5G－3

‘仮想クロック計算機’の提案

高畠 一哉　宮内 秀和　岡田 三郎
工業技術院 中国工業技術研究所

1．はじめに

数値計算用並列計算機ユーザーにとって並列処理用言語の習得が容易であること，プロセッサの台数やそれらの接続関係を意識せずプログラミングできることは魅力であろう．また作る側にとってハードウェア，言語，コンパイラ開発に要する労力が小さいことは大きなメリットである．ここでは既存のマイクロプロセッサによる比較的簡単なハードウェア構成と既存言語の小さな拡張により実現できる並列計算機‘仮想クロック計算機’を提案する．

2．概要

ジョブは1つのマスタプロセスと任意個のスレーブプロセスからなる．マスタプロセスはジョブの開始から終了まで動き続ける．clock文なる文がマスタプロセス中に置かれる．1つのclock文実行から次のclock文実行までの期間を仮想クロックサイクルという．スレーブプロセスはマスタプロセスか他のスレーブプロセスのfork文によって起動されその仮想クロックサイクル内に終わる．マスタプロセスのclock文は仮想クロックサイクル内の全スレーブプロセスが終るまで実行が待たされる．

変数にはプロセス内部変数とクロックド変数とアポストロフィ変数がある．プロセス内部変数は他のプロセスから参照代入不可である．クロックド変数とアポストロフィ変数はaとa'というように対で存在しマスタプロセス中でclock文が実行されたときa'の値がaに代入される．仮想クロックサイクル内でa'に代入がなかった場合aの値は次の仮想クロックサイクルに継承される．クロックド変数は全プロセスから参照可能で代入不可，アポストロフィ変数は全プロセスから代入可能で参照不可である．複数のプロセスが同一のアポストロフィ変数に代入を行う場合その順序は非決定的でよいものとする．

仮想クロックサイクル内においてクロックド変数は不変であるので各プロセスは互いに干渉しない．複数のプロセッサが任意の順序でスレーブプロセスを処理して構わない．

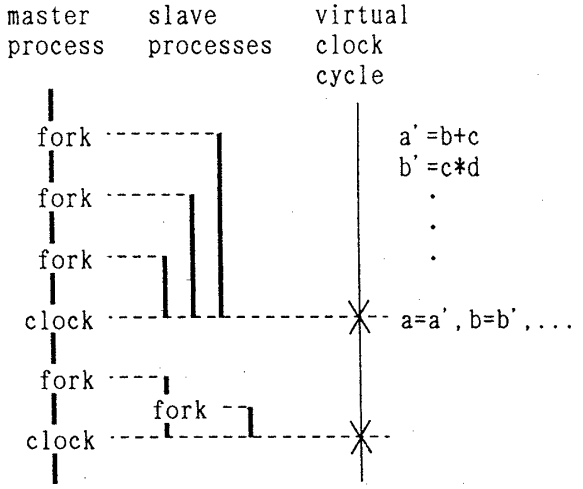


図1 プロセスの流れと仮想クロックサイクル

```
main(){
  clocked double A[N][N], B[N][N];
  .
  .
  for(i=0;i<N;i++)for(j=0;j<N;j++)B'[i][j]=A[i][j];
  clock;
  for(k=1;k<n;k++){
    for(i=0;i<N;i++)for(j=0;j<N;j++)fork mult(A,B,i,j);
    clock;
  }
  .
  .
}
process mult(A,B,i,j)
clocked double A[][N], B[][N];int i,j;{
  int k;double tmp;
  for(k=0,tmp=0.;k<N;k++)tmp+=A[i][k]*B[k][j];
  A'[i][j]=tmp;
}
```

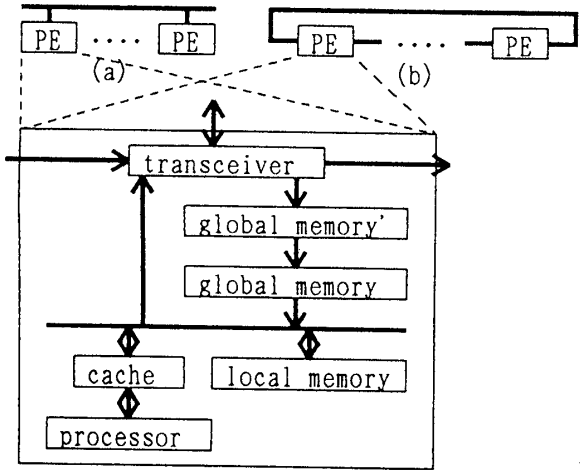
図2 プログラム例

Proposation of 'Virtual Clocked Machine'
Kazuya Takabatake(takahata@giric.go.jp),
Hidekazu Miyauchi, Saburo Okada
Government Industrial Research Institute, Chugoku
2-2-2 Hiro-Suehiro Kure Hiroshima 737-01

行列Aのn乗を計算するプログラムの例を図2に示す．この言語はc[1]を拡張したものである．main()がマスタプロセス，mult(A,B,i,j)は行列ABのi,j成分を計算するスレーブプロセスである．プロセス間通信はfork時の引数（multの場合A,B,i,j）とクロックド変数，アポストロフィ変数のみにより行われる．

3．ハードウェア

ハードウェアには図3(c)に示すプロセシングエレメントをバス(a)あるいはリング状接続(b)でつないだものを用いる．プロセシングエレメントの1つはマスタプロセス用に割り当て（マスタエレメント），残りはスレーブプロセスに割り当てる（スレーブエレメント）．



(c)

図3 ハードウェア構成

スレーブエレメントは全てのスレーブプロセスのコードを持っている．グローバルメモリにはクロックド変数，グローバルメモリ’にはアポストロフィ変数を置く．各エレメントでのクロックド変数の参照はそのエレメント内のグローバルメモリに対して行い，アポストロフィ変数への代入は全グローバルメモリ’へのブロードキャストにより行われる．グローバルメモリ’への書き込みに即時性は特に必要とされないのでプロセッサからグローバルメモリ’への経路中どこでキューイングしても良い．

スレーブエレメントへのスレーブプロセス割当の概念を図4に示す．fork文が実行されるとプロセストークン（スレーブプロセスの開始番地と引き数）がプロセスキューに投入される．空いているスレーブエレメントにプロセストークンが渡されスレーブプロセスが実行される．この割当処理は割り込みによりマスタエレメントが行えば良い．

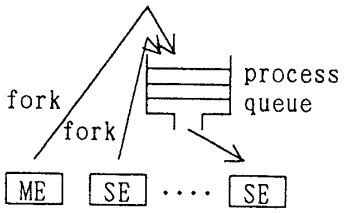


図4 スレーブプロセスの割当

clock文によりマスタエレメントは仮想クロックサイクル内でforkされた全スレーブプロセスの終了と全グローバルメモリ’への書き込み終了を待ってから全エレメントにグローバルメモリ’のグローバルメモリへのコピーとキャッシュのフラッシュを指示する．グローバルメモリ’からグローバルメモリへのコピーはシリアルポート付DRAMの1パッケージ内をグローバルメモリ部分とグローバルメモリ’部分に分けRAM-SAM間転送を用いれば高速に行える．これはパッケージ毎に並列に行われるため記憶容量を増やしてもかかる時間は変わらない．マルチキャッシュコヒーレンシが必要なのはクロックド変数だけであるがこれはclock文実行時にキャッシュをフラッシュするだけでよい．指示した内容が全エレメントで終わると次のクロックサイクルが始まる．

4．まとめ

1．で述べた長所を持つシステムを提案した．本システムではスレーブエレメント数に対しスレーブプロセスが十分多くある状況では均等な負荷分散と高い並列効率が期待できる．しかし不向きな処理も考えられる．今後行列計算などを中心にどのような処理であれば適しているか調べたい．

[1] B.W.Kernighan, D.M.Ritchie:“プログラミング言語C　第2版”共立出版