計算機アーキテクチャ　68－10
(1987・11・13)

ＦＰグラフリダクションマシンにおける グラフ構造の分散配置方法

A Memory Allocation Scheme for the FP Graph Reduction Machine

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

東北大学 工学部
Tohoku Univ.

あらまし　関数型言語FPを使用言語とし、汎用パイプラインを簡約エンジンとするFPグラフリダクションマシンが提案されている。本論文では、本システムにおけるメモリアクセス競合の緩和を目的とした、グラフ構造のメモリバンクへの分散配置方法を考察する。ハードウェアによる評価の結果、この方法による処理時間の短縮は、パイプラインへのタスクの展開方法、およびパイプラインへのタスクの投入形態により影響をうけることがわかった。

Abstract---In a multiprocessor system with a shared memory, memory access conflicts bring a heavy degradation of system performance. In this paper, we discuss a memory allocation scheme best suited for the FP graph reduction machine featured by a general-purpose pipeline as a reduction engine. The graph structures are assigned to the multiple memory banks shared by the segments of the pipeline. Our strategy for conflict-free memory access is that any linear list should traverse the array of memory banks. The simulation results show the effectiveness of our approach.

1. はじめに

関数型言語は、アルゴリズムの記述性と、並列処理の可能性の観点から特徴付けられる。しかし、従来のノイマン型計算機システムでは、関数型言語との間に大きなセマンティックギャップが存在するため、効率的な処理は困難である。そのため、関数型言語を指向した様々な並列計算機が研究されている。

著者らによって提案されているFPグラフリダクションマシン[(1)]は、これらの並列計算機の１つとして位置付けられる。本マシンは、関数型言語FPを使用言語としており、汎用パイプラインを用いることにより、再帰的リスト操作に内在するImplicitな並列性を引き出すことをねらっている。汎用パイプラインとは、パイプラインの各セグメントの機能を、データと共に供給することによって可変なものとし、従来のパイプラインに汎用性を持たせたものである[(2)]。本グラフリダクションマシンはデータ(オブジェクト)およびプログラム(関数)をすべてリスト構造で保持し、FPの評価規則に従いグラフリダクションを行なう。本マシンの基本的構成は、多バンクの構造体メモリがパイプラインの各セグメントによって共有される形式であるため、リスト構造の構造体メモリへの配置方法、およびリデックスの簡約処理(タスク)のパイプライン各セグメントへのマッピング方法が、本マシンにおけるリスト操作の効率に大きく影響する。

そこで本稿では、ハードウェアシミュレータによるシミュレーションにより構造体メモリのアクセス競合がタスクの処理時間に与える影響について議論する。はじめに関数型言語FPの特徴、および本FPグラフリダクションマシンの構成について簡単に述べる。次に、本システムの評価を目的としたハードウェアシミュレータについて述べる。次に、アクセス競合を緩和することを目的とした構造体メモリへのリスト構造の分散配置方法について述べる。最後に、ハードウェアシミュレータを用いて、構造体メモリへのアクセス競合によるタスクの処理時間に対する影響について検討する。

2. 関数型言語ＦＰ

FPは、J.Backusによって提案された関数型言語である(3)。FPシステムは、オブジェクトの集合、関数の集合、作用(application)、関数形式(functional form)の集合、および関数定義の集合から構成される。

オブジェクトは、与えられたアトムの集合から再帰的に構成される。関数は、これらのオブジェクトをオブジェクトに写像する。関数は、原始関数(primitive)であるか、関数形式であるか、あるいは定義されたものである。FPの関数はすべて1引数である。

関数形式は、関数を表現するまとまった１つの式であり、いくつかの関数をプログラム構成子(PFO,Program Forming Operator)を用いて結合することで表わされる。

FPシステムの唯一の演算は作用である。fを関数、xをオブジェクトとするとき、f:xは作用であり、その評価結果は新しいオブジェクトとなる。

3. ＦＰグラフリダクションマシン

3.1 システム構成

FPグラフリダクションマシンのシステム構成を図１に示す。本システムは構造体メモリ(SM)、簡約エンジン(RE)、SMとREを結合するネットワーク、そしてリダクションコントローラ(RC)から構成される。

FPプログラムは、FPの"作用"を繰返し実行することによって処理される。本マシンでは、FPプログラム中に存在する"作用"部分を表す作用セルへのポインタをREへ投入することで、FPプログラムを実行する。REは汎用パイプラインとスケジューラからなる。REに投入された作用の簡約処理(タスク)は、より細かいプロセスに分解される。このプロセスは、複数のセグメント(空間軸方向)と同一セグメント内(時間軸方向)に展開される。REの各セグメントは前段のセグメントから転送される情報と、同じセグメント内ですでに実行されたプロセスの履歴情報によりプロセスを決定し、これを実行する。SMは、プロセスの実行中にREの各セグメントから転送される命令に従い基本的なリスト操作を実行する。

3.2 構造体メモリ

構造体メモリ(SM)には、FPプログラムがリスト構造の形で保持される。SMは機能メモリであり、REから送られるリスト操作命令に従い、FPプログラムを表わすリスト構造の操作を行なう。またSMは、REからのアクセス競合を軽減するために多バンク構成をとっている。

本システムにおいては、FPプログラムのグラフは作用セルとオブジェクトセルから構成される。オブジェクトセルは大きく分けてCARフィールド、CDRフィールドから構成される。作用セルは、関数とオブジェクトの作用部分を表わし、大別してRET(return)フィールド、FUN(function)フィールド、OBJ(object)フィールドから成る。作用セルのFUNフィールドには関数へのポインタが、OBJフィールドにはオブジェクトへのポインタが格納される。RETフィールドには、作用の結果を返すセルへのポインタが格納される。

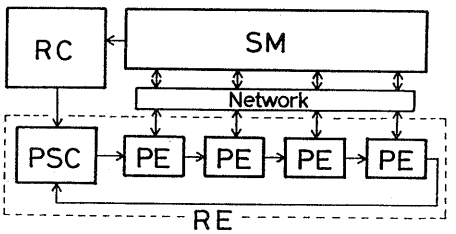

図1　FPグラフリダクションマシンの構成

3.3 簡約エンジン

簡約エンジン(RE)は、パイプラインスケジューラ(PSC)と、複数のセグメント(PE)を縦続接続した汎用パイプライン処理部から構成される。PSCは、RCから投入されるパケットと最終段のセグメントから再投入されるパケットをスケジューリングして、汎用パイプライン処理部にパケットを供給する。汎用パイプライン処理部の各セグメントは前段から送られてくるパケットの内容と、セグメント内に残っている前のプロセスの情報から次に実行すべきプロセスを決定する。プロセスの実行中に、各セグメントがSMに対して機械語レベルのリスト操作命令(read,pipelin_consなど)(4)をネットワークを介して転送する。

3.4 簡約コントローラ

簡約コントローラ(RC)は、SM内部の作用セルを管理し、簡約可能となった作用セルへのポインタをパケットとしてREに供給することで、グラフリダクションの実行制御を行なう。作用セルは、その依存関係に基づき、複数のスタック(簡約スタック)を用いて管理されている。依存関係があり、実行順序の制約される作用セルは、同一スタックに格納される。一方、依存関係がなく、並列に実行可能な作用セルは、別々のスタックに格納される。制御の詳細は、文献(5)で述べられている。

4. ハードウェアシミュレータ

ハードウェアシミュレータは、本システムの各処理要素の動作の非同期性を自然な形で再現でき、実際の

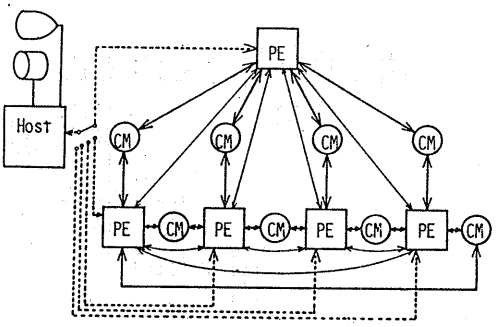

図2 ハードウェアシミュレータの構成

システムに近い状態でシミュレーションを行なうことができるという利点がある。本ハードウェアシミュレータの構成を図2に示す。シミュレータは、同型のプロセッシングエレメント(PE)が5台、隣接PE間での共有メモリ(CM)が8台、ホストコンピュータ (VAX11/730)から構成される共有メモリ型マルチマイクロプロセッサシステムである。

PEは、CPU (MC68000)、ローカルメモリ(128KByte)、ホストコンピュータとの通信を行なうシリアルI/O、そして計測用のタイマから構成される。各PEは最大4個のCMと接続可能である。また、近傍の4個のPEからの割り込みを調停するアービタを有する。各PEには、メモリ管理、ホストコンピュータとの間のプログラム転送等をサポートするモニタが載っている。また、CMは最大2個のPEからアクセスされるため、その競合を避けるためのアービタが備えられている。CMのメモリ空間は32KByteである。

5.リスト構造の構造体メモリへの分散配置方法

5.1 参照リストの分散配置方法

構造体メモリは多バンク構成であり、バンク毎に並列にリスト操作を実行できる。しかし、リスト構造の配置に偏りがあると、一部のバンクにアクセスおよび処理が集中し、システム全体のスループットが低下する。構造体メモリのバンク競合は、パイプラインの各セグメントで並列に実行されるプロセスにおいて参照が必要とされるセルの、構造体メモリバンクへの配置に依存する。そこで、パイプラインの時空間図上に展開された各プロセスがどのセルをアクセスするかを考慮し、アクセス競合が生じないようにリスト構造を構造体メモリの各バンクに分散配置する必要がある。そこで、十分な数の構造体メモリのバンクが存在し、かつ構造体メモリとRE間が完全結合であると仮定し、アクセス競合の緩和を目的としたリスト構造の構造体メモリバンクへの分散配置方法について検討する。

一般に、リスト処理においては、リストの要素を参照するため"リストをたどる"という逐次的な操作が必要となる。本マシンは、この逐次的な操作をパイプラインの空間（セグメント）方向に展開し、各要素に対する処理を時間的に重ね合わせることにより、スループットの向上を図っている。したがって、リストの要素を参照する際に各セグメントからのアクセス競合が生じないようにするための基本的戦略は、線形リストを構成するセルをそれぞれ異なるメモリバンクに分散配置することである。図3にconstructionの簡約前と簡約後のリスト構造を、図4にその時空間図を示す。図5にFPのプログラム構成子constructionの、アクセス競合の生じない分散配置の例を示す。この例では、cdr方向につながるセルを別のバンクに配置している。これは、constructionの簡約処理(タスク)が時空間図において、cdr方向の要素の処理が空間方向に展開されているためである。

5.2 実行時に生成されるリストの分散配置方法

次に、FPプログラム簡約途中に生成されるセルの構造体メモリのバンクへの分散配置方法について述べる。FPグラフリダクションでは、REに実行可能な簡約処理(タスク)を順次投入することで、リスト構造の簡約が進む。したがって、アクセス競合が生じないように分散配置したリスト構造を簡約処理した結果できるリスト構造も、次のタスクを実行する際にアクセス競合が起きないように分散配置されていなければ意味がない。すなわち、セルの生成時に他のセグメントとアクセス競合を起こさず、かつ新しく構成されるリスト構造がその参照時にアクセス競合を起こさないように分散配置されるという条件を満足するバンクを選んで新しいセルを生成するget_cellの戦略が必要である。この条件を満足する構造体メモリのバンクは、原始関数またはプログラム構成子の種類、第1セグメントが参照したバンク、およびセルの生成を行なうプロセスを実行するセグメントにより決定される。したがって、プロセスを時空間図上へマッピングする時点で、セルを生成する適切なバンクの位置を、プロセスを実行するセグメントとの相対的な位置で指定すればよい。この方法でセルを生成することで、元のリスト構造の配置がたとえアクセス競合を起こすものだったとしても、アクセス競合の生じない配置のリスト構造を生成できる。

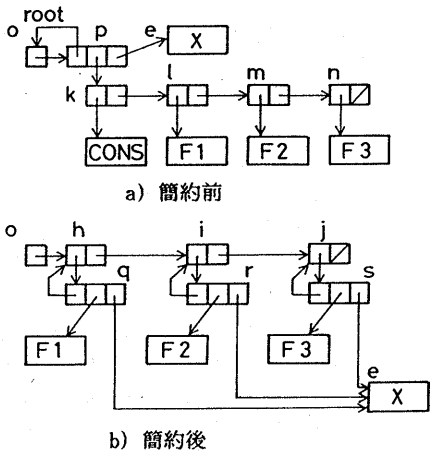

図3　constructionの簡約前後のリスト構造

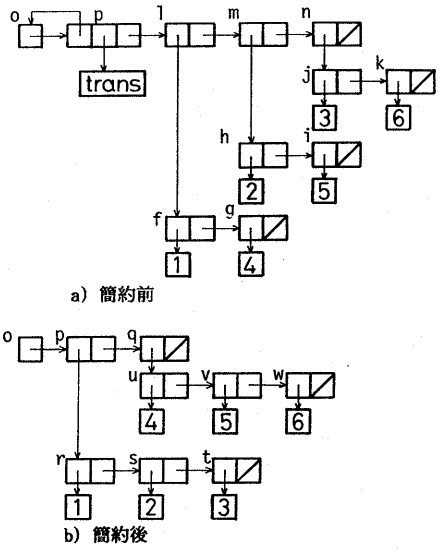

図6　transposeの簡約前後のリスト構造

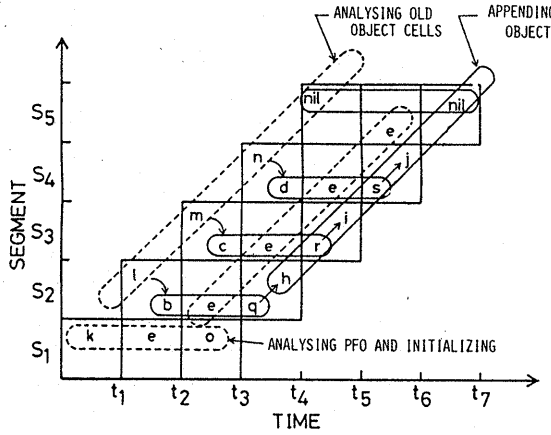

図4　constructionの時空間図

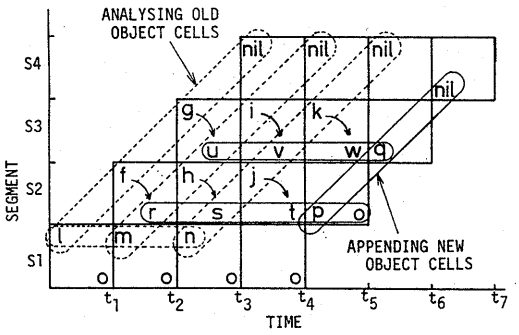

図7　transposeの時空間図

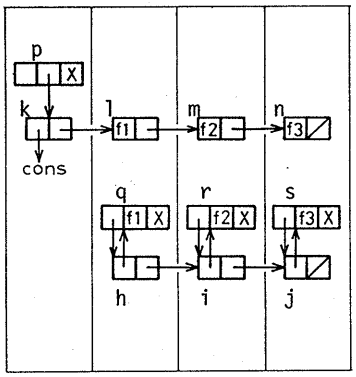

図5　constructionのリスト構造の分散配置例

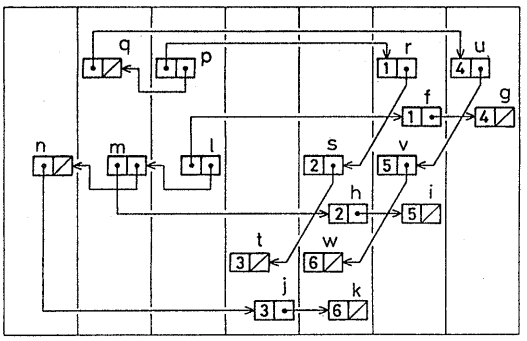

図8　transposeのリスト構造の分散配置例

新しいセルの生成例としてconstructionの場合を図５に示す。一方、原始関数の例としてtransposeをあげる。図６にtransposeの簡約前と簡約後のリスト構造を、図７にtransposeの時空間図を示す。transposeにおける代表的な1つのプロセスの処理内容は、１回のセル参照と１回のpipeline_consの対から成る。したがって、transposeの場合は、図８のようにリスト構造を分散配置することで、構造体メモリのバンクへのアクセス競合が避けられる。この際、get_cell戦略は1プロセス内で参照したセルが存在していたバンクに新しいセルを生成するということである。これにより、この分散配置に再現性を持たせることが可能となる。

5.3 評価方法

リスト構造の分散配置方法の有効性を評価するために、次の２つの指標を定義した。まず、あるタスクの時空間図上への展開方法が有する、構造体メモリに対するアクセスの潜在的な並列性を表す指標として、次のものを定義する。

$$\text{アクセス競合指数} \equiv \frac{t1}{t2} - 1$$

ただし、t1：各セグメントのSMアクセス時間の合計

t2：SMのリスト処理時間

t1,t2はリスト構造が１バンクに集中している場合のものである。

この値が大きい程、タスクの展開方法に潜在的なSMアクセスの並列性があることを示す。この値が0の展開方法には、構造体メモリに並列にアクセスできる部分がないことを意味する。

次に、実際的な評価量として、処理時間費を定義する。

$$\text{処理時間比} \equiv \frac{P1}{P2}$$

ただし、P1：リスト構造が１バンクに集中している場合の全タスク処理終了時間

P2：アクセス競合の起きないようにリスト構造が分散配置されている場合の全タスク処理終了時間

6. 性能評価

構造体メモリへのアクセス競合状態、およびアクセス競合によるタスクの処理時間への影響を検討することを目的として性能評価を行なう。実行するタスクとしては、FPのプログラム構成子であるconstruction、composition、および原始関数であるapndl、addを用いた。

表１　ハードウェアシミュレータへの機能の割付

ハードウェアシミュレータ	FPグラフリダクションマシン
上段のPE	SM,RC
下段左端野PE	PSC,1段目のセグメント
下段左端以外のPE	2~4段目のセグメント
下段のPE間のCM	セグメント間のパケット用キュー
中段左端のCM	RC-PSC間のパケット用キュー SM-第1セグメント間のパケット用キュー×2
中段左端以外のCM	SM-第1セグメント間のパケット用キュー×2

6.1シミュレーション環境

FPグラフリダクションマシンの構造体メモリ、簡約コントローラ、パイプラインスケジューラ、セグメントの機能は、各々ソフトウェアでハードウェアシミュレータ上に実現される。今回のシミュレーションでは、各部の機能を表1のようにハードウェアシミュレータ上に割り付けた。これらのプログラムは、全てVAX11/730上のＣ言語クロスコンパイラを用いて開発した。シミュレータ上で実行されるFPプログラムは、VAX11/730上でS式からリスト構造へ変換され、ハードウェアシミュレータの各PEにロードされる。

シミュレーションにおける時間の計測は、上段のPE内のタイマモジュールを用いる。図２における下段のPEは、上段のPEへ割込をかけることにより、必要なポイントでの時刻を上段のPEメモリ内に計測値として記録する。

6.2 シミュレーション方法

前提条件として、セグメント間のデータの転送時間を525μsec、セグメント-構造体メモリ間のデータの転送時間を525μsec、構造体メモリの平均命令処理時間を500μsec、セグメントのプロセス決定時間を375μsecとした。これらの値は、クロスコンパイラにより得られた機械語コードをそのまま実行したときの値である。これらの比率はそれ程偏っていないので、重み付けを行なわずにシミュレーションを行なった。セグメント数は4台とした。この条件のもとで、構造体メモリの１バンクに簡約すべきリスト構造が集中している場合と、5.1節で述べたようにリスト構造が分散配置されている場合についてシミュレーションを行なった。

構造体メモリの内容は、図2中の上段のPEのLM内部に格納される。中段の４台のCMは、リスト操作命令とデータの転送に用いられる。上段のPEは中段のCM内のリスト操作命令を読み、要求されたデータを各CMに置く。これにより、ネットワークを介した構造体メモリがCM上に仮想的に実現される。

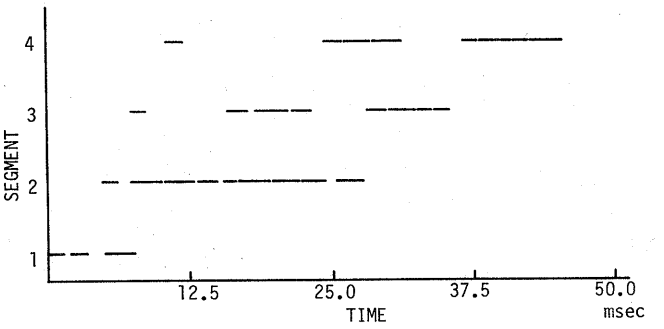

図9　constructionの構造体メモリアクセス状態

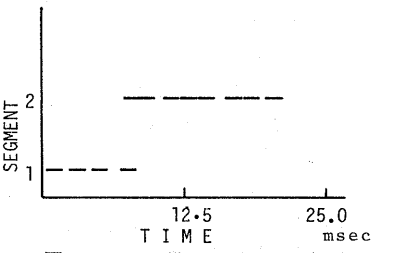

図11　apndlの構造体メモリアクセス状態

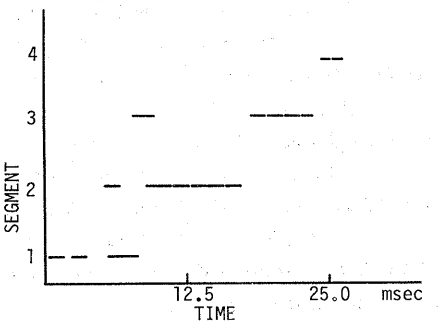

図10　compositionの構造体メモリアクセス状態

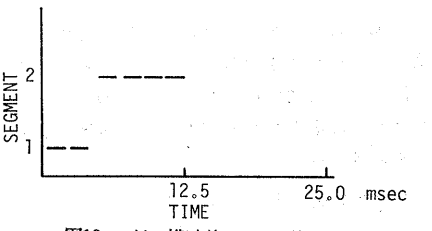

図12　addの構造体メモリアクセス状態

6.3 シミュレーション結果

図9～図12に、リスト構造が1バンクに集中している場合のconstruction、composition、apndl、addの構造体メモリへのアクセス状態を示す。constructionとaddの構造体メモリへのアクセス状態を比較すると、constructionでは構造体メモリに対して複数セグメントが同時刻にアクセスを行なっている箇所が存在するのに対し、addの場合全く存在しない。構造体メモリへのアクセス状態は、タスクの時空間への展開方法により大きく変化する。表2に、construction、composition、apndl、addのアクセス競合指数を示す。

FPグラフリダクションマシンでは、リストをたどるという逐次的な操作を時空間上へ展開し処理している。したがって、構造体メモリに対するアクセスに、より大きな並列性を見出せる展開方法が望ましい。今回用いたタスクの展開方法に限れば、construction、composition、apndl、addの中では、constructionが最もアクセス競合指数が高く、この中では最も構造体メモリに対するアクセスに並列性を見出せる関数例であることがわかる。

つぎに、各タスクについての処理時間比は次のようになった。

construction	1.23
composition	1.08
apndl	1.03
add	1.0

この結果は、リスト構造の分散配置による処理時間への影響の大きさを示している。アクセス競合指数、つまり展開方法に内在するメモリアクセスの並列性が大きいタスクほどリスト構造の分散配置の効果が現われている。すなわち、構造体メモリにアクセス競合が生じないようにリスト構造を分散配置することによって処理時間の短縮を得るためには、構造対メモリアクセスに多くの並列性が得られるようにタスクを時空間図上へ展開することが重要であることがわかる。

次に、同じ種類のタスクを連続して簡約エンジンに投入する場合を考える。図13は、処理時間比と投入するタスク数の関係を示している。1つのタスクのみを投入したときに比べ、全てのタスクで処理時間比が向上しており、リスト構造を分散配置した効果が現れている。この結果は、パイプライン処理部にタスクを連続して投入する場合、このリスト構造の分散配置方法が

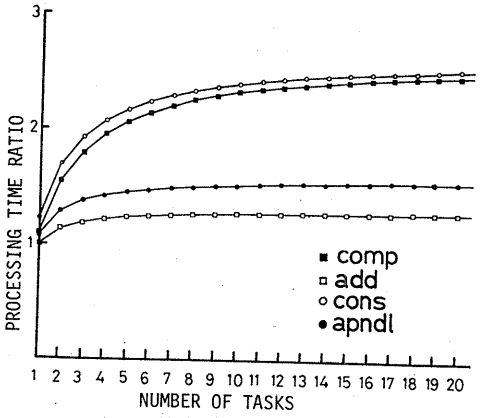

図13　処理時間比対タスク数

表2　アクセス競合指数

	アクセス競合指数
CONS	0.46
COMP	0.09
APNDL	0.06
ADD	0

さらに有用になってくることを示している。これは、各プロセスが他のタスクのプロセスと時間的に重なることで、構造体メモリに対するアクセスの並列性が増えたためである。

7. おわりに

FPグラフリダクションマシンでは、構造体メモリでのリスト操作の効率がシステム全体の効率を左右することが予想される。本稿では、構造体メモリの各バンクの並列性を生かし、リスト構造アクセス競合の生じないように分散配置する方法を示し、さらに構造体メモリへのアクセス競合がシステムの処理時間に与える影響について、ハードウェアシミュレータを用いて検討した。この結果、タスクの時空間への展開方法の良し悪しにより、リスト構造の構造体メモリのバンクへの分散配置による処理時間の短縮の度合が大きく影響を受けることがわかった。また、1回の実行ではほとんど分散配置による処理時間の短縮が見られないタスクについても、連続してREに投入することで処理時間の短縮が得られた。今後、より大きなFPプログラムを実行させ、より実際的な想定でシミュレーションを行なう予定である。

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)遠藤中村、重井："階層化汎用パイプラインシステム"，信学論(D),j68-D,7,pp.1376-1383(1985-07)

(3)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).

(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)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.

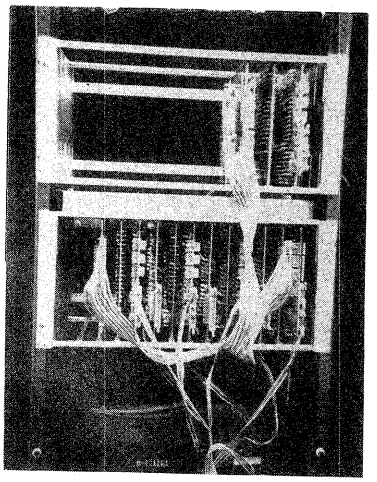

写真1　ハードウェアシミュレータの外観