計算機アーキテクチャ 59−8
オペレーティング・システム 29−8
(1985・10・24)

科学技術計算用データ駆動計算機 ＳＩＧＭＡ－１ 予備試作版の性能評価

An evaluation of a single processorof a data flow computer SIGMA-1 for scientific computations

島田俊夫 Toshio Shimada　平木　敬 Kei Hiraki　関口智嗣 Satoshi Sekiguchi　西田健次 Kenji Nishida

電子技術総合研究所
Electrotechnical Laboratory

1　まえがき

データ駆動方式はプログラムの持つ並列性を自然に引き出すという特徴があり、従来のマルチプロセッサ方式の困難さを克服する方式として、１９７０年頃から各国で活発な研究が行なわれてきた［１～９］。

これらのプロジェクトのうちの幾つかは、実際にハードウェアを製作したが、これらはデータ駆動方式の潜在能力を示すだけの速度や規模を持っていないものが大部分であった。このため、データ駆動計算機の実用性に対して懐疑的な批判も出ている。これらの批判はデータ駆動方式の１台のプロセッサに関するものとデータ駆動方式のマルチプロセッサに関するものの２種類に分けることができる。

１台のプロセッサに関するものは次の２つである。

１．　レジスタがないため、データアクセス時間が増し、プロセッサ部分の性能が遅くなる。そのため並列性の低い問題はフォンノイマン型に比べて性能が低下し、広い範囲の応用に適用できない。

２．　データの待ち合わせを行う機構が複雑であり、性能低下の原因となる。また、　ハードウェア量の増大を招く。

マルチプロセッサに関するものは次の２つがある。

３．　メモリへのアクセスが集中して性能が低下するというマルチプロセッサシステムに共通の問題をデータ駆動計算機も抱えている。

４．　ジョブアロケーションを効率良く行う方法がない。

これらの批判に対しては以下のように答えることができる。

１．　並列性の低い場合には計算したデータを直ちに利用する場合が多い。この場合データ駆動計算機はパイプラインの遅れがあるのでデータアクセス時間が増大する。しかしこの遅れを小さくするようアーキテクチャを設計することができる。ＳＩＧＭＡ－１はこのことを考慮してショートパイプラインのアーキテクチャを採用している。またデータ駆動計算機はレジスタがないためデータが一方向に流れ、ハードウェアがシンプルになるのでマシンサイクルを速くすることができる。そのため十分フォンノイマン型に対抗できる性能が発揮できる。

２．　待ち合わせ機構をフルアソシアティブの連想メモリで作った場合速度は速いが現在の技術

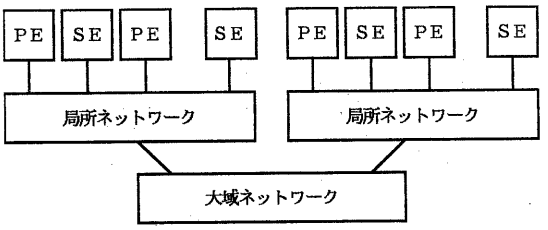


図1　ＳＩＧＭＡ－１の全体構成

では大容量を達成できない。しかし、連鎖ハッシュ方式を用いた場合にはハードウェア量も少なく高速の待ち合せ機構が実現できる［１０］。

３．　フォンノイマン型ではメモリアクセスを行えばデータをフェッチしてくるまで実行が中断される。これに対してデータ駆動計算機はパケットを出してデータを取りに行くので、データを取ってくるまで待つことなく別の仕事をすることができる。並列性が十分ある時は別の仕事が十分あるのでその間プロセッサは仕事を行うことができ性能低下は起こらない。

４．　ジョブアロケーションは応用問題と密接にかかわっているので問題に応じてアロケーション方式を考えなければならない。ＳＩＧＭＡ－１では関数単位でジョブアロケーションを行っているが、関数は副作用がないので実行時にジョブを割り付けるプロセッサを選択することが容易である。このため負荷を自動的に分散するための機構を容易に実現できる利点がある。

以上のように批判に答えることはできるが、最終的には実用的なデータ駆動計算機を製作し、その性能評価を行うことが必要である。この目的のために我々は２００台規模のシステムで平均速度１００ＭＦＬＯＰＳを目指したデータ駆動計算機ＳＩＧＭＡ－１を１９８２年より開発中であり、現在予備試作版基本プロセッサ１台が稼動中である。今回はこの基本プロセッサの性能評価を行う。

２　ＳＩＧＭＡ－１の構成

ＳＩＧＭＡ－１は科学技術計算を指向したデータ駆動計算機で、全体の構成は問題のローカリティを効率良く処理するように階層型のネットワークで接続されている（図１）。局所ネットワークはクロスバーで、４台の演算処理装置（ＰＥ）と４台の構造体処理装置（ＳＥ）を接続しており、これをグループと呼んでいる。大域ネットワークは各グループを接続しており、局所ネットワークを多段接続して構成する。

ＰＥはデータ駆動方式の計算機であり、構造メモリへの操作以外の全ての演算を処理する。ＰＥの構成は図２に示すように２段のパイプライン構成をしており、全体が同期式で動作する。Ｂユニットはネットワークとのインタフェースであり、４ＫＷのバッファメモリを持っている。Ｆユニットは１６ＫＷのプログラムメモリを持ち、実行するプログラムをストアする。Ｆユニットは入力トークンのタグから行く先命令のアドレスを計算し、その命令をフェッチし、Ｅユニットに送る。Ｍユニットは２入力命令のうち最初に到着した入力トークンをストアしておき、対となるトークンが到着したときに、タグを用いて最初に到着したトークンを取り出す連想メモリである。ＳＩＧＭＡ－１はこの連想に連鎖ハッシュ方式を用いて効率の良い連想メモリを実現している。ＥユニットはＡＬＵで算術論理演算を行う。浮動小数点演算は３２ビットである。Ｄユニットは行く先命令のアドレスやタグを計算しＥユニットの計算結果をパケット形式とし、出力トークンとして送り出す。

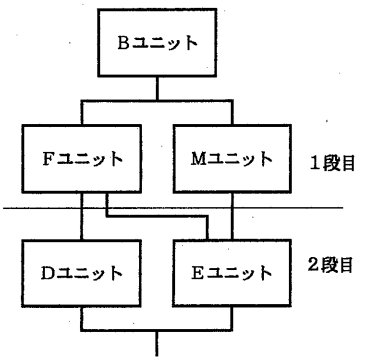


図２　ＰＥの構成

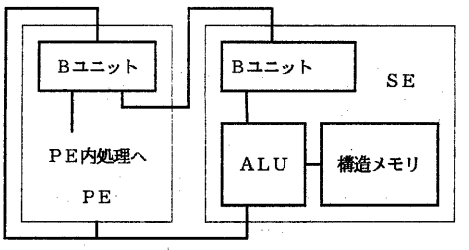


図３　ＳＥの接続と構成

```
trapezoidal (n)
int  n;
{int  i;
 float  sum, x, y;
 for (i=0, sum=0, x=0. ; i<n ; new  i=i+1)
        {new  sum=sum+x*x:
         new  x=x+0. 1;}
 return ((new  sum+0. 5) *0. 1) ;}
```

図4　TRAPEZOIDALのＤＦＣプログラム

ＳＥはＡＬＵと６４ＫＷのメモリを持ち、空きメモリとＲＥＡＤパケットの待ち行列の管理を行う。予備試作版ではネットワークがないのでＳＥはＰＥのＢユニットに接続されている（図３）。

命令名	直列実行時間	並列実行時間
ADD	3	3
MUL	4	3
DIV	15	14
FADD	5	3
FMUL	5	3
FDIV	16	15
GE, EQ	3	3
SWITCH	4～5	3～4
DIST	3	2
CALL	4	3
RETARG	4	3

表1　命令の実行時間

3　ＳＩＧＭＡ－１のプログラム

ＳＩＧＭＡ－１の高級言語はＣ言語に単一代入規則を導入したＤＦＣ言語である。Ｃ言語との相違は単一代入規則に基づくもので、ループ内での配列と変数の使用に注意が必要である。プログラムの例を図４に示す。これはx^2の積分を台形則で求めるものである。

高級言語のコンパイラは現在簡単な構造の文をコンパイルできる段階である。その出力コードは冗長で効率が悪いため、今回の測定で使用したプログラムは次に述べるアセンブラで記述したものを用いた。

アセンブラＳＡＳの形式を次に述べる。

ラベル（命令　［定数］行く先アドレスのリスト）

ラベルはその関数内ではユニークでなければならない。命令はＳＩＧＭＡ－１の命令名である。定数を持つ命令はここでその値を書く。定数のタイプとして整数、浮動小数点、アドレス等７種類がある。行く先アドレスはＰＥ番号、行く先ラベル、マッチングフラグで構成される。実際の命令では行く先アドレスは最大３個までしかゆるされないがアセンブラではその制限はない。上記のプログラムＳＡＳで記述した例を図５に示す。

4　予備試作版ハードウエアの性能

ＰＥは４層のプリント基板６枚で実現されており、８０ｎｓのクロックで動作可能である。ＳＥは両面基板１枚にラッピングで実現されており、１５５ｎｓのクロックで動作する。今回の評価はＰＥとＳＥの両方を用いるので１５５ｎｓのクロックで動作させた。

予備試作版の主要な命令の実行時間を表１に示す。この表から予備試作版の速度は０．４３～３．２ＭＩＰＳまたは０．４～２．１ＭＦＬＯＰＳということになる。しかし良く使用される命令は３～５クロックの範囲にあるので予備試作版の実効速度は１．３～２．１ＭＩＰＳ（ＭＦＬＯＰＳ）

```
;-------------------------------------------------------------------------
;     y=x*x
;     x=x+h
;-------------------------------------------------------------------------
sw2:    (*SWITCH    (true (paf mull: normal_l) (paf mull: normal_r)
                          (paf faddl: pairs_r))
                    (false (paf syn: normal_r)))
mull:   (*FMUL      (paf fadd2: normal_l))                  ;y=x*x

faddl:  (*FADD      (paf i_incl: sl))                       ;x= x+h h=0.1
i_incl: (*I_INC     (paf sw2: normal_r))
```

図5　trapezoidalのＳＡＳプログラムの一部

である。

5　性能評価

5．1　測定方法

ＳＩＧＭＡ－１予備試作版の評価として表２のプログラムを用いた。測定は対象プログラムの始めと終わりに特殊な命令を挿入し、その命令間の実行時間、実行命令数をロジックアナライザで測定した。

5．2　測定項目

測定項目はプログラムのコードサイズ、実行命令数、ｆｌｏａｔ命令数（浮動小数点命令数）、ＭＩＰＳ値、ＭＦＬＯＰＳ値である。これらの項目の定義は以下の通りである。

コードサイズ：　マシンにロードされたプログラムをバイト数で測ったものである。ＳＩＧＭＡ－１の命令は５バイト長である。

実行命令数：　マシン上で実行した命令数である。

ｆｌｏａｔ命令数：浮動小数点命令数をプログラムから手計算で求めた。

ＭＩＰＳ値：実行命令数／実行時間

ＭＦＬＯＰＳ値：　実行浮動小数点命令数／実行時間

5．2　ループプログラムの評価

プログラム１は構造メモリを使用しないプログラムであり、構造メモリを使用するものと比較するため用いた。プログラム１でのＳＩＧＭＡ－１予備試作版の性能は１．４１ＭＩＰＳであり、他のプログラムによる結果と大差ない。このことは構造メモリの使用による性能低下は無いことを意味している。

プログラム２～９はパイプライン方式の計算機の性能を測定するためにLowrence Livermore研究所が提案したベンチマークプログラムLivermoreループの中の幾つかを抜き出したものである。これらのプログラムは幾つかの商用スーパコンピュータでの性能評価が報告されているので［１２］、ＳＩＧＭＡ－１の現状を知り、将来を見通す上で有用である。ＳＩＧＭＡ－１はベクトル計算も並列処理可能であるが、今回は１台のＰＥによる評価のため並列処理については言及しない。この点については別の機会に報告する。

最初に４節で述べたＳＩＧＭＡ－１のハードウエア性能をもとに、プログラムを２つ解析し理論的に可能な最大性能を求め、ＳＩＧＭＡ－１が理論通り動作しているかを確かめる。ＳＩＧＭＡ－１はパイプライン構成をしているのである状況ではパイプが詰まらず遅れを生じる。それを以下に述べる。

ＳＩＧＭＡ－１のＰＥでは２入力命令の２個の入力トークンのうち最初の１個が到着したときは、そのトークンはマッチングメモリにストアされる。したがってその次のマシンサイクルではＥユニットはアイドルすることになる。即ち２入力命令が多いとＭＩＰＳ値は最大５０％まで低下する。Ｅユニットがアイドルするもう１つの状態は出力トークンの数が２個以上あるときに起こる。１個の出力トークンを送り出すのに２クロックかかり、トークンを送り出している間はＥユニットがアイ

プログラム名	サイズ(BYTE)	実行命令数	float命令数	ＭＩＰＳ値	ＭＦＬＯＰＳ値
1 TRAPEZOIDAL	200	1207	300	1.41	0.35
2 流体1	370	8443	2000	1.39	0.33
3 MLP内積2	605	6832	2000	1.25	0.37
4 内積3	230	11034	2000	1.34	0.24
5 三角化消去5	655	12399	1998	1.24	0.20
6 三角化消去6	690	12735	1998	1.26	0.20
7 状態方程式7	705	5590	1920	1.42	0.49
8 総和11	290	15023	1000	1.32	0.088
9 差分12	285	15037	1000	1.32	0.088
10 階乗ループ	45	126	0	1.61	0
11 階乗再帰	55	220	0	1.33	0

表2　プログラムの評価結果

ドルするので２個目の出力トークンからは２クロックづつアイドルする。

これをプログラム１で実際に計算する。プログラム１は図４に示したものである。プログラム１のループ内の命令数は１１である。１１命令の並列実行時間の和をＴ１とすると、

Ｔ１＝Σ各命令の並列実行時間＝３３クロック

である。

このうち２入力命令は５個あるのでこれらの命令への最初の入力トークンのためのアイドル時間をＴ２とすると１マシンサイクルは３クロックなので、

Ｔ２＝３×５＝１５クロック

である。

ループ内命令の出力トークンの総数は１６個である。このうち１１個の出力トークンは実行時間とオーバラップして送り出されるが、残りの５個は２クロックづつの遅れを作り出すのでその合計Ｔ３は、

Ｔ３＝２×５＝１０クロック

である。

このループを１回実行するのに要する時間Ｔ４は、

Ｔ４＝Ｔ１＋Ｔ２＋Ｔ３＝５８クロック

である。ループは１００回実行したので総クロック数Ｔ５は

Ｔ５＝５８×１００＝５８００クロック

である。

１クロックは１５５ｎｓなので総実行時間Ｔ６は、

Ｔ６＝１５５×５８００

＝８９９０００ｎｓ＝０．８９９ｍｓ

となる。実際の実行時間は０．８５９ｍｓなのでその差は０．０４０ｍｓである。その理由は遅延する出力トークンが２入力命令の最初の入力トークンであった場合には遅延とならないので、これがループの繰り返しごとに１回起こったと考えられる。Ｍユニットのマッチングメモリでトークンが衝突し、連鎖を作ると遅延の原因となるがこのプログラムでは起こっていない。また今回の測定に用いたプログラムの規模ではこのようなことは殆ど起こらなかった。

次にプログラム２を解析する。プログラム２はＬｉｖｅｒｍｏｒｅループ１の流体の計算をするプログラムで、ＤＦＣによる記述は図６の通りである。この図で大文字は定数を表す。

プログラム２がプログラム１と異なる点はＳＥが使用される点である。プログラム２のＰＥが実行するループ内の命令数は２１で、ＳＥが実行する命令数は３である。ＳＥとＰＥは並列に動作するが、今回は図３のような接続を行っているので出力が衝突し逐次実行と同じ状況になっている。

総並列実行時間Ｔ１は

Ｔ１＝７０クロック

である。このうち２入力命令は６個あるのでこれによるアイドル時間Ｔ２は

Ｔ２＝３×６＝１８クロック

である。

ループ内命令の総出力トークン数は２６でこのうち５個は遅れの原因となるのでその時間Ｔ３は

Ｔ３＝２×５＝１０クロック。

このループを１回実行するのに要するクロック数Ｔ４は

Ｔ４＝９８クロック

である。このループは４００回実行したのでその総クロック数Ｔ５は

Ｔ５＝９８×４００＝３９２００クロック。

１クロックは１５５ｎｓなので総実行時間Ｔ６は

Ｔ６＝１５５×３９２００

＝６０７６０００microsec＝６．０７６ｍｓ

```
ryuutai (n)
int  n;
 {int  k;
 float  x (400) , y (400) , z (411) ;
    for (k=1;k<n;new  k=k+1)
 x (k) =Q+y (k) * (R*z (k+10) +T*z (k+11) ;}
```

図６　流体のＤＦＣプログラム

で、実際の実行時間とほぼ一致している。その差はプログラム１のところで述べたように出力トークンの遅れと２入力命令の最初のトークンの遅れが重なったものと考えられる。

以上の考察からＳＩＧＭＡ－１はほぼ理論どうりの動作をしていると結論できる。

さらにこの考察によれば２入力命令の多いプログラムではＭＩＰＳ値が低下することになる。表２の測定結果によれば予備試作版のＭＩＰＳ値は１．２５～１．４２であり、その変動は１３％と小さい。

この値は他の実験でも１６％であり［１３］、このことはプログラムにおける２入力命令と１入力命令の比率がそれほど変動しないことを予想させる。

データから１～９のプログラムにおける予備試作版の平均速度は１．３２ＭＩＰＳであることがわかる。この値は４節で述べた性能（１．３～２．５ＭＩＰＳ）低い方に近い。その理由は上の解析で述べた。データ駆動計算機では命令の実行速度と実際のプログラムによる実効速度とはかなり異なるのである。

５．３　ループのオーバヘッド

今後の性能向上への指針を得るためループの実行内容を分析した。対象とするプログラムは１である。このプログラムのループ内の命令の数は前にも述べたように１１個である。その内訳は以下の通りである。

（ａ）　ループ制御　　４命令
（ｂ）　変数がループを回るための制御　４命令
（ｃ）　算術演算　　３命令

７０％以上の命令が制御に費やされており、制御のオーバヘッドがかなり大きい。プログラム４、９、１０のように浮動小数点演算の数が少ないプログラムを高速に実行するためにはループ制御の効率化が必要であることが分かる。

ＭＩＰＳ値を計算するときはこのオーバヘッドは実行命令にカウントされるので性能低下の原因となっていないように見えるが、これはオーバヘッドであるので実際には性能が低下しているのである。このことからＭＩＰＳ値はハードウエアの性能は示すが、必ずしもデータ駆動計算機のアーキテクチャの評価を表していないことが分かる。一方ＭＦＬＯＰＳ値の計算では上述のオーバヘッドは実行命令数に含まれないので、正しくデータ駆動計算機の実効性能を表しておりフォンノイマン型の計算機との比較にも用いることができる。しかしＭＦＬＯＰＳ値はプログラムに浮動小数点演算が含まれていないときは測定できないし、プログラムによって大きく変動するので必ずしもハードウエアの能力を表していない。このことはプログラム８と９を比べてみると良く解る。プログラム８はループ内に１６の浮動小数点演算があるが、プログラム９は１個しかない。そのため同じハードウエアでもＭＦＬＯＰＳ値に大きな差がある。しかしベンチマークプログラムを定めて相互に比較を行う時は良い指標である。

５．４　ｓｔｉｃｋｙトークンの評価

科学技術計算ではループ構造は非常に重要な位置を占めている。現在スーパコンピュータに採用されているパイプラインアーキテクチャはループの中のベクトル計算を高速に処理する方式であるが、この種のベクトル計算は科学技術計算の中でかなりの割合を占めている。しかし科学技術計算にはその他にスカラーレベル、関数レベルの並列性を持った問題も数多くある。ところがこれらの計算もプログラム上ではループとして記述されることが多い。それゆえに科学技術計算を対象とする計算機ではループ構造を効率良く処理することが大切である。ＳＩＧＭＡ－１はこのことを考慮しループインバリアントに特別の対策を行っている。

ループインバリアントとはループ外で定義され、ループ内ではその値を使用するだけでその値は変えない変数のことである。ＳＩＧＭＡ－１は関数型言語を使用するが、この言語では大域変数は存在しない。一方ループは１個の関数として処理されるので外部で定義された変数を内部で使用するときは引数としてループに渡さなければならない。ところがデータ駆動方式ではメモリの概念がない

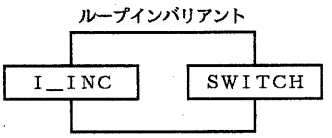


図7（a）　ループインバリアントの循環による保持

のでこれらの引数のうちループインバリアントとなるものをストアしておく場所がない。そこでデータフローグラフ内を循環させてループインバリアントを保持する方法が用いられる。しかしこの方法は図７（ａ）に示すように１個のループインバリアントについて２命令が必要であり、ループ内で多くのループインバリアントがある場合著しく効率をそこねる。

ＳＩＧＭＡ－１ではループインバリアントはｓｔｉｃｋｙトークンを用いて前述のオーバヘッドを無くしている［１１］。

データ駆動方式では、命令の入力アークのすべてにトークンが到着するとその命令は発火し、入力アークのトークンは消滅する。入力トークンの一方がｓｔｉｃｋｙトークンの場合はｓｔｉｃｋｙトークンは消滅せず、その入力アーク上に残り続ける。この様子を図７（ｂ）に示す。ｓｔｉｃｋｙトークンはＳＩＧＭＡ－１のマッチングメモリに保持され、その処理はＭユニットのタグの連想処理の中で行われるのでオーバヘッドは生じない。ｓｔｉｃｋｙトークンはループ終了時に消去する必要がある。これはループ終了時にｕｎｓｔｉｃｋトークンをｓｔｉｃｋｙトークンとマッチさせて行う。

次にｓｔｉｃｋｙトークンの効率向上を解析する。ループインバリアントがｋ個あるとする。ｓｔｉｃｋｙトークンをｋ個用いた場合はループ内に余分な命令を付加する必要はない。しかもｋ個の命令は本来２入力命令であるがｓｔｉｃｋｙトークンが入力アークにくっつくと１入力命令となりマシンの遅れが２ｋクロック減り、生成するトークンの数もｋ個減る。ループ終了時にｓｔｉｃｋｙトークンを消去するためｋ－１個の命令が必要と

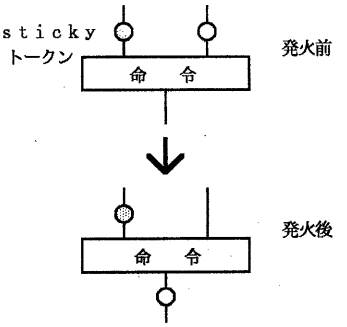


図7（b）　ｓｔｉｃｋｙトークン

なるがこれは１回実行するだけである。ｓｔｉｃｋｙトークンのオーバヘッドはループ回数に関係なくｋ－1命令である。

一方ｓｔｉｃｋｙトークンを使用しない場合はループ内に含まれる命令の数は図７（ａ）に示すように２ｋ個増える。さらにｉ＜Ｎの命令はｋ個のＳＷＩＴＣＨ命令にＴ／Ｆを配らなければならないので、これらのトークンを出力するＤＩＳＴ命令が

「（ｋ－１）／２

個増える。この場合のＳＷＴＣＨ命令は２個出力トークンがあるので実行時間は６クロック、Ｉ_ＩＮＣ、ＤＩＳＴ命令の並列実行時間がそれぞれ４、３、２クロックである。したがって

６ｋ＋３ｋ＋２「（ｋ－１）／２　＋Ｋ

＝１０Ｋ＋２「（ｋ－１）／２

である。最後のｋはＳＷＩＴＣＨ命令が２入力命令であるための遅れである。

これを表２のプログラム１０をとりあげて解析結果を確かめる。このプログラムのＤＦＣで記述

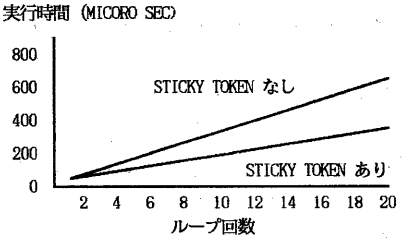


図8　ＳＴＩＣＫＹトークンの効果

すると次のようになる。

```
for (i=1; i<N; new i=i+1)
    x(k) = y(k+1) - y(k);
```

このプログラムはｓｔｉｃｋｙトークンが４個使用されている。１個はループ終了を検出するｉ＜ＮのＮである。Ｎの値はこのｆｏｒ文に入る前に定義され、このループ内では定数として扱われる。他の３個は配列ｘ、ｙのデータをフェッチする際のアドレスでそれぞれｘ（ｋ）、ｙ（ｋ＋１）、ｙ（ｋ）のＦＥＴＣＨ命令に使用される。

このプログラムのループ内の命令数はｓｔｉｃｋｙトークンを使用する場合１５である。総命令実行時間Ｓ１＝６８クロックである。

ｓｔｉｃｋｙトークンを使わない場合は総命令数２５となり、総命令実行時間Ｔ１＝１１２クロックである。先の解析によればその差は４４クロックであるのでその結果は一致している。ループ回数Ｎを１から２０まで変化させて実行時間を測定した結果を図８に示す。

ループ回数Ｎが十分大きいときその実行時間の比はＴ１／Ｓ１＝１．８１となる。Ｎ＝２０のときこの比の実測値は１．７７であり、Ｎ＝１０００のときこの比の実測値は１．８６である。なおＮ＝１の時でもｓｔｉｃｋｙトークンを使用する時の命令数は２５で、使用しない時の２８より少ない。

５．５　ループとリカーションの比較

プログラム１１、１２ではループとリカーションの比較を行った。階乗のループではループ制御に５命令、変数がループを回るのに２命令、算術演算が１命令なので繰り返し回数をｎとすると実行時間Ｔ１は８ｎのオーダとなる。

一方階乗のリカーションは関数呼び出しを行う。関数呼び出しは環境の設定に６命令、変数の受け渡しに２命令、値を返すのに１命令が必要である。実行命令数は１６ｎのオーダとなる。したがってデータ駆動計算機でもループよりリカーションの方がオーバヘッドが大きい。図９は階乗の計算のｎ＝２～１３をループとリカーションで実行した結果を示している。リカーションはループの約２倍の実行時間がかかっている。

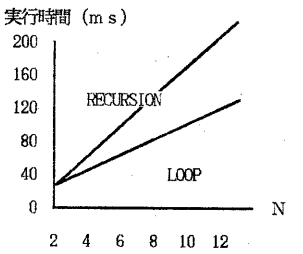


図９　ループとリカーションによる階乗計算の比較

６　アーキテクチャ改良の方法

今回の評価から明らかになった点について改良の方法を示す。

６．１　ループ制御

ループ変数生成はＳＩＧＭＡ－１の高級言語ＤＦＣでは、

```
for (i=1; i<n; new i=i+1)
```

と表現される。これは機械語では４命令で実現される。ループ内部の処理が少ないプログラムではこれは大きなオーバヘッドとなる。これを１命令で実現すればオーバヘッドを減少できる。この命令は１マシンサイクルごとにループ変数を生成するのでｕｎｆｏｌｄｉｎｇの速度も増す。

６．２　ベクトル計算の並列処理

データ駆動方式はデータを出力することによって次の命令を駆動するので、通常は出力トークンを出さない命令はＮＯＰ命令以外にはない。しかしＳＩＧＭＡ－１でデータ構造を使用する場合、即ちベクトル計算の場合には出力トークンを出さない命令が存在する。その命令はｓｔｏｒｅ命令で、この命令は値を構造メモリに書き込むだけで他の命令を駆動する必要がない。なぜならこのデータを必要とする場合は図１０に示すようにＦＥＴＣＨ 命令を使ってｒｅａｄパケットを出して必要な値を構造メモリから読めばよいからである。しかしこの場合は面倒な問題が起こる。ループ制御は繰り返し回数ｎ個のループ変数を生成すると、ループ終了のＲＥＴＵＲＮ命令を発火してしまう（図１１）。そうするとこのループのカラーが回

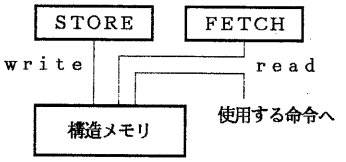


図10　構造メモリのreadとwrite

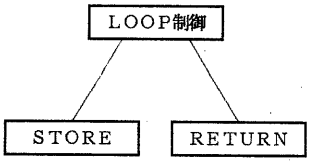
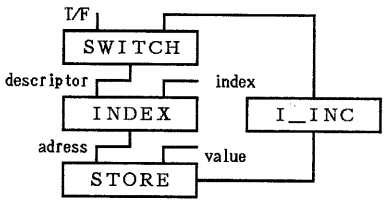
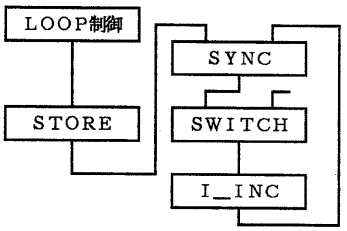


図11　STORE命令の問題

収されてしまうので、ＳＴＯＲＥ命令のアーク上にまだ実行されていないトークンが残ってしまうことがあり、エラーの原因となる。これを防ぐためにはＳＴＯＲＥ命令も出力トークンを出し、ループ制御と同期を取る必要がある。
その方法として図１２（ａ）のようにＳＴＯＲＥ命令自身にフィードバックをかけ、ＳＷＩＴＣＨ命令でループ終了を検出する方法が考えられる。しかしこの方法はＳＴＯＲＥ命令を逐次実行することになり、ｕｎｆｏｌｄｉｎｇ機構の良さが失われる。この場合は図１２（ｂ）のようにａｂｓｏｒｂルーチンを用いればＳＴＯＲＥ命令が早く実行されるのでこのストアされた値を利用する部分の処理が促進される。ｓｔｏｒｅ命令の数がｎ個の場合（ａ）のオーバヘッドは２ｎであるが（ｂ）の方法では２×［ｎ／２］＋３となりオーバヘッドが小さい。



(a) フィードバックによるSTORE命令の逐次実行



(b) ａｂｓｏｒｂルーチンによるSTORE命令の並列実行

図12　ベクトル計算の並列処理の方法

７．　結び

ＳＩＧＭＡ－１プロジェクトの目的は実用規模のプログラムを実用的な速度で実行するデータ駆動計算機を開発し、データ駆動計算機の実用性を証明することである。今回の予備試作版の評価の結果データ駆動計算機は同じテクノロジーを用いたフォンノイマン型の計算機に匹敵する性能を持っていることがわかった。

またアーキテクチャの改良特にループ処理の高速化が非常に重要であり、ｓｔｉｃｋｙトークンが効果的であることが確かめられた。さらに今後のループ処理を高速化するための指針を与えた。今後はより大規模なプログラムによる評価を予定しているが、そのためにはコンパイラの整備が重要である。

データ駆動計算機で良く問題となる資源の枯渇の問題は今回は生じなかった。今後ＳＩＧＭＡ－１がマルチプロセッサ構成となり、並列処理が行えるようになれば問題になると思われるのでその対策を考えたい。

ＳＩＧＭＡ－１プロジェクトでは現在４台のＰＥと４台のＳＥを局所ネットワークで接続したシステムを製作中である。このシステムはＣＭＯＳゲートアレイにより実現され、クロックは１１０ｎｓの予定である。今回の結果からみて１台のＰＥの性能は１．８ＭＩＰＳ程度と予想される。このシステムによりデータ駆動計算機の並列処理の実用的速度についてある程度の示唆が得られるであろう。更にその後は２００台規模のシステムに

より平均速度１００ＭＦＬＯＰＳの性能を達成することをを目標としている。

謝辞

本研究は大型プロジェクト「科学技術用高速計算システムの研究」の一環として行った。研究の遂行にあたり御指導、御討論いただいた柏木　寛電子計算機部長及び弓場敏嗣計算機方式研究室長および同僚諸氏に感謝いたします。

参考文献

(1) Dennis,J.B."First version of a data flow procedure language", Lecture Notes in Computer Science, Vol.19. Springer-Verlag, 1984.

(2) Syre,J.C.,et al."LAU sytem--A parallelda ta driven software/hardware system based on single assignment",In Parallel Computers--Parallel Mathematics, Elsvier North Holland, 1977.

(3) Arvind, Gosterow,K.P.,and Plouffe,W. "An asynchronous programming language and computing machine", Tech.Rep.TR114a,Dept. of Information and Computer Science, Univ.of C alifornia, Irvine, 1978.

(4) Gurd,J.R.,Watson,I.,and Glauert,J.R.W. "Data driven system for high speed parallel computer--part2: hardware design", Computer Design, July, 1980, pp.97-106.

(5) Amamiya,M,Hasegawa,R.,Nakamura,O.,and Mikami,H."A list processing oriented data flow machine architecture", Proc. of NCC, 1982, pp.143-151.

(6) Suzuki,T.,Kurihara,K.,Tanaka,H,. and Motooka,T."Procedure level data flow processing on dynamic structure multimicroprocessors", Joun. Inf. Process., IPS Japan, Vol.5, No.1, 1982, pp.11-16.

(7) Kishi,M.,Yasuhara,H., and Kawamura,Y." DDDP:a distributed data driven processor", Proc.10th Ann. Symp. Computer Architecture, 1983.

(8) Shimada,T.,Hiraki,K., and Nishida,K."An architecture of a data flow machine and its evaluation", Proc. of COMPCON, Spring,1984, pp.486-490.

(9) Yamaguchi,Y.,Toda,K.,Herath,J., and Yuba, Y."EM-3: A lisp-based data-driven machine", Proc. Int. Conf. Fifth Gen. Computer Systems, ICOT, 1984, pp.139-143.

(10) Hiraki,K.,Shimada,T., and Nishida,K."A hardware design of the SIGMA-1--a data flow computer for Scientific computations",P roc. Int. Conf. Parallel Processing, 1984,pp. 524-531.

(11) 島田、平木、西田「科学技術計算用データ駆動計算機ＳＩＧＭＡ－１のループインバリアントの処理法について」第２８回情報処理大会　1984.

(12) 唐木「世界を制した超高速国産コンピュータ」日経コンピュータ　1984.3.5.

(13) Gurd,J.R.,Kirkham,C.C., and Watson,I. "The Manchester prototype dataflow computer", Comm. ACM, Vol.28, No.1, 1985,