間をプロセッサ台数の増加と共に短縮できることが確かめられた。

参考文献

[1] 中田,田辺,梶原,松下,小野塚,浅野,小池,並列回路シミュレーションマシン Cenju,情処論,Vol.31 No.5,pp.593-601,1990年5月

[2] P.Antognetti and G.Massobrio,"Semiconductor Device Modeling with SPICE",McGraw-Hill.(1988)

[3] Y.Fukui, H.Yoshida and S.Higono,"Supercomputing of Circuit Simulation", Proc. Supercomputing'89, pp81-85 (1989)

[4] 笠原,成田,橋本,OSCARのアーキテクチャ,信学論D, Vol.J71-D No.8,pp.1440-1445,1988年8月

[5] H.Kasahara,H.Honda and S.Narita, "Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR", Proc. of IEEE Suprecomputing'90, pp.856-864,(1990)

[6] B.K.Brayton, F.G.Gustavson, G.D.Hachtel,"A New Efficient Algorithm for Solving Differential Algebraic Systems Using Implicit Backward Differential Formulas", Proc. IEEE,60,1, pp98-108,(1978)

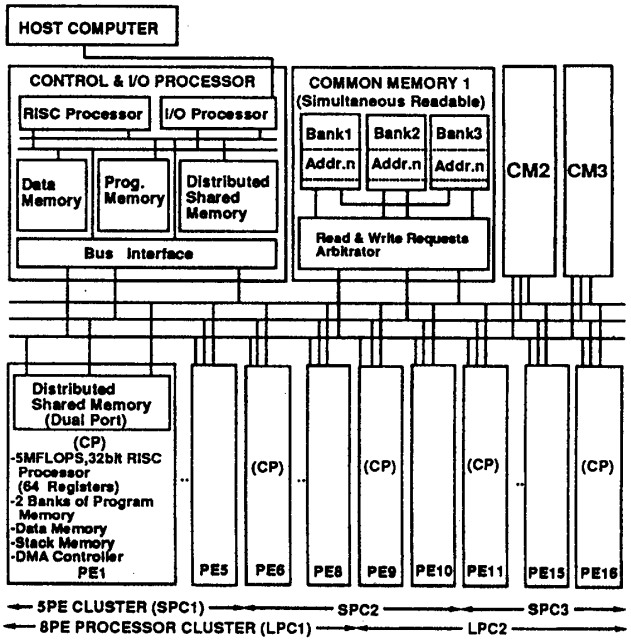

図1 OSCARのアーキテクチャ