

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 processor of 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

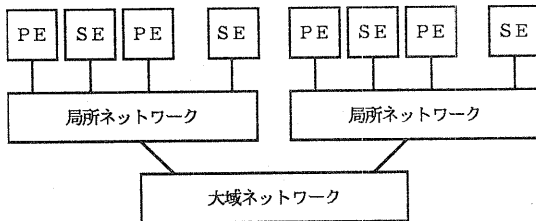


図1 SIGMA-1の全体構成

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ーフェースであり、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。

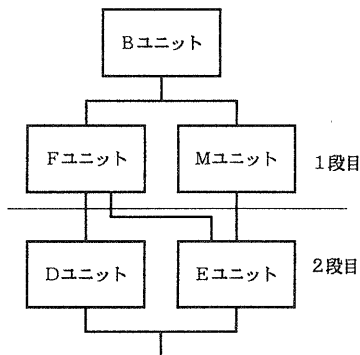


図2 PEの構成

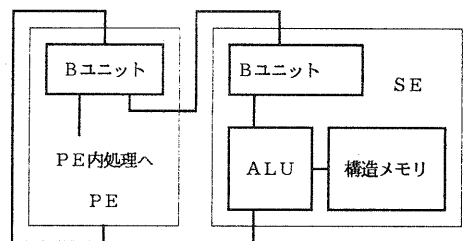


図3 SEの接続と構成

```

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)。

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ので、今回の測定で使っ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ういた。

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、

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

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ェアの性能

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 fadd1: pairs_r))
        (false (paf syn: normal_r)))
mull:  (*FMUL    (paf fadd2: normal_l))          ;y=x*x
fadd1:  (*FADD   (paf i_incl: s1))                ;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ー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のハードウェア性能をもとに、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クロックずつアイドルする。

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、

$T1 = \Sigma \text{各命令の並列実行時間} = 33 \text{クロック}$
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マシサイクルは3クロックなので、

$T2 = 3 \times 5 = 15 \text{クロック}$
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ーラップして送り出されるが、残りの5個は2クロックずつの遅れを作り出すのでその合計T3は、

$T3 = 2 \times 5 = 10 \text{クロック}$
である。

このループを1回実行するのに要する時間T4は、

$T4 = T1 + T2 + T3 = 58 \text{クロック}$
である。ループは100回実行したので総クロック数T5は

$T5 = 58 \times 100 = 5800 \text{クロック}$
である。

1クロックは155nsなので総実行時間T6は、

$$T6 = 155 \times 5800 \\ = 899000 \text{ ns} = 0.899 \text{ ms}$$

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

$T1 = 70 \text{クロック}$

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

$T2 = 3 \times 6 = 18 \text{クロック}$

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

$T3 = 2 \times 5 = 10 \text{クロック}$ 。

このループを1回実行するのに要するクロック数T4は

$T4 = 98 \text{クロック}$

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

$T5 = 98 \times 400 = 39200 \text{クロック}$ 。

1クロックは155nsなので総実行時間T6は

$T6 = 155 \times 39200 \\ = 6076000 \text{ microsec} = 6.076 \text{ ms}$

```
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 ループのオーバーヘッド

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、制御のオーバー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ー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、これはオーバー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値はハードウェアの性能は示すが、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ー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ェ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ェアでも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。現在スーパー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エントに特別な対策を行っている。

ループインバリエ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

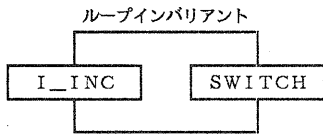


図7 (a) ループインバリエントの循環による保持

のでこれらの引数のうちループインバリエ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エ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エントについて2命令が必要であり、ループ内で多くのループインバリエ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エントはstickyトークンを用いて前述のオーバー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。入力アークの一方が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ーヘッドは生じない。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。ループインバリエントが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

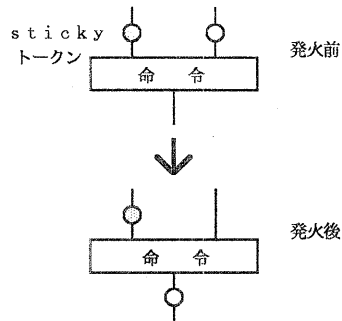


図7 (b) stickyトークン

なるがこれは1回実行するだけである。stickyトークンのオーバー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

$$\lceil (k-1) / 2 \rceil$$

個増える。この場合のSWITCH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

$$6k + 3k + 2 \lceil (k-1) / 2 \rceil + K = 10K + 2 \lceil (k-1) / 2 \rceil$$

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

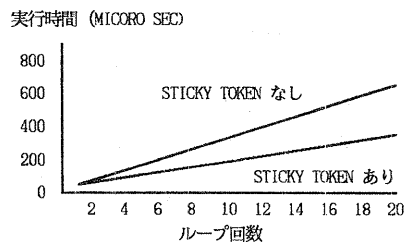


図8 STICKYトークンの効果

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 `stick` 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 `stick` 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。

`stick` 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 `stick` 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 $8n$ のオーダーとなる。

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 $16n$ のオーダーとなる。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ーヘッドが大きい。図 9 は階乗の計算の $n = 2 \sim 13$ 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。

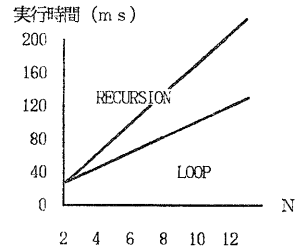


図9 ループとリカーションによる階乗計算の比較

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ーヘッドとなる。これを 1 命令で実現すればオーバー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

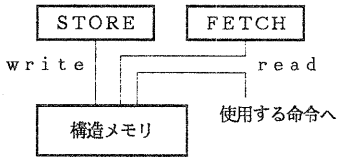


図10 構造メモリのreadとwrite

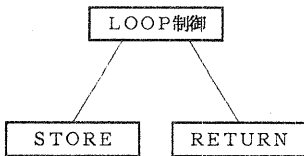
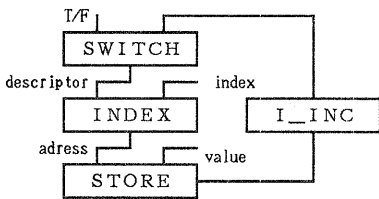


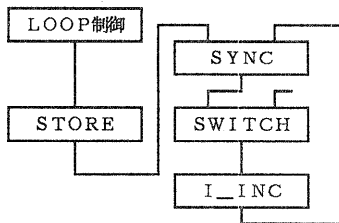
図11 STORE命令の問題

取されてしまうので、STORE命令のアーキテクチャ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



(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

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)のオーバーヘッドは2nであるが(b)の方法では $2 \times [n/2] + 3$ となりオーバーヘッドが小さい。

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 system--A parallel data driven software/hardware system based on single assignment", In Parallel Computers--Parallel Mathematics, Elsevier 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 California, 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", Proc. 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のループインバリエ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,