

5C-1

科学技術計算用データ駆動計算機 SIGMA-1のソフトウェア環境

島田俊夫 関口智嗣 平木 敬

電子技術総合研究所

1. はじめに

科学技術計算用データ駆動計算機SIGMA-1は市販のMSI、SSI版1台の予備試作を経て、現在LSI版4台のプロトタイプが稼働中である。本稿では、このプロトタイプ及び現在開発中である128台のシステム(図1)を使用する際のソフトウェア環境について、データ駆動計算機に関わる特徴を持つものを中心に述べる。SIGMA-1のソフトウェア環境は、大きく分けて、言語処理環境、制御環境、デバッグ環境の3つがある。以下に各々を概観する。

2. 言語処理環境

SIGMA-1のプログラムは高級言語DFCまたはマクロアセンブラのSASで書く。これらはホスト計算機vax11/750の上に実現されている。ロードモジュールはパケットにより、ネットワークを通してSIGMA-1にロードされる。

DFCはC言語とできる限り互換性を保つように設計された言語である。データ駆動計算機で実行されるため、副作用なし、単一代入則、実行制御はデータ駆動、という性質を持つ。DFCの1つの特徴はループのunfoldである。DFCではCとの互換性を追求したため、unfold用の予約語を導入せず、for文の(式1;式2;式3)の式3で、ループの世代を越える変数に単一代入則を破る形で代入を行ってunfoldを実現している。unfoldの利点はループ内で関数呼び出しがある場合や、2重ループの場合に、関数単位または内側ループ単位で自然に並列処理が行われる点である。

DFC言語のコンパイラのもう1つの特徴として、ごみトークンの処理がある。SIGMA-1では関数の実行に際してリンク番号(いわゆるカラー)を与えるが、リンク番号の数は1グループで256しかないため、関数の終了時点でリンク番号を回収する必要がある。そのためには終了した関数にトークンが残らない、つまりself-cleanなデータフローグラフを生成する必要がある。そのためには条件文の中の一部の命令や、アレイへの書き込み命令などからごみトークンを発生し、全ての命令が実行されたことを確認する必要がある。DFCコンパイラはこのごみトークン処理をサポートしている。

コンパイラの出力コードの最適化は、ほぼ達成されているので、今後はCISCアプローチに当たるマクロ命令の導入を更に進め、ベクトル処理の高速化を計る必要があろう。

マクロアセンブラSASはデータフローグラフを記号的に表現したものである。その形式は、ラベル (命令 [定数] 行き先命令リスト) である。ラベルは関数内でユニークでなければならない。行き先命令リストの形式は、

(割付関数 行き先ラベル マッチングフラグ)

である。この割付関数は、グループ内のPE4台の命令レベルでの負荷分散を決定する。グループ間の負荷分散はネットワークに組み込まれたハードウェアで動的に行っている(1)。グループ内の割付関数には、静的なもの

的なものがある。静的なものはデータフローグラフの形状とネットワークの遅れを考慮して最適負荷を計算し(2)、PE番号を直接、割付関数部分に書き込む。動的な場合には下の式で決定される。

PE番号 = 静的PE番号 + [ループカウンタ(下2ビット)]

+ [リンク番号(下2ビット)] + [自分のPE番号]

ここで[]はオプションを示す。静的PE番号に修飾をかける理由は、ループやリカーションで同じコードを繰り返し実行するときに、長い実行時間の命令を同じPEで実行し、そこがクリティカルパスを作ること避けるためである。これらの修飾を使うかどうかは、ユーザが指定することができる。

行き先ラベルはこの関数内での行き先命令のラベルである。マッチングフラグは行き先命令でのデータの入力タイプを示している。これには1入力型と、2入力型があり、2入力型には通常型、即値型、ループ不変型、特殊型がある。

SASでマクロになっている部分は、関数のcall、return等である。SASはループカウンタを使わないループや、アレイの同じ場所に1回以上データを書き込むシステムプログラムを作るとき有用である。

SASはアセンブラによりASSMと呼ぶ絶対番地のコードに変換され、次いでバイナリコードに変換されロードされる。

3. 制御環境

ローダ：ローダのプログラムはホスト計算機上に実現されている。ロードの手順は次の通りである。まず1グループを管理するPEにリンク番号を要求するパケットを送る。PEは空いているリンク番号をホストのローダに送り返す。ローダはこのリンク番号を使ってパケットの形式でプログラムをPEにロードする。グループ内のPEにはすべて同じプログラムをロードする。

SIGMA-1ハードウェアは動的ローディングをサポートする能力があるが、この方法は実行速度が遅くなるので、サイズが64KW以下のプログラムの場合には全てのPEに同じプログラムをロードし、それ以上のサイズについてはグループごとに異なったプログラムをロードすることとする。

入出力：並列計算機における入出力の問題点の第一は並列プロセッサと入出力プロセッサの速度差である。並列プロセッサ1台の速度はそれほど高速ではないが、100台以上のプロセッサが並列実行を行い、入出力要求を行えば、入出力プロセッサがそれらに同時に応答することはできない。このような状態になったときでも、並列プロセッサの実行を妨げないことが重要である。

SIGMA-1では、実際の入出力はサービスプロセッサが行うので、入出力の要求は最終的にはサービスプロセッサに送られる。サービスプロセッサ上の入出力ルーチンはこれらを検知し、ファイルや端末に対して実際の入出力を行う。しかし各PEが、直接サービスプロセッサとデータ

Software Environment for a Dataflow Computer SIGMA-1

Toshio Shimada, Satoshi Sekiguti, Kei Hiraki

Electrotechnical Laboratory

交換を行うと上述の問題が起きる。そこで入出力データは、一旦SEの構造メモリにストアされ、状況に応じてメンテナンスプロセッサがサービスプロセッサに送ることとする。SEはPEと同数存在するので、入出力データのストアがネックとなることはない。SEの構造メモリが一杯になったときはメンテナンスプロセッサのメモリにデータをストアする。

第二の問題は入出力データの到着順序が不定なことである。並列計算機では、複数のプロセスの処理が並列に進むため、出力要求も並列に起こる。更に命令レベルデータフローの場合、プログラムの文の単位でも実行順序を実行前に決めることは難しい。このためどこかで出力データの整列を行う必要がある。すなわちバラバラのデータをどこかで順序付けされた一連のデータ列であるストリームに変換することが必要である。SIGMA-1ではSEのB構造が関数的データ構造となっており、ストリームと同等の機能を持っているので、B構造に出力データを書きこめばデータの整列ができる。

データ駆動計算機では、更に第三の問題として、いつ入出力データが揃ったかを検出する問題が加わる。この場合の問題点は、いつストリームの要素が全て揃ったかわからないので、どの時点でメンテナンスプロセッサがストリームをサービスプロセッサに送ってよいか分からないことである。これに対する解は二つある。一つはプログラムの実行が終了した時点でSE内の出力データを順にメンテナンスプロセッサが読み出し、サービスプロセッサに送る。この方法は大部分の科学技術計算に通用するが、グラフィックスを用いた対話型の計算には対応できない。もう一つの方法は実行途中でもデータをサービスプロセッサに送る方法である。print命令がPEで実行されると、メンテナンスプロセッサに対して、指定したB構造を読み出す命令を送る。メンテナンスプロセッサはそのB構造の内容を最初から順に読み出し、自分のメモリ内に識別子をつけて順にストアして行く。もし全てのデータが書かれていれば、その一連のデータをまとめてバスを通してサービスプロセッサへ送る。もしある要素にデータが書かれていなかったら、その要素の待ちキューに出力トークンを置いておく。データが書き込まれたら、その要素からデータの読み出しを再開しメンテナンスプロセッサへ送り、最後まで到達すればまとめてサービスプロセッサへ送る。もし最後までデータが書かれなかったら、プログラムの実行終了時点でサービスプロセッサが各メンテナンスプロセッサに対し待ちキュー内の出力トークンを起こし、書いてない要素は飛ばしてB構造を読み出し、まとめてサービスプロセッサへ送るよう指示する。

端末から対話形式で入出力を行う場合や、既に用意されているファイルから順に読み出す場合などは、ユーザがこれらの事象が逐次に起こるようなプログラムを書く必要がある。これらはループで行うか、その前の入出力データを使って次の入出力を駆動する方法による。

4. デバッグ環境

並列計算機におけるデバッグは逐次計算機よりも難しい。その理由は実行が並列であるため、論理を追いかける人間も頭の中で並列実行をシミュレートしなければならないことと、同期部分が正しいかどうかのチェックが困難であることのためである。データ駆動計算機では、更に実行の順序がプログラムに記述された通りでないため、論理を追いかけるのに苦労する。

SIGMA-1では関数型言語でプログラムを書くので、同期の問題はシステムが処理し、この部分でエラーが発生することはない。そのためSIGMA-1では、複数台で実行した場合と1台で実行した場合の結果が同じになるという利点がある。

したがってSIGMA-1では、プログラムのデバッグは1台の計算機で行うことができる。

デバッグの道具としてはトレーサとステップがある。トレーサは、実行中に発生したパケットを全てホスト計算機に送る。ステップは、1ステップごとの実行を可能にする。この方法の欠点は、大量の情報を発生するため人間がフォローしきれない場合があることである。条件付きで動作するトレーサとステップが必要である。

プログラムが大きい場合には通常の実行を行い、途中で止まった結果をダンプして状況を調べるのも有力な方法である。この場合、プロセスを管理しているリンクレジスタと、実行途中で相手のトークンを待っている待ち合わせ記憶のダンプが有用である。デバッグの手順は以下の通りである。まず待ち合わせ記憶内から一番新しいトークンを見つけ出す。このトークンの行き先命令に相手のトークンが到着しないから実行が止まったわけで、相手のトークンがなぜ到着しないのかをアセンブラを逆に辿り、待ち合わせ記憶の中のトークンとつき合わせながら調べる。データ駆動方式の場合1台で実行しても、実行順序が複数のプロセスに渡るので、辿る過程でリンクレジスタのプロセス番号に応じてトークンをソートしてダンプするルーチンが有用である。このとき使用するアセンブラコードとしてバイナリコードを記号的に表現したものが用意されている。

5. おわりに

データ駆動計算機SIGMA-1のプログラム開発を行う時のソフトウェア環境について、データフローに特長な点を中心に述べた。今後は上記のソフトウェア環境で、応用プログラムを開発し、SIGMA-1の評価を行う予定である。その過程でコンパイラの最適化を進めるとともに、より使い易い環境を整える。より高度なトレーサやダンプ、数値計算用の基本的ライブラリ、評価用情報を得るための監視プログラムなどの開発が必要であろう。

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」の一部として行った。研究の遂行にあたり、御指導、御討論頂いた柏木電子計算機部長、弓場計算機方式研究室長及び同僚諸氏に感謝致します。

参考文献

- (1) 平木、関口、島田「科学技術用データ駆動計算機SIGMA-1LSI版のネットワーク構成」情報処理学会第33回全国大会5c-3。
- (2) 大塚、坂井、弓場「データ駆動計算機における静的負荷分散方式の検討」情報処理学会第33回全国大会5c-6。

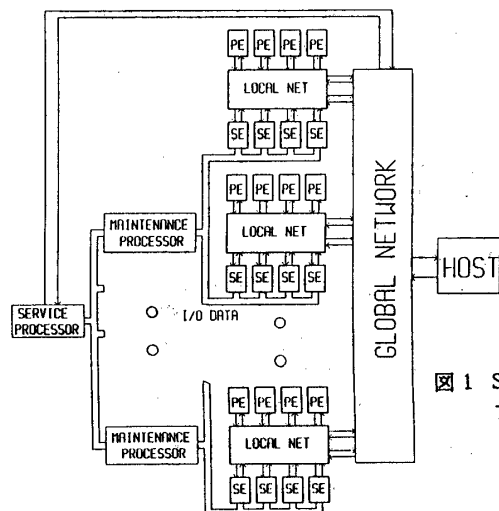


図1 SIGMA-1の
ブロック図