計算機アーキテクチャ　68－11
(1987・11・13)

ＦＰグラフリダクションマシンのシミュレーションによる性能評価

Performance Evaluation of the FP Graph Reduction Machine by Simulation

伊波　通晴　　高井　昌彰　　池部　優　　中村　維男　　重井　芳治
Michiharu IHA　Yoshiaki TAKAI　Masaru IKEBE　Tadao NAKAMURA　Yoshiharu SHIGEI

東北大学　工学部
Tohoku Univ.

あらまし－関数型言語FPを直接実行するマシンの１つとして、汎用パイプラインを簡約エンジンに用いたFPグラフリダクションマシンが提案されている。本システムの特徴は、線形再帰的なリスト処理関数をパイプライン上に動的に展開し、処理することにある。本論文では、本システムの心臓部である簡約エンジンを中心に、シミュレーションによる性能評価を行なう。簡約エンジンの評価量として、スループット並びに、各セグメントの稼働率を取り上げ、その結果に基づき、効率的な関数のマッピング方法および、簡約エンジンのマルチタスク処理の有効性について論ずる。

Abstract---This paper shows simulated performance of the FP graph reduction machine equipped by using a general-purpose pipeline as a graph-reducer. The key to executing functions in pipeline fashion is based on the scheme of unfolding and folding of linear recursions. According to the simulation results, we discuss efficient process allocations to the segments connected linearly, and effectiveness of multitasking scheme in a general-purpose pipeline.

１．まえがき

関数型言語は、いくつかの関数を関数形式によって結合し、新しい関数を再帰的に構成することでプログラムを記述する。関数の評価には本質的に副作用が存在しないことから、プログラムの検証性、変換性に優れ、さらに並列処理の可能性のため、関数型言語を指向した並列処理計算機の研究開発が現在盛んに行なわれている。

このような関数型言語を効率よく実行するマシンの１つとして、FPグラフリダクションマシンが先に提案されている[(1)]。本マシンの特徴は、簡約実行部である簡約エンジンに、汎用パイプラインを用いている点にある。処理は、関数型言語FP[(2)]のプログラム(関数とオブジェクト)のグラフ表現をリスト構造に変換し、リデックス(作用セル)を随時評価することによって遂行される。

本稿では、シミュレーションによる本マシン(特に簡約エンジン部)の性能評価について述べる。簡約エンジンのスループットおよびセグメントの稼働率から、シングルタスクとマルチタスクの両処理方式を比較検討する。また、この結果に基づき、効率の良いマッピング方法およびマルチタスク導入の有効性について検討する。

２．ＦＰグラフリダクションマシン

システムの構成を図１に示す。

構造体メモリ(SM)は、各セグメントからパケットで転送される、基本的なリスト処理要求を実行する機能メモリである。SMは、簡約エンジン(RE)の各セグメントからのアクセス競合を軽減するため、パイプラインのセグメント数と同数の多バンク構成としている。本システムにおいて、FPプログラムはオブジェクトセルと作用セルによって構成されるリスト構造に変換され、この構造体メモリに保持される。作用セルはリデックス(簡約可能な関数)を表わす。ここで、作用セルの簡約処理をタスク

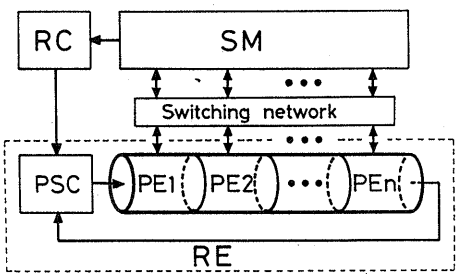


図1　FPグラフリダクションマシン

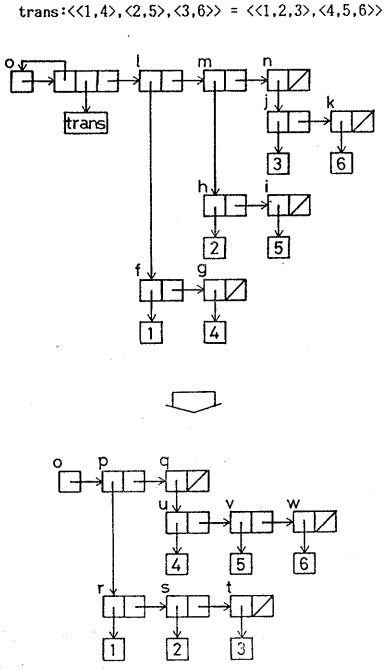
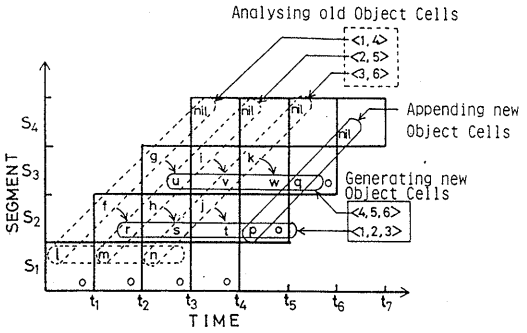


図2　transposeの簡約グラフ

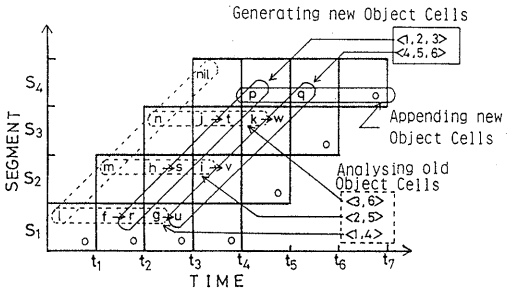
と定義する。

簡約コントローラは、複数の簡約スタックを用いて、構造体メモリ内部に生じた、作用セルを管理する。さらに、簡約可能となった作用セルのポインタを簡約エンジンに供給し、FPプログラムの簡約実行を統括的に制御する。

簡約エンジンは、パイプラインスケジューラ(PSC)と各々独立に動作する複数のプロセッシングエレメント(PE)を多段に縦続接続した非同期の汎用パイプラインから構成されている。簡約エンジンは、簡約コントローラ(RC)から作用セルのポインタを受取り、それによって指示



パイプライン時空間図へのマッピング
(行→時間軸方向、列→空間軸方向)



パイプライン時空間図へのマッピング
(行→空間軸方向、列→時間軸方向)

図3　transposeの時空間図

される関数と引数のリストに対し、各セグメントで保持している関数の書き替え規則に従って、簡約を実行する。パイプライン処理の詳細は文献(1)(3)(4)を参照。

パイプライン上でリスト処理を実現するため、1つのタスクはさらに複数のプロセスに分解され、パイプラインの各セグメント上に展開され、処理される[5][6]。各セグメントで実行される、これらのプロセスの集合は、パイプライン処理の時空間図によって表現できる。時空間図へのプロセスの配置を関数のマッピングという。1プロセスの処理内容は、セグメント内部のマイクロプログラムによって決定されているが、プロセスの繰返し数は、引数リストの大きさに依存する。即ち、一般にタスクは、時間－空間に2次元の広がりを持つ。しかし、簡約エンジンを構成するパイプラインの物理的なセグメント数の有限性から、タスクは空間方向に分割されなければならない。これをタスクの空間方向への分割と呼び、分割された単位をサブタスクと呼ぶ。文献(7)では、時間方向へのタスク分割も行なっているが、本稿ではシミュレーションの簡単化のため、物理的なパイプラインの長

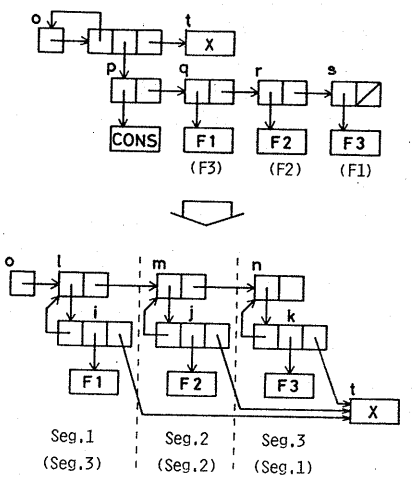


図4　CONSTRUCTIONの簡約グラフ

さによる空間方向の分割のみとする。

3．再帰関数のパイプライン処理

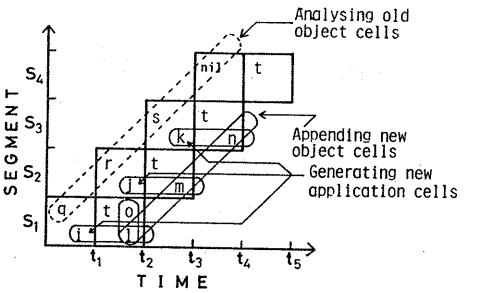
3.1 再帰関数のマッピング

パイプラインにおける単純な線形再帰関数の展開方向には、時間方向と空間(セグメント)方向の２つが可能である。時間方向に展開する場合、処理は通常の単一プロセッサにおける、繰返し化された再帰関数処理と同じように進行する。空間方向に展開する場合は、再帰呼出しが起動される時点で、引数は隣接するプロセッサに転送され実行される。どちらへの展開が効率的かは、関数の内部性質や作用させるデータ構造に依存するため、明確には決まらない[4]。しかし、パイプラインのセグメント数が十分あるとすれば、各セグメントを有効利用するため、空間方向に線形再帰関数を展開するほうが、一般には好ましいと言える。

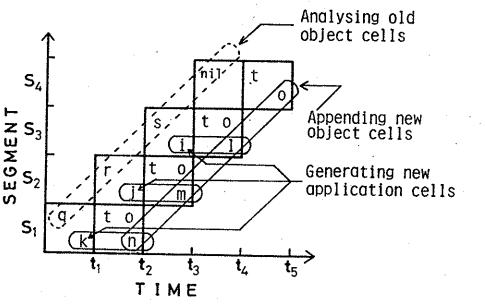
一方、線形再帰の２重構造をもつ再帰関数の場合、タスクは時間－空間の両方向に同時に展開される。原始関数"transpose"はその代表的な例である。図2は、transpose簡約化のグラフ、図3は、オブジェクトの行と列の展開方向を替えた、２種類の時空間図へのマッピングである。一般に、transposeが作用するオブジェクトの形は、動的に決定されるため、実行前に最適な展開方向(タスクのマッピング)を決定することは困難である。5.2節で、関数の展開方向がパイプライン処理に及ぼす効果について、シミュレーションの結果に基づき検討する。

3.2 マッピングの効率化

次に、FPプログラムをFFPに変換する際に、その引数



(a)pipeline consを使用するマッピング(CONS1)



(b)通常のconsを使用するマッピング(CONS2)

図５　CONSTRUCTIONの時空間図

の順序を変えることにより、マッピングが変更できることを関数CONSTRUCTION(以下CONS)を例に示す。CONSを用いたFP式 [F1, F2, F3] は、通常次のようにFFPに変換される。

[F1,F2,F3]→(CONS F1 F2 F3).

このとき、引数である関数列の順序を次のように逆にすることにより、パイプライン処理を効率良く実行することができる。

[F1,F2,F3]→(CONS F3 F2 F1).

図4がグラフの変換例(カッコの部分が改良部分)、図5がそのマッピング例である。パイプライン処理のため、図5(a)のマッピング方法では、"再帰→繰返し"の変換をPipeline cons[8]によって実現している。これに対し、図5(b)のマッピング方法では、通常のconsで処理が可能となり、各セグメントでの構造体メモリアクセスを、１回づつ減らすことが可能である。

4．簡約エンジンの制御

4.1 必要性

シングルタスクにおける以下の短所がシミュレーションの結果から明かになっている[9]。

(1)各セグメントから構造体メモリをアクセスする遅延時間が、処理時間に大きく影響する。

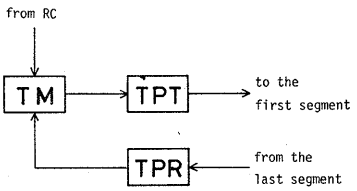


図6　パイプラインスケジューラ(PSC)の構成

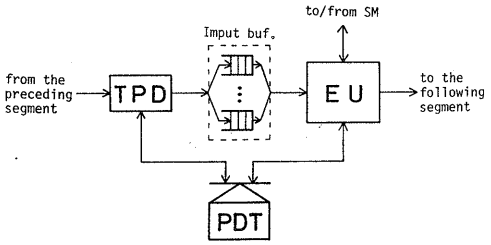


図7　各セグメントの構成

(2)タスクを構成する各プロセスの処理時間のばらつきにより、1タスクの実行中にアイドル状態になるセグメントが多数生じる。

従って、この欠点を克服するため、依存関係のない複数のタスクを同時にパイプラインに投入し、独立に実行可能なプロセスを並行に処理する、マルチタスクの手法を導入する。これにより、セグメントのアイドル時間を別のタスクの処理時間で隠してしまい、セグメントの稼働率とスループットを向上させることが可能である。

4.2　制御機構

パイプラインスケジューラ(PSC)は、簡約コントローラから送られてきた作用セル(タスク)をreadし、タスクパケットを組立て、パイプラインに転送する。マルチタスク実現のため、上記の機能に加え、タスク別にタスクidを付加する。また、パイプラインの最終セグメントから戻されるタスクパケットは、管理テーブルと情報の書き替えを行なった後、再びパイプラインの第1セグメントに投入される(図6参照)。

一方、マルチタスクを実現するための、各セグメントの構成を図7に示す。セグメントは、プロセスのデコード・実行および結果の転送を行なうEU(Execution Unit)に加え、タスク別の入力バッファ、それぞれの入力バッファに対しタスクidごとにタスクパケットを格納するTPD(Task Packet Distributer)、およびタスク切換えに必要なデータを登録しておくプロセス記述テーブル(PDT)から構成される。EUは実行可能なタスクを選択し、TPDとは独立に動作する。また、タスクの切換えは、メモリアクセスが生じた時点で発生する。マルチタスク制御の詳細については、文献(6)を参照。

5．シミュレーションによる性能評価

シミュレーションを行なう際、各部の処理時間として以下のものを考慮する。

◦パケット転送／授受時間：マシンの各ユニット間における、パケットの転送／授受にかかる時間。

◦構造体メモリにおける処理時間：構造体メモリ内でのパケットの分解・組立て時間を含む read/write 時間。

◦簡約コントローラ及びセグメント内デコード時間：パケットの内容から処理を決定する時間。

◦簡約コントローラおよびセグメント内管理テーブル書き替え時間：RC内部における、簡約スタックの動的チェイン並びにセグメント内における、inputバッファやタスク切換えのためのテーブルの参照および更新にかかる時間。

簡単化のため、本シミュレーションでは特に断らない限り、これらの時間をすべて1(ユニット時間)とし、これ以外の時間は無視する。また、多バンク構造体メモリに対するget-cellの戦略として、次のことを仮定する。すなわち、FPプログラムの簡約に伴い、新たにセルが必要になった時には、常にそのセグメントと同じ添字を持つメモリバンクにアクセスし、そこにセルを生成するものとする。

シミュレーションの際、パイプラインと比較されるユニプロセッサは、FPグラフリダクションマシンと同様の簡約コントローラを持ち、簡約エンジンに通常の計算機と同じように逐次的に処理を進める、1台のプロセッサから構成されているものとする。

シミュレーション1

"transpose"が作用する行列の展開方向として、行を時間方向へ、列を空間方向へ展開するマッピング法をtrans1とし、行を空間方向、列を時間方向に展開するマッピング法をtrans2とし、両者のスループット(1/プログラムの処理時間)を求める。シミュレーションに用いたFPプログラムは、

p1：$trans_1 \circ trans_2 \circ \cdots \circ trans_{14}$：8×8 matrix

p2：$[trans_1, trans_2, \cdots, trans_{14}]$：8×8 matrix

である。p1は並列処理可能なタスクが存在しないプログラム、p2は14個のタスクが並列処理可能なプログラムである。図8はtrans1のtrans2に対するスループット比を示す。セグメント数およびシングルタスク、マルチタスクにかかわらず、trans1の処理能力が5~10%上回っている。どちらのマッピングも全体としては同数のプロセスを必要とするが、trans2の場合、時間方向のプロセスが

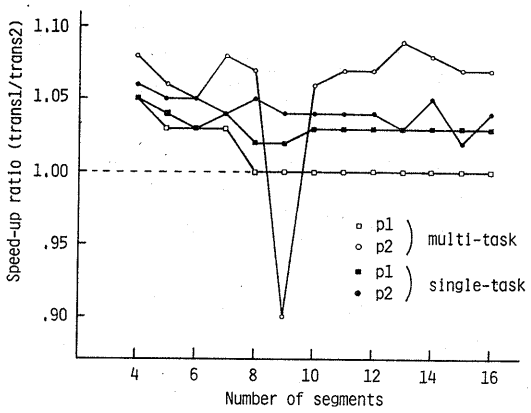


図8　セグメント数 対 処理速度向上率

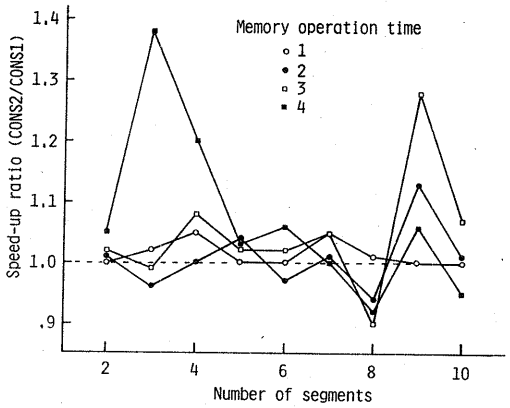


図9　セグメント数 対 処理速度向上率

1つ多い。このことは、trans2の各セグメントにおけるタスクの滞在時間がtrans1よりも長いことを意味し、そのためtrans1に比べ、スループットが低く抑えられている。パイプラインの性質上、各セグメントを有効に利用するため、タスクの滞在時間を短かくし、次段のセグメントに、より多く展開するようなマッピングが好ましい。

シミュレーション2

3.で述べたCONS1とCONS2を比較する。先に述べたようにマッピング2(CONS2)は、メモリアクセス回数が少なく効率が良いと考えられるが、ここでシミュレーションにより検証する。プログラムは次のものを使用する。

$$[\,[f_1,f_2,\cdots,f_8]_1,\ [f_1,f_2,\cdots,f_8]_2,\cdots \cdots\ [f_1,f_2,\cdots,f_8]_{14}]$$

ただし、"CONSTRUCTION"の純粋な処理時間比較のため、$f_i(i=1,2,\cdots,8)$の処理は行わないものとする。

構造体メモリ内の処理時間をパラメータとし、セグメント数の変化に対するCONS2を使用した場合のスループットと CONS1を使用した場合のそれとを比較した結果を図9に示す。ほとんどの場合、CONS2を用いたときのスループットが高く、マッピング改良の効果が現れている。ただし、物理的パイプラインの長さ8の点では、CONS2の使用が効率的とは言えない結果となっている。これは、オブジェクトセル生成の仮定により、アクセス競合の度合が変化することによるものである。

シミュレーション3

次の4種類のFPプログラムをグラフリダクションマシンに投入し、シングルタスクとマルチタスク処理を行なった場合の処理時間を、ユニプロセッサで処理した場合の処理時間と比較する。

p1：$[f_{1,1},f_{1,2},\cdots,f_{1,14}]\circ\cdots\circ[f_{8,1},f_{8,2},\cdots,f_{8,14}]$：vector length 15
p2：$[f_{1,1},f_{1,2},\cdots,f_{1,8}]\circ\cdots\circ[f_{14,1},f_{14,2},\cdots,f_{14,8}]$：vector length 15
p3：$[f_{1,1}\circ f_{1,2}\circ\cdots\circ f_{1,5},\ \cdots\ ,f_{5,1}\circ f_{5,2}\circ\cdots\circ f_{5,5}]$：14×14matrix
p4：$[f_{1,1}\circ f_{1,2}\circ\cdots\circ f_{1,14},\ \cdots\ ,f_{5,1}\circ f_{5,2}\circ\cdots\circ f_{5,14}]$：5×5 matrix

ここで、p1、p2のfはリスト長を求める関数"lengthで"、p3、p4のfは転置を行なう関数"transpose"である。また、p1、p2は並列処理可能なタスクの割合が小さなプログラム、p3、p4は並列処理可能なタスクの割合が大きなプログラムである。これらのプログラムのタスク依存関係をグラフ表現すると図10のようになる。

図11は、セグメント数を変化させ、ユニプロセッサに対するパイプラインのスループットの比を測定したシミュレーション結果である。どの結果も物理的なパイプの長さが4以上でユニプロセッサの処理能力を上回っている。p1、p2は並列性に乏しいため、セグメント数8の条件において、スループットはマルチタスクで25~45%、シングルタスクで55~75%、向上している。それに対し、p4ではセグメント数8で約3倍の向上となっている。これは、プログラムにおけるタスクの並列度に大きく影響を受けている。

p1、p2において、マルチタスクは、シングルタスクに比較してスループットが低い結果となっている。これは構造体メモリにおける処理時間を1に設定したため、並列性の乏しいp1、p2では、マルチタスクにおけるタスクの切換のオーバーヘッドが、処理速度に大きく影響したためと考えられる。図12に示すように構造体メモリ内の

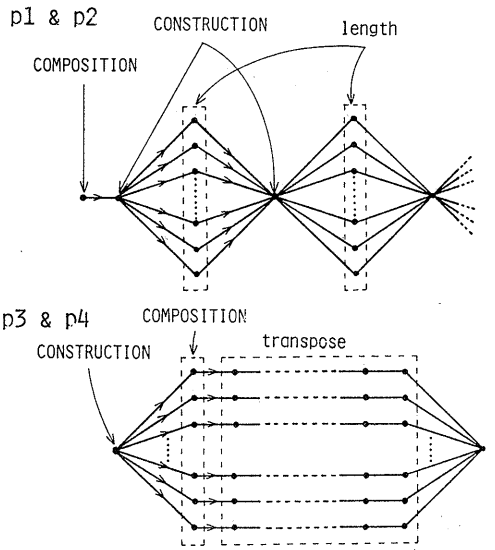
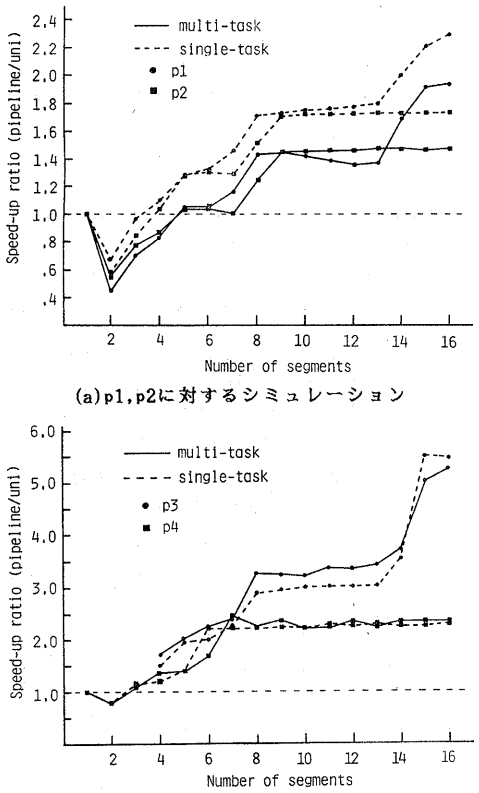


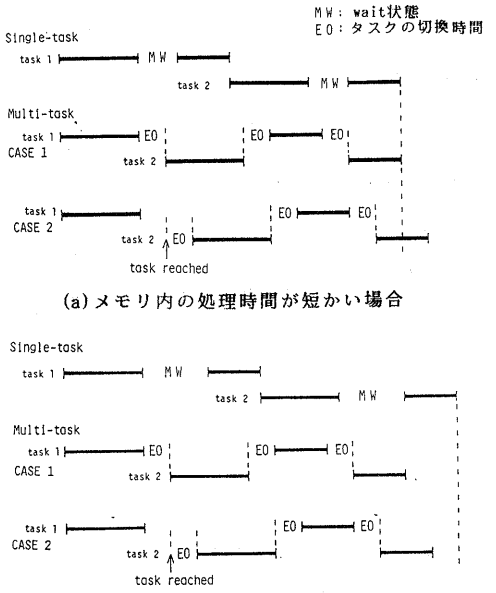
図10　タスクのグラフ表現



(a)p1,p2に対するシミュレーション

(b)p3,p4に対するシミュレーション

図11　セグメント数 対 処理速度向上率



(a)メモリ内の処理時間が短かい場合

(b)メモリ内の処理時間が長い場合

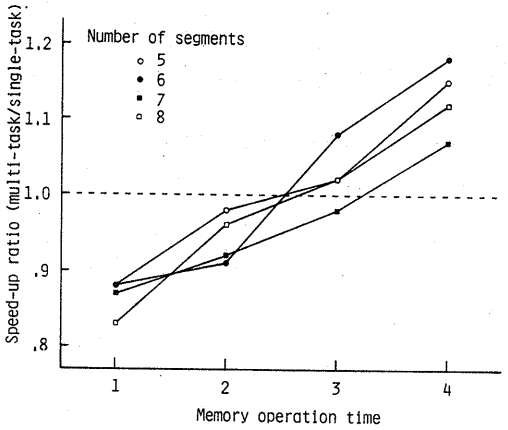
図12　プロセスのタイムチャート

処理時間が短かいと、(a)のようにタスク切換えのオーバーヘッドが大きく、メモリアクセスの少ないプログラムでは、マルチタスクの効果が得られない。さらに、case2のようにメモリアクセスが発生した時点で他のタスクがセグメント内に無く、wait状態の途中でタスクが到来すると、タスク切換えのオーバーヘッドによって、シングルタスクより処理速度が悪くなる。(b)のように、メモリ内での処理時間が長い場合、マルチタスクの効果がはっきりと現れる。この点に関しては、さらにシミュレーション４で確かめる。

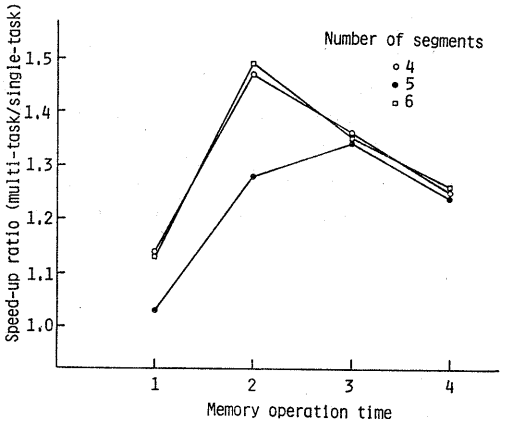
シミュレーション４

シミュレーション３において、メモリ内の処理時間を１(ユニット時間)とした場合、p1とp2において、マルチタスクは、シングルタスクよりスループットが劣っていた。そこで、メモリ内の処理時間を変化させ、スループットを比較する。処理するFPプログラムとしては、シミュレーション３で用いたp1及びp3を使用する。

図13がその結果である。グラフのパラメータはパイプラインの物理的な長さである。p3の場合、メモリ内の処理時間が２になった時点で、シングルタスクに対するマルチタスクのスループットの比は、ピークをむかえている。また、p1の場合、メモリ内の処理時間が３以上でなければ、マルチタスクの効果は得られない。これは、TPDの処理に3ユニット時間かかるとしたため、メモリアクセスの少ないlengthを有するp1では、EUにアイドル状態

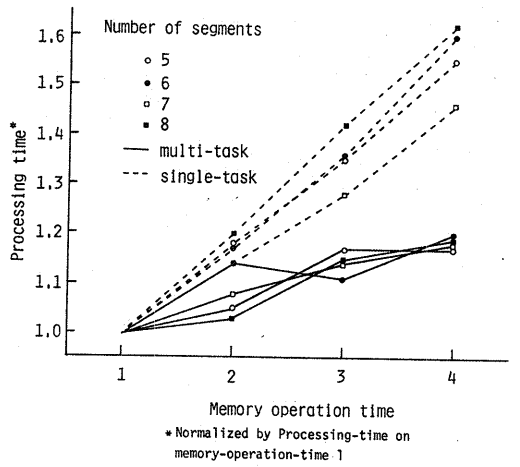


(a)セグメント数5でp1を使用したシミュレーション

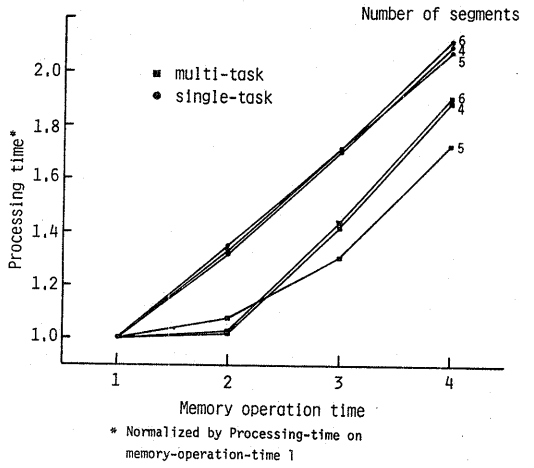


(b)セグメント数4でp3を使用したシミュレーション

図13　メモリ内処理時間 対 スループット比



(a)セグメント数5でp1を使用したシミュレーション



(b)セグメント数4でp3を使用したシミュレーション

図14　メモリ内処理時間 対 処理時間増加率

が生じるものと思われる。

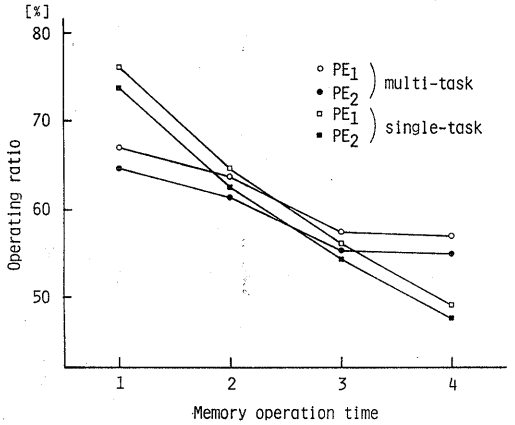
図14は、シングルタスクおよびマルチタスクにおけるメモリ内の処理時間に対する、プログラムの処理時間の関係を示している。ただし各ジョブの処理時間を、メモリ内の処理時間が１の場合のプログラム処理時間で正規化している。両者を比較した場合、明かにマルチタスクを行なったものの処理時間の増加が少ない。

次にセグメント内部のEUの稼働率を測定する。ここで稼働している状態とは、タスクがEU内に存在し、かつこのタスクが構造体メモリからの結果パケットを待っていない状態と定義する。図15は物理パイプの長さをp1に対し５、p3に対し４としたときの各セグメントの稼働率を表わしている。p1では、メモリ内の処理時間が３(ユニット時間)で、シングルタスクとマルチタスクの稼働率が逆転し、マルチタスクにおける各セグメントの稼働率が良くなっている。また、p3では２を境に、シングルタスク、マルチタスクの差が小さくなっている。これは、図13の結果と一致する。

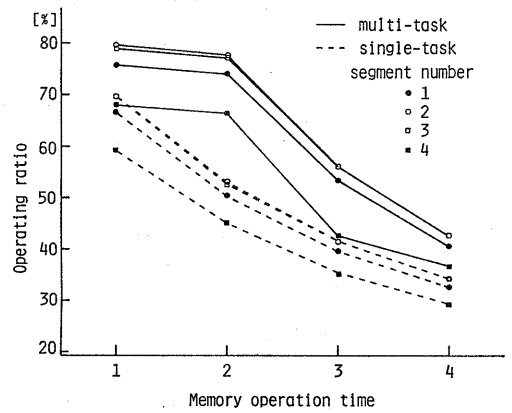
６．むすび

本稿は、簡約エンジン部の性能評価を中心に幾つかのシミュレーションを行なった。さらに、この結果に基づき、FPグラフリダクションマシンにおける効率的なマッピング方法、マルチタスクの有効性について検討した。

シミュレーション1により、関数のマッピングは、可能な限り滞在時間を小さくするように行なわなければならないことが明かになった。また、マルチタスク導入により、プログラムによっては、約40%の処理能力の向上が得られた。



(a)セグメント数5でp1を使使用したシミュレーション



(b)セグメント数4でp3を使用したシミュレーション

図15　メモリ内処理時間 対 セグメントの稼働率

現在、ガーベジコレクションの機能を保持していないため、大規模な実際的FPプログラムのシミュレーションは不可能である。従って、今後、ガーベジコレクションを含め、さらにシステム各部の詳しいシミュレーションを行ないシステム全体の性能評価を行なっていく予定である。

参考文献

(1)高井、池部、伊波、中村、重井："汎用パイプラインを簡約エンジンに用いたFPグラフリダクションマシン"，信学技報,CAS86-130(1986).

(2)J.Backus："Can Programing Be Liberated from the Neumann Style? A Functional Style and Its Algebra of Programs," CACM,vol.21,No.8,pp613-614(1978).

(3)Y.Takai,T.Nakamura,and Y.Shigei, "Control Scheme of Function-Level Computing on the Brain Structured Computer," Proc. IEEE 11th Int'l Comput. Software & Applications Conf.(COMPSAC87), pp.493-500, Oct.1987.

(4)池部、高井、伊波、中村、重井："FPグラフリダクションマシンのハードウェアシミュレータ",信学技報,CPSY86-68(1987).

(5)三科、坂井、中村、重井："パイプライン処理システムにおける関数型言語の実行方式",信学技報,EC85-42(1985).

(6)坂井、三科、中村、重井："リスト処理指向パイプライン処理システムに関する一検討",信学技報,EC85-43(1985).

(7)高井、伊波、池部、中村、重井："FPグラフリダクションマシンの階層的制御機構",第2回「データフローワークショップ」論文集.

(8)T.Nakamura, Y.Takai, and T.L.Kunii, "Pipelined supercomputing with list-structured data," Proc.2nd Int'l Conf. on supercomputing(ISC87), Vol.1, pp. 410-415, May 1987.

(9)伊波、高井、池部、中村、重井："FPグラフリダクションマシンの性能評価",信学技報,CPSY86-69(1987).