ショートノート

スレッド内無従属性表現とそれに基づく細粒度コンピュータアーキテクチャの提案†

小西宏和†† 李鼎超††
有田隆也†† 曽和将容††

本論文では異なったスレッドの命令間にのみ本質的な従属性をもたせた実行モデルと，この実行モデルに基づいた計算機アーキテクチャを提案する．

1. はじめに

コンピュータの高性能化をめざして並列処理コンピュータの研究が積極的にされるようになっており，その典型的なコンピュータとしてマルチプロセッサシステムがある．マルチプロセッサ用プログラムは，複数本のスレッドと呼ばれる命令流からできており，マルチプロセッサを高性能化することにはこのスレッドを増し，並列度を増やすのが一般的である．しかしながらスレッドの増加は，当然のことながらコンピュータのハードウェア量を増加させる．

マルチプロセッサではスレッドの命令の1つのプロセッサで順番に実行される．このことは，スレッドがプロセッサ従属性，すなわち，命令がプロセッサを使う順番を表していることを意味している．一般にスレッドには，プロセッサ間通信を減らす目的で，プロセッサ従属性以外に，データ従属性などの従属性（混乱が生じない限り以後単に従属性という）が同時に組み込まれることが多かった[1]．しかし，最近のコンピュータのパイプライン化やスーパスカラコンピュータの並列化処理などのように，コンピュータ高性能化のためスレッド内の並列処理を行うと，この従属性がデータハザードなどを起こしスレッド内の並列処理を阻害する原因となるとともに，これを検出するための複雑なハードウェアが必要となっていた．比較的並列度の小さい細粒度並列コンピュータでは，プロセッサ間通信はマルチポートレジスタなどを用いて簡単に行えるので，プロセッサ間通信はあまり大きな問題とならず，それよりもスレッド内の従属性をなくし，スレッド内の並列度を増すとともに並列性抽出回路を簡単化することが，コンピュータ高性能化のために重要である．

本論文では，スレッド内にプロセッサ以外の従属性を含まず，従属性はスレッド間のみに存在することを前提とした実行モデルを提案する．そしてこのモデルのもとで，スレッドを極力増やさずにスレッド内の並列性を抽出する一コンピュータアーキテクチャを提案する[2]．

2. スレッド内無従属性表現

マルチプロセッサ用プログラムを**図1**に示す．この図で円は命令を，実線アーク（矢印）は命令間の従属性を，点線アーク（縦線）はプロセッサ従属性を表している．点線アークでつながれた命令列はスレッドであり，各スレッドはそれぞれ専用のプロセッサで実行される．実線アークを受け取っている命令は，プログラムカウンタがその命令の実行を指示したとき，そのアークにコントロールトークンと呼ばれる実行許可信号が到着していれば実行可能となる．実線アークを送出している命令は実行が終わったときトークンを送出する．このプログラムでは，スレッド内に命令AC間やEG間のような従属性が組み込まれているので，AとCやEとGを並列に実行することは不可能である．**図2**は本論文で提案するプログラムである．このプログラムには，スレッド内には従属性は含まれておらず，従属性はスレッド間だけにある．それゆえ，スレッド間の従属性さえ満足されれば，スレッド内の命令

† A Multi-threaded Execution Model without Intra-thread Dependence and a Fine-grain Parallel Computer Architecture Based on It by HIROKAZU KONISHI, DINGCHAO LI, TAKAYA ARITA and MASAHIRO SOWA (Department of Electrical Engineering and Computer Science, Faculty of Engineering, Nagoya Institute of Technology).

†† 名古屋工業大学工学部電気情報工学科

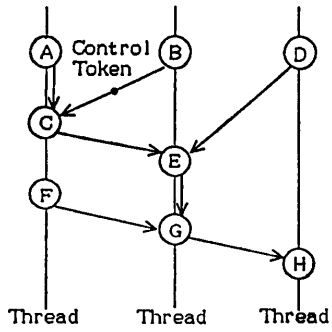


図 1 マルチプロセッサ用プログラム
Fig. 1 A program for a multi-processor.

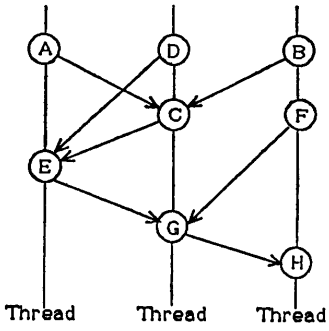


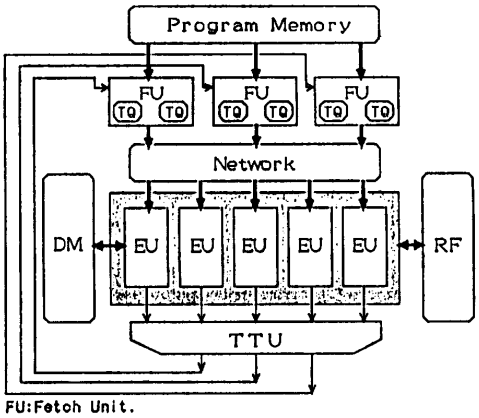
図 2 スレッド内無従属性のプログラム
Fig. 2 A program without dependence in the thread.

は並列に実行できる．スレッドは命令フェッチの順番のみを示すことになる．この方式では従来のパイプライン処理のハザード検出のように，従属性を常時監視する必要はない．また，スレッド間の従属性はコンパイル時に付加され，スレッド内には従属性がないので，スレッド内の並列性の抽出に，従来のスコアボード方式のような実行時に並列性を抽出する複雑なハードウェアを必要としない．それゆえこのプログラムを実行するコンピュータには並列性検出の時間が含まれず高速である．

命令間の従属性を表す基本ブロック単位のグラフ表現が与えられたとき，スレッド数＞入力アーク数の最大，であるならば，ある命令をスケジュールする際にその命令に向けてアークを出していないスレッドが少なくとも1つはあるので，そのスレッドにその命令をスケジュールすることによりスレッド内無従属性にすることが可能である．そうでない場合には新たな従属性を命令間に付加することで変換可能である．しかし，レジスタマシン型アーキテクチャでレジスタ数が十分大きいときやレジスタリネーミング機構を装備するときは，半フロー従属関係を考慮する必要がないので，入力アーク数の上限が定まり，スレッド数＞入力アーク数，となるように設計することが可能である．本論文は，スケジュールの質の検討を対象としていないが，スレッド数があまり十分でないときには，静的スケジュールの重要性が増すことが考えられる．

3. コンピュータアーキテクチャ

図 3 はスレッド内無従属性のプログラムを実行する計算機アーキテクチャである．図はフェッチユニット (FU) 3台，実行可能命令の実行ユニットに送るネットワーク，実行ユニット (EU) 5台，トークン送出ユニット (TTU)，データメモリ (DM)，レジスタファイル (RF) からなる場合を示している．レジスタファイルは，すべての実行ユニットに接続されたマルチポートレジスタである．TQ は，送られてきたトークンを格納するトークンキューで，1つのフェッチユニットには，それぞれ他のフェッチユニットに対応した専用のキューを置くので，この場合はフェッチユニットごとに2つずつ装備している．あるフェッチユニットから送られたトークンはそれ専用のキューに格納される．したがって，この場合は各トークンキューを使った実行制御方式では，1つのスレッドから他のスレッドに送出するトークンの追越しが許されていない．すなわち，あるスレッドから他のスレッドに送られるトークンのうち，先に送り出されるようにプログラムされたトークンを，その後から送り出されるようにプログラムされたトークンが追い越すことは許され



FU:Fetch Unit.
EU:Executive Unit.
TTU:Token Transmitting Unit.
TQ:Token Queue.
DM:Data Memory.
RF:Register File.

図 3 計算機アーキテクチャの例
Fig. 3 A parallel architecture based on this execution model.

ない．本方式では同一スレッド内の命令が並列に実行されるので，命令の実行開始の順序がプログラムどおりでも，命令の実行時間に差があるとトークンの追越しを引き起こしてしまう．

これを避けるために本トークンは色付けされており，また，フェッチユニットには，次に送るべきトークンのカラーを保持するカウンタと，カラー数マイナス1個のバッファが装備されている．そして，実行終了したトークン送出命令のカラーがフェッチユニットに戻ってきて，もしカラーカウンタと一致したならば，トークンは送出されカウンタはインクリメントされ，もし一致しないならば，一致するまでバッファで保持される．これによって，カラーの異なる命令の実行終了順序が変更しても問題が生じない．カラー数については，4章のシミュレーションでは1ビット2色として行っている．詳細な検討が必要であるが，2ビット4色程度で十分ではないかと思われる．

本コンピュータは次のように動作する．それぞれのフェッチユニットはプログラムカウンタの示す命令を，それぞれのプログラムメモリからフェッチし，トークンキューを参照して実行可能であるかどうかを判断する．実行可命令であればネットワークを通して実行ユニットに送る．実行不可命令であれば実行可になるまで待つ．実行ユニットは命令の実行を終わると，もしその命令がアークを送出しているならば，トークン送出ユニットを通してトークンをフェッチユニットに送る．命令のフェッチは，それぞれのスレッドのフェッチユニットに実行不可命令が存在しない限り，先行命令の実行終了を待たずになされ，それが実行可であれ，実行ユニットに空きがある限り実行ユニットに送られるので，1つのスレッドで命令が並列実行される．トークンキュー方式の実行制御機構（一般化静的順序制御機構[4]）は，トークンとしてパルスを送れば良く，またキューではなくカウンタでよいので簡単で高速である．

データフローマシンでは，トークンはデータ自体と制御の両方を一体化して表している．GAOのマシン[5]ではこれを分離し，制御のみをスケジュールユニットによりトークンとして流すことにより，効率的なパイプライン実行や従来どおりのデータメモリの実装を可能としている．本アーキテクチャでは一般化静的順序制御機構[4]を利用して制御トークンを送っており，制御のデータからの分離により効率的なパイプライン実行を狙っている点が GAO のマシンと共通する．ただし，本アーキテクチャでは，データフローマシンや GAO のマシンのようにプログラムの広範囲部分から並列性を抽出するのではなく，あくまでもプログラム中の細かい局所部分の並列性に限定し，より実用性を目指している点がアプローチとして異なる．

4. 簡単な実行例

図3の構成で，図4の虚数の積を求めるプログラムを実行したときのトレース結果を図5に示す．なお，図4では1命令でひとつしかトークンを消費できない

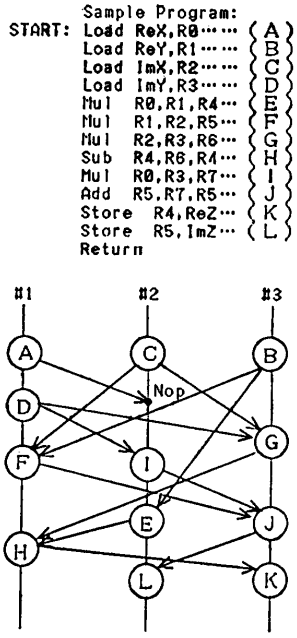


図4　サンプルプログラム
Fig. 4　A sample program.

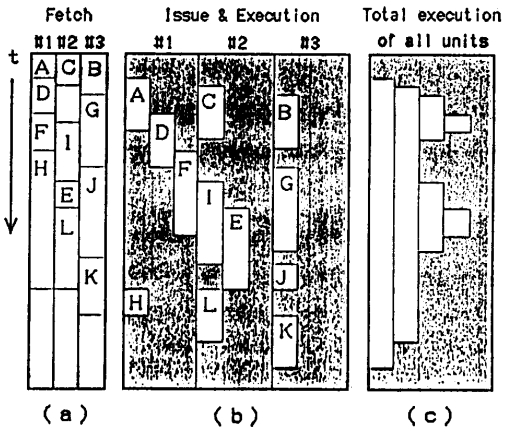


図5　サンプルプログラムのトレース結果
Fig. 5　A trace of the execution on the program of Fig. 4.

表 1 トレース結果のまとめ
Table 1 A summary of the measurements.

	命令数	実行時間		速度向上率	平均並列度	
		PN方式	本方式		PN方式	本方式
サンプル1	12	97	40	143%	1.3	2.6
サンプル2	20	144	72	100%	1.2	1.9
サンプル3	18	138	50	176%	1.1	2.3

型の一般化静的順序制御機構を想定しているので，NOP が加わっている．図5(a)は命令フェッチの様子を示し，(b)は命令の発行と実行の様子を示す．同図(c)は並列度をみるために(b)をまとめたものである．図5よりこのプログラムのスレッド数が3にもかかわらず，4の並列度が出ていることがわかる．**表1**にサンプルプログラムを実行した場合の速度向上率と平均並列度を示す．サンプル1は図4のプログラム，サンプル2は，メモリ内の整数と実数を整数実数変換を行いながら四則演算するプログラムで，比較的並列度の変化の激しいもの，サンプル3は，サンプル2と同様な処理を行うプログラムであるが，並列度のばらつきがすくないものである[2]．PNコンピュータ[3]に比べて速度向上率が100～176%となっており，並列度が1.1～1.3のものが本方式では1.9～2.6程度となっている．

5. おわりに

従属性をスレッド間のみに制限するという，スレッド内無従属性表現を提案し，一例として，それをトークンキューによる実行制御機構を用いて実行するアーキテクチャを提案した．そして，このアーキテクチャ上で簡単なプログラムを実行し有効性を確認した．

本論文は着想の提示に重点を置いてきたが，今後，静的スケジュール法を含んだプログラム構築アルゴリズムやカラー数の決定を含んだ詳細なコンピュータアーキテクチャについて検討する予定である．

参考文献

1) Shetler, J. and Butner, S. E.: Multiple Stream Execution on the DART Processor, *IEEE ICPP '90*, pp. I-92-96 (1990).
2) 小西，有田，曽和：本質的先行性明示化スレッド表現とそれに基づく細粒度並列実行方式，電子情報通信学会技報，CSPY 91-38 (1991).
3) 曽和：PN (Parallel Neumann) コンピュータの提案，情報処理学会コンピュータアーキテクチャシンポジウム論文集，pp. 109-114 (1988).
4) 高木，有田，曽和：問題が持つ先行関係のみを保証する高速な静的実行順序制御機構，情報処理学会論文誌，Vol. 32, No. 12, pp. 1583-1592 (1991).
5) Dennis, J. B. and Gao, G. R.: An Efficient Pipelined Dataflow Processor Architecture, *Proc. Supercomputing*, pp. 368-374 (1988).

(平成4年2月10日受付)
(平成4年10月8日採録)

小西 宏和
1968年生．1991年名古屋工業大学電気情報工学科卒業．同年同大学大学院工学研究科博士前期課程入学．電子情報通信学会会員．

李 鼎超（正会員）
1962年生．1983年北京航空航天大学情報工学科卒業．1991年福井大学大学院博士前期課程修了．現在，名古屋工業大学大学院博士後期課程在学中．並列処理システムに関する研究に従事．電子情報通信学会会員．

有田 隆也（正会員）
1960年生．1983年東京大学工学部計数工学科卒業．1988年同大学大学院工学系博士課程修了．工学博士．同年名古屋工業大学工学部電気情報工学科助手．並列計算機アーキテクチャ，プログラミング方法論に興味を持つ．IEEE，電子情報通信学会各会員．

曽和 将容（正会員）
1974年名古屋大学大学院博士課程（電気電子専攻）修了．同年群馬大学工学部情報工学科助手．1976年助教授．1987年名古屋工業大学教授．この間，並列処理，計算機アーキテクチャ，特にデータフロー計算機，コントロールフロー計算機など次世代コンピュータの研究に従事．工学博士．IEEE，ACM，電子情報通信学会各会員．