スレッド実行機構を導入したDatarolアーキテクチャについて

2L−8

川野哲生　星出高秀　日下部茂　谷口倫一郎　雨宮真人

九州大学大学院総合理工学研究科

1　はじめに

我々は従来のデータフローアーキテクチャの問題点を克服するものとしてDatarol アーキテクチャ[1][2][3] を提案してきた. Datarol マシンのプロセッサエレメント (PE) は循環パイプラインを基に設計したもので, プログラム中に内在する並列性を実行時に自然に抽出し循環パイプラインをアクティブなスレッドにより満たすことにより, 多重並行処理を効率的に行うことが出来る.

しかし, 並列性の低い処理ではPE 内のアクティブなスレッドの数が循環パイプラインの段数より少なくなり, パイプライン中に空きを生じ処理効率が低下する. また, 命令レベルでの同期処理および継続命令の指定による実行メカニズムはPE の高速化を考える上でボトルネックとなる可能性が高い.

今回 Datarol マシンにスレッドの排他的実行機構の導入を行うことにより, 逐次処理に対する効率改善, および同期処理の軽減について検討したので, その概要について報告する.

2　スレッド実行機構の導入

1 で述べた問題点を解決するために Datarol プロセッサにスレッドの排他的実行機構を導入する. スレッドとは, 外部とのインタラクションすなわちリモートメモリアクセスやリモート手続き呼出し等のレイテンシのある命令に対する結果待ちを行わず, 途中で中断することなく一括して実行出来るプログラムブロックである. スレッドは図1 に示すように入り口の同期処理部分, スレッドの本体部分及び継続スレッドの起動のための fork 部分とから成る.

図 2 にスレッド実行機構を導入したプロセッサエレメントの概略図を示す. 同図に示すように各プロセッサエレメントは FU(Function Unit), MU(Memory Unit), AC(Activation Controller), CU(Communication Unit), RQ(Ready Queue), MQ(Matching Queue) から成る. スレッドの実行は図 1 での同期処理部分と fork 部分を AC で, スレッド本体部分を FU で処理する. すなわち AC 内でスレッドの起動制御を行い, 実行可能となったスレッドを FU へディスパッチし, FU はこれを受けてスレッド本体部分をプログラムカウンタ (PC) を用いて排他的に一括して実行する.

FU での具体的な処理の流れは以下のとおりである.

1.RQ から実行可能なスレッドの情報 (インスタンス番号, スレッド先頭アドレス) を取り出す.

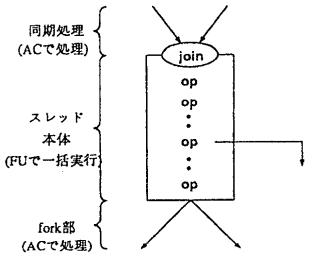

図 1: スレッドモデル

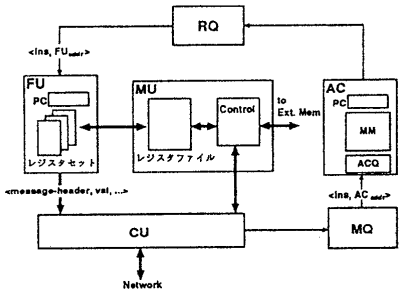

図 2: スレッド実行機構を導入したプロセッサエレメントの構成

2.FU 内のレジスタセットへ MU からレジスタファイルの内容を読み込む.

3.PC にスレッド開始アドレスをセットし, スレッドを一括実行.

4. レジスタ内容を MU へ書き出す.

FU 内に3 枚程度のレジスタセットを用意し, レジスタファイル内容の転送と FU 内でのスレッドの実行処理とをそれぞれ異なるレジスタセットを用い並行して行う. これによりスレッドの長さがある程度以上確保できた場合, レジスタ内容の転送に伴う FU でのコンテクストスイッチングのオーバーヘッドは隠蔽できる.

AC ではスレッド間の同期・起動制御を行なう. AC では MQ からスレッドの起動要求を取り出し, AC プログラムに従ってスレッド入り口での同期処理を行う. 同期に成

Thread execution mechanism on Datarol architecture

Tetsuo KAWANO, Takahide HOSHIDE, Shigeru KUSAKABE, Rin-ichiro TANIGUCHI and Makoto AMAMIYA

Graduate School of Engineering Sciences, Kyushu University

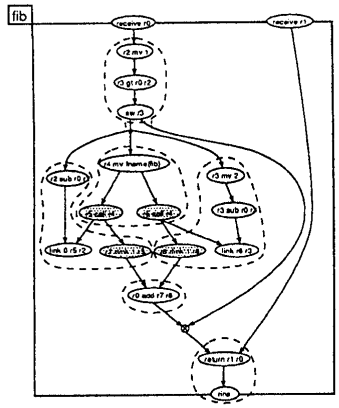

図 3: Fibonacci 数を求める Datarol グラフ

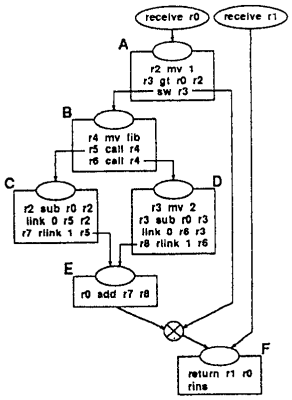

図 4: Fibonacci 数を求めるスレッドグラフ

功した場合 FU へスレッド本体部分の実行のためのディスパッチを行う．これに続いて継続スレッドの起動のための fork 命令部分を実行する．この fork 命令により発行されたスレッド起動要求は MQ とは別に AC 内に設けられた ACQ にストアされる．ACQ 内のスレッド起動要求を MQ に対して優先して処理することにより，同一インスタンス内の実行可能なスレッドを FU へ連続してディスパッチすることができ，FU でのコンテクストスイッチングの回数を減少させることができる．

このようなスレッドの排他的実行機構を導入することにより，従来問題となっていた逐次的な処理は 1 つのスレッドとして FU 内で一括実行され，アクティブなスレッドが少ない場合においても処理効率の低下を抑えられる．また，従来命令単位で行われていた同期処理や継続命令の指定は，スレッド単位で行われるようになり，従来 PE の高速化を考える上でボトルネックとなる可能性の高かった同期や継続処理の軽減が図れる．

3 プログラム例

図 3 に fibonacci 数を求める Datarol グラフを，図 4 にスレッド化したグラフを示す．ここで，図 3 中，破線で囲ったノードは図 4 中の 1 つのスレッドへと変換されていることを示す．

この例について，従来の Datarol マシンで実行した場合と，スレッド実行機構を導入した場合との比較を行う．ここでは，このプログラムが実行されている PE 内には他のインスタンスが無く，かつ call,rlink 命令によるレイテンシを無視して考える．従来の Datarol マシンでは図 3での最長パスより少なくとも 9 回循環パイプラインをトークンが回る必要があり，循環パイプラインの段数を 5 とすると，最短でも 45 clock 必要である．一方，スレッド実行機構を導入した場合最長パスは 5 スレッドであり，循環パイプライン 5 周分とスレッド内の命令実行分 (13 命令) を合わせても $5*5+(13-5)=33$ clock である．さらに，理想的な場合スレッド E, F は AC で連続してディスパッチされ，この場合は循環パイプライン 4 周となり $5*4+(13-4)=29$ clock であり，従来のものと比べかなり少ないクロック数で実行できる．また，同期や継続処理に関しても図 3 でのノード間のアークの数が 21 あるのに対し図 4 では 9 であり，join の数も従来のものが 4 あるのに対し，スレッドを導入した場合は 2 に減少し，これらの処理が軽減されている．

4 結論

循環パイプライン型の並列計算機において，プログラム中の並列度が，プロセッサエレメントの台数と循環パイプラインの段数の積より少ない場合，循環パイプライン中に空きができ，処理の効率が低下する．本稿では Datarol マシンにスレッドの排他的実行機構を導入することにより，並列度の低い逐次的な処理に対する効率の改善，及び PE の高速化を考える上でボトルネックとなる可能性の高い同期や継続処理の軽減を行った．

今後，多くのアプリケーションに対し，本方式の有効性についての評価を行うとともに，ハードウェアの詳細な設計を行う．

参考文献

[1] M.Amamiya and R.Taniguchi, “Datarol: A Massively Parallel Architecture for Functional Language”, Proc. SPDP, pp.726-735, (1990)

[2] 薗田，上田，谷口，雨宮, “Datarol プロセッサの最適化設計と負荷制御方式”，情報処理学会「並列処理シンポジウム JSPP ’90」，pp.121-128, (1990)

[3] 星出，薗田，谷口，雨宮, “並列計算機 Datarol マシンにおける資源管理と負荷制御方式”, 信学技法, CPSY91-7, pp.25-32, (1991)