計算機アーキテクチャ 45－1
（1982. 6. 1）

連想汎用レジスタを使用したデータフローコンピュータ

曽和将容
（群馬大学 工学部）

1. まえがき

要求駆動マシン[1]，リダクションマシン[2]など，データフロー系マシン[3]～[10]の研究が，特に，ここ数年盛んになってきており，マシンの試作も国内外で行なわれている。しかしながら，これらの試作は，小規模なシステムが大多数で，一時に可能な並列処理数が10^2～10^3程度の大規模システムになった場合には，問題がでてくる可能性のあるものも少なくない。これらの問題には，ハードウェアの複雑化，特に，処理ユニット間や処理ユニットとメモリを結ぶインタコミュニケーションネットワークの天文学的数字に近いような増大や，それに伴うオーバヘッド，および，高速化に伴って起るボトルネックなどである。

ボトルネックを避けるためには，処理系全体をできる限り並列処理可能にし，直列処理を含まないようにすることが必要である。しかしながら，これらを実現しようとするとハードウェアの増大を招き，ハードウェアの増大をさけようとすると直列処理が入りこむ。

本論文で述べるデータフローコンピュータ DAGR[11]（Data Flow Computer with a Associative General Purpose Multiport Register）は，連想汎用レジスタ（または，トークンメモリ）と呼ばれる概念を導入して，インタコミュニケーションネットワークの複雑化とボトルネックの問題を大きく前進させようとするものである。DAGRは，我々の研究室で試作が行なわれていたシステムで，基本動作の確認が完了したので報告する。

2. データフローコンピュータDAGRのプログラムとトークンパケット，ノードパケット

図1に示すノードが，DAGRで用いるデータフロープログラムグラフのノードの種類である。ノードへの，またはノードからの入出力アークが，最大4本に制限されていることを除けば，Dennis[3]らの言語とほぼ同じである。データフロープログラムグラフでは，ノードの中に，そのノードで行なわれる処理を表わすファンクションを書き，処理の順序をアークとよばれる矢印で表わす。ノードは，処理に必要なデータが入力アーク上にあり，出力アーク上にデータがないとき実行可能となる。処理に必要なデータはトークンとよばれ，データを表わすトークンをデータトークン（黒丸），プログラムのコントロールを表わすトークンはコントロールトークン（白丸）とよばれる。図1の破線で示されるアークは，コントロールトークン用であることを示す。

図2に示すようにデータフローコンピュータDAGRでは，1つのノードが

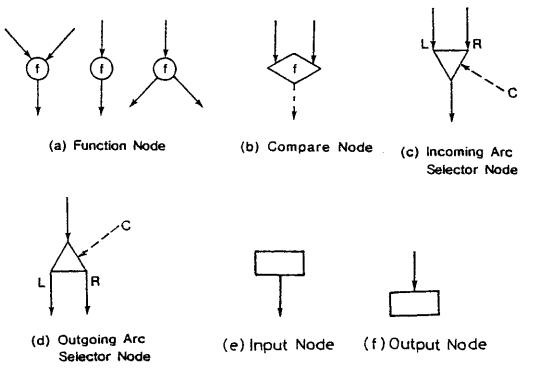


図1. ノードの種類

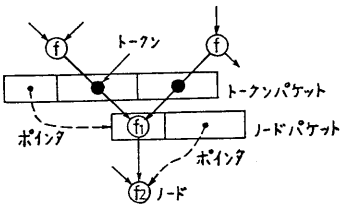


図2　データフロープログラムとトークンパケット，ノードパケット

必要とするデータと，そのデータが送られるべきノードへのポインタを一まとめにしたものを"トークンパケット"と呼ぶ．特に，1つのノードがそのファンクションを実行するために必要とするすべてのトークンがそろったトークンパケットを"完全トークンパケット(CTP)"と呼ぶ．また，ノードのファンクションと，そのファンクションが実行されたとき，結果を出力すべき次のノード(処理結果転送先ノード)へのポインタの集まりを"ノードパケット"とよぶ．

本コンピュータアーキテクチャでは，このトークンパケットとノードパケットを別々のメモリに格納することが大きな特徴で，これが故に，「ハードウェアを極端に複雑化，大容量化することなしに，並列処理を導入したデータフローコンピュータ」を構成できる．トークンパケットは，従来のコンピュータのアキュムレータに相当する，"トークンメモリ(TM)"(連想機能を持つ)とよばれるメモリに格納され，ノードパケットは，"プログラムメモリ(PM)"とよばれるマルチポートメモリに格納される．

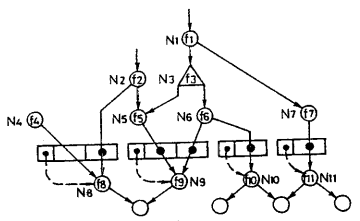


図3.　データフロープログラムの並列度

図3は，データフロープログラムグラフの実行中の状態を示す図である．図では，N9～N11の入力アーク上に完全トークンパケットがあるので"N9～N11が実行可能(ファイヤブル)となっており，N8も，あと左側入力アーク上にトークンが届けば実行可能ノードとなる．すなわち，このプログラムでは，並列に実行することのできるノードの最大数(プログラム上最大並列度とよぶ)は4である．実際には，これらのノードは，"ファンクショナルユニット(FU)"とよばれるプロセッサで実行されるので，たとえば，FUの数が2の場合には，並列に実行できるノードの数(実行最大並列度とよぶ)は2に制限されてしまう．すなわち，たとえ，プログラム上最大並列度が100であったとしてもファンクショナルユニットの数が50であったならば，実行最大並列度は50であり，FUの数を越えることはできない．1つのノードを実行するために必要なデータは，完全トークンパケットとしてまとめられているので，最低，FUの数だけ完全トークンパケットがあれば，システムは最大の効率を発揮することができる．実際には，FUは，完全トークンパケットの読出し，書込みとノードの実行を主に行うので，単純に考えて，これらの動作に全く同じ時間がかかるとすると，平均して，FUの数の1/3の完全トークンパケットがあればよいことになる．このことは，トークンパケットを格納するトークンメモリのワード数が少なくてもよく，TMのハードウェア量は，実行最大並列度にほぼ比例して増大することを意味する．もちろん，TMには不完全トークンパケットも格納されるので，TMの語数は多い方が

よいことは確かであるが，それほど膨大な量とはならないと考えられる。TMの語数が少なくてよいということは，TMを少し位複雑にしてもハードウェア量が極端に増加しないことを意味するので，TMに高度な機能を持たせることができるというメリットが生まれる。

3. データフローコンピュータ DAGRのアーキテクチャ

DAGRの構成は，図4のようであり，トークンメモリTM，プログラムメモリPM，ファンクショナルユニットFU，ホストコンピュータHCよりなる。TMは，トークンパケットを格納するメモリであり，第4章で述べるように，相互排他，連想および，マルチポートメモリの機能を持つ。プログラムメモリPMは，ノードパケット，したがって，データフロープログラムグラフを格納するメモリで，一般的なメモリがFUの数だけ用意されている。各メモリには同一のプログラムが格納されており，FUに対しては，読出し専用となっている。すなわち，PMは，FUに対して，読出し専用マルチポートメモリとなっている。ただし，ホストコンピュータHCにとっては，PMは全体で一枚のメモリであり，HCは，データフロープログラムグラフの書込みを行う。FUは，PM内のノードが持っているファンクションを認識し，実行する。データフロープログラムグラフへのデータの入力，または，プログラムグラフからのデータの出力は，FUがホストコンピュータHCに対し，入出力要求を出すことにより行なわれる。

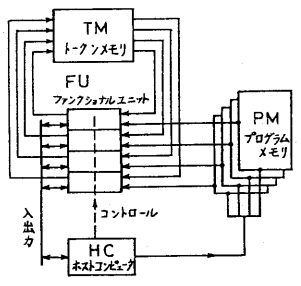


図4. データフローコンピュータ DAGRのアーキテクチャ

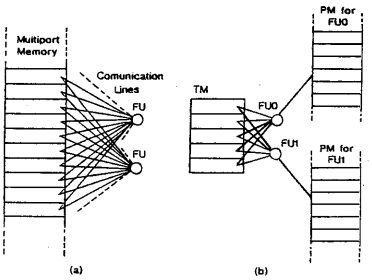
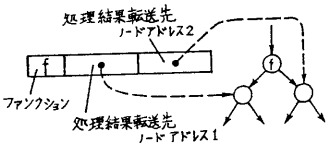


図5. インターコミュニケーションネットワーク

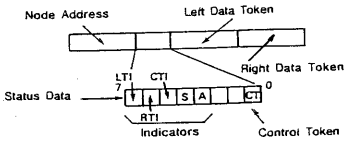
一般に，完全に並列処理を行うには，図5(a)に示すように，マルチポートメモリの各語と各FUとの間に通信線（インターコミュニケーションネットワーク），またはバスが必要であり，また，同一語のアクセス競合をさける相互排他回路も必要となる。メモリの語数は，プログラムの増大に従って増え，また一方，実行最大並列度を増すためには，FUを増さなければならないので，並列度の高い，大プログラムを実行できるようなデータフローコンピュータを構成しようとするとインターコミュニケーションネットワークを中心とするハードウェア量が膨大なものとなる。我々のアーキテクチャでは，同図(b)に示すように，語数の少ないTMの各ワードとFUとの間，および，各FUに与えられたPMとの間に1本の通信線が必要であるだけであるので，インターコミュニケーションネットワークを中心とするハードウェア量の増大をさけることができる。そして，このハードウェア量は，実行最大並列度

には影響されるがプログラムの長さには，原則として影響されない。

ノードパケットは図6(a)に示されるように構成され，トークンパケットは同図(b)に示すように構成される。トークンパケット内のインジケータLTI，RTI，CTは，それぞれ，左側入力アーク上のトークン（左トークン），右側入力アーク上のトークン（右トークン），コントロールトークンが，このトークンパケット上に格納されていることを示し，Sは，トークンパケット上に1つ以上のトークンがあることを，Aは，このトークンパケットがFUによってアクセス中であることを示す。



(a) ノードパケット



(b) トークンパケット

図6 ノードパケット，トークンパケットのフォーマット

4. トークンメモリ

図7にトークンメモリTMの構造を示す。TMは，VWA，CTA，NAM，SDM，LDTM，RDTMよりなっており，NAMにはトークンパケットのノードアドレスが，SDMにはインジケータとコントロールトークンを含むステイタスデータが，LDTMにはレフトデータトークン，RDTMにはライトデータトークンが格納される。VWAは空ワード割

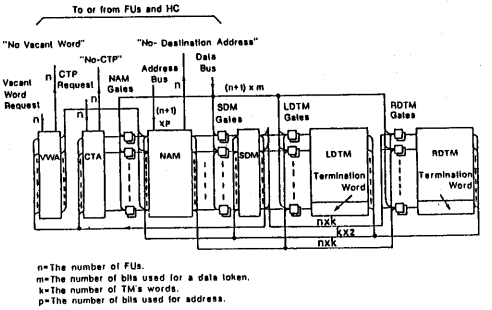


図7. トークンメモリTMの構造

付けモジュールと呼ばれるモジュールで，FUからの空ワード要求に対して，NAM，SDM，LDTM，RDTMからなる空ワードを1語割付ける。CTAは完全トークン割付けモジュールと呼ばれるモジュールで，FUからの完全トークンパケット要求に対して1つの完全トークンパケットを割当てる。

ノードアドレスメモリNAMは連想メモリであり，ノードアドレスを指定することによって，そのノードアドレスが格納されている語をアクセスすることができる。この機能は，実行を行おうとするノードの出力アーク上にトークンがあるかどうかを調べる，いわゆる，"セイフティチェック"を行うために用いられる。なお，TMの各モジュールへのアクセスは，複数のFUによって同時に行なわれることが普通であるので，全てのユニットは，並列処理可能なマルチポート構成となっている。

5. 実行アルゴリズム

FUは，完全トークンパケットCTPをTMより取出し，CTP内のノードアドレスをもとに，プログラムメモリPMから，実行すべきファンクションと実行結果を送るべきノード（処理結果転送先ノードとよぶ）に関する情報をえる。次に，実行すべきノードの出力

アーク（処理結果転送先ノードの入力アーク）上に，トークンがあるかどうか調べるために，ノードアドレスメモリNAMに処理結果転送先ノードアドレスを出力し，セイフティチェックを行う．セイフならばノードのファンクションを実行し，その結果を処理結果転送先ノードアドレスと接続し，トークンパケットとする．その後，TMに空ワードを要求し，そのトークンパケットをTMに格納する．詳細なアルゴリズムは次のようである．

1. FUはCTAにCTPを要求する．
2. CTAはCTPがあればNAM，SDM，LDTM，RDTMゲートを閉じFUに割当てる（このとき，割当てた語のSDMインジケータAを1とする）．なければ，"no-CTP"信号をFUに出す．
3. もし，FUがno-CTP信号をうけとったなら，ステップ1に行く．そうでないなら，FUはCTPのノードアドレスを読み，そのノードに関する情報をPMから読みとる．
4. FUは，処理結果転送先ノードアドレスをNAMに送りセイフティチェックを行う．
5. もし，NAM上に，いま送られてきた処理結果転送先ノードアドレスがなければ，NAMは，"処理結果転送先ノードアドレスなし"の信号をFUに送る．もし，処理結果転送先ノードアドレスがあったら，そのFUにSDMゲートを開き，SDMとFUを接続する．
6. もし，FUが"処理結果転送先ノードアドレスなし"の信号をうけとったならば，ステップ7に行く．そうでないならば，SDM上のステイタスデータを読み，実行しようとしているノードの出力アーク上にトークンがあるかどうかをSDMのインジケータから調べる．もし，あったら，セイフではないので，ステップ1に行く．もし，なかったらステップ7に行く．SDMゲートは，このステップの終りで閉じられる．
7. FUは実行しようとしているノードの出力アークが他にないかを調べ，あったならばステップ4に行く．なかったならば，FUは，実行しようとしているノードの入力アーク上にあるトークンパケットの左，または右のデータを読出す．臨時的に，このトークンパケットが格納されている語のゲートを閉じる．
8. FUは，ノードファンクションを実行し結果トークンに，処理結果転送先ノードアドレスを接続し，トークンパケットとする．次に，このトークンパケットをTMに格納するために，空ワード割付けモジュールVWAに空ワード要求を出す．
9. VWAはTM内の空ワードを探し，もし，あれば，空ワードをそのFUに割付ける．なければ，"空ワードなし"信号"no-vacant word"を出す．"空ワードなし"信号を受取ったFUは，ステップ8に行く．
10. FUは，NAMに格納すべきトークンパケットのノードアドレスと，SDMにステイタスデータ（S=1，A=1，CTIまたはLTI，RTIのうち1つが1）を書く．
11. VWAは，いまNAMに書かれたノードアドレスと同じノードアドレスが，以前，または，いま，同時に格納されたかどうかをチェックする．もし，同じノードアドレスが格納されていなかったらステップ13へ行き，あれば，同一ノードアドレスを持っている語の中から，次の優先順位に従って，1つの語を選ぶ．
    1. このノードアドレスが書かれる前に，すでに格納されていた不完全トークンパケットを格納してい

る語
2. コントロールトークンを格納したばかりの語
3. 左側データトークンを格納したばかりの語
4. 右側データトークンを格納したばかりの語

12. VWAは，同じノードアドレスを持つ語のSDMの内容を，ステップ11で選ばれた語にコピーする．
13. 左，または，右側データトークンの格納が要求されておれば，左，または，右側データトークンメモリのゲートを開きFUに接続する．そうでないならばステップ15に行く．
14. FUは，左，または，右側トークンをデータメモリに書込み，ターミネーションワードをアクセスする．
15. VWAは，いま書込んだ語の全てのゲートを閉じ，FUと語を分離する．
16. VWAは，FUの要求によりステップ7で臨時的に閉じられたゲートを開き，インジケータ部のSとAを0にし，この語を空ワードとした後，すべてのゲートを閉じる．
17. ステップ1へ行く．

なお，外部入出力機器からプログラムへのデータ入力は，FUがステップ3を数回操返した後要求駆動方式で行なわれ，データ出力は，出力ノードの実行によって行なわれる．

6. むすび

本報告では，メモリをTMとPMにわけることにより，ハードウェア量を極端に増やすことなく，トークンの取出し，セイフティチェック，ファンクションの実行，結果トークンの格納を並列に処理することのできるデータフローコンピュータのアーキテクチャを示した．なお，本コンピュータで扱う処理のレベルは特に固定されておらず，命令レベル，プロシジャーレベル，もしくは，これらをミックスしたレベルでも使用可能である．このため，ブロック駆動[10]のような考え方で処理を実行することも可能であり，オーバヘッドの改善を期待することができる．

文献

(1) B. Jayaraman, R.M. Keller: "Resoure control in demand driven data-flow model", Proc. of International Conference on Parallel Processing, pp.118-127, Aug. 1980.
(2) K. Berkling: "Computer architecture for correct programming", Proc. of the 5th Symposium on Computer Architecture, pp.78-84, Apr. 1978.
(3) J.B. Dennis, D.P. Misunas: "A preliminary architecture for a basic data flow processor", 2nd Annual Symp. on Computer Architecture, IEEE, N.Y., pp.126-132, 1975.
(4) A.L. Davis: "The architecture of DDM1; A recursively structured data driven machine", tech. Reports UUCS-77-113, Department of Computer Science, University of Utah, Oct. 1977.
(5) J. Rumbaugh: "A data flow multiprocessor", IEEE Trans. on Computers, Vol. C-26, No.2, pp.138-146, Feb. 1977.
(6) Arvind, K.P. Gostelow: "Data flow computer architecture; Reserch and gols", Tech. Report 133, Department of Information and Computer Science, University of California Irvine, Feb. 1978.
(7) A.D. Plas, O. Comte, O. Gelly, J.C. Syre: "LAU system architecture; Parallel data-driven processor based on single assigments", Proc. of the 1976 International Conference on Parallel Processing.
(8) J.R. Gvand, I. Watson, J.R.W. Glauert: " A multilayered data flow computer architecture", Department of Computer Science, University of Manchester, July 1978.
(9) M. Sowa, K. Hayakawa: "Procedure level data flow computer system -GMMCS-", Paper of Tech. Group Computer Architecture 36-1, IIP Japan, 1979.
(10) T.L. Chang, P.D. Fisher: " A block-driven data-flow processor", Proc. of the 1980 International Conference.
(11) M. Sowa, T. Murata: "A data flow computer architecture with program and token memories", Communications Lab. Report 80-2, University Illinois Chicago, March 1980.