1N-8 ＳＩＧＭＡ－１用言語ＤＦＣに対する拡張機能の提案

関口智嗣，島田俊夫，平木敬
（電子技術総合研究所）

1．はじめに

電子技術総合研究所において，次世代のスーパコンピュータを目指した科学技術計算用データ駆動計算機ＳＩＧＭＡ－１の開発を進めてきている．このＳＩＧＭＡ－１は数多くの実用規模アプリケーション問題に適用して，性能を評価することにより，データ駆動計算機の実用性を検証することが目標である．実用規模アプリケーションを記述するためのユーザ用高級言語としてＤＦＣ(Data Flow C)の開発を行った[1]．このＤＦＣはＣ言語との互換性を維持しつつ，単一代入則の導入により副作用の排除を行ったものである．ここでは，具体的な実用プログラムを例にして，ＤＦＣで記述した場合の問題点を指摘し，その対策をＤＦＣに対する拡張機能として提案する．

2．記述対象のアプリケーション

具体的にＤＦＣで記述対象としたアプリケーションとその特徴を示す．

・ＱＣＤ：格子ゲージ理論による色量子力学の数値シミュレーション．これは４次元空間の扱い，乱数の生成と消費によるモンテカルロ計算を行い，最終的には大規模疎行列の解法が必要となる．

・ＰＩＣ：プラズマ粒子モデル等で現れる粒子と場を交互に扱い，統計処理やＦＦＴが必要となる．さらに，粒子を負荷とみた場合の分散手法が必要となる．

・ＫＥＮＯ：ボルツマン輸送方程式に基づく中性子輸送問題のモンテカルロ計算による解法の具体的コード名．やはり乱数の生成と消費が必要であり，物質・物理定数の参照や粒子の履歴を保存するための統計処理が必要である．

・LINPACK/IMSL：科学技術計算用の数学ライブラリの例である．線形計算，ＦＦＴ，乱数発生，固有値問題の解法が基本アルゴリズムとして必要である．

これらの記述をＤＦＣで行おうとした場合に以下に順に示す並列記述要素が必要となる．

3．プロセス情報の管理

ＳＩＧＭＡ－１では４台の演算処理装置（ＰＥ）と４台の構造体処理装置（ＳＥ）がローカルネットワークで結合してひとつのグループを形成している．このグループが単位となり，プロセス（＝関数，サブルーチン）の処理を行う．全体システムは３２グループで構成され，各グループ間は２段のオメガネットワークで結合されている．これにより，任意のグループ間は通信距離という意味で一様な関係になる．

ＳＩＧＭＡ－１のネットワークは自動負荷分散の機能を有しており，新たに発生したプロセスは負荷最小のグループで実行されるように，動的負荷分散がおこなわれる．これは負荷が実行時に動的に定まるような問題に対して非常に有効である．また，アルゴリズムの性質から実行前に負荷分散が可能な場合には，プロセスの割り当てるグループを予め与えた静的負荷分散も有効である．

さて，動的負荷分散，静的負荷分散，いずれの場合でも割り当てられたプロセス間の通信が機能として必要となる．例えば，ＦＦＴのような場合を考えると，『回転因子を乗じて，加算を行う』といった基本演算単位をプロセスとすると，この演算結果を他のプロセスと交換する必要がある．現在のＤＦＣでは，ＦＦＴのlogN段の各ステージ毎に全グループに対してプロセスを割り付けるという操作(CALL)が必要である．この操作は正味の演算量と比較して重く，オーバヘッドとなる．

ＤＦＣではプロセスの番号，すなわち割り付けられたグループの番号とそこでのカラーをデータとして記述することができない．この記述により，３次元，４次元格子，対数的結合，不規則など何等かの構造を持ったプロセスの関係と，割り付けられた論理的グループ空間との関係が写像できる．

これに伴って，実際にプロセス間の通信を記述するプリミティブが必要となる．すなわち，ＤＦＣで記述できるプロセス間の通信は親子関係に基づいて，引き数と結果の値による場合だけであるので，プロセス間通信を行うポートをプロセス中に定義し，記述する．このプロセス間通信は非同期メッセージ送信，同期メッセージ送信／通信で実現する．

4．配列への再書き込みの記述

ＱＣＤ計算でのリンクの更新や，連立一次方程式のRed-Black法によるSOR法など，配列データの全要素に対して定数回ｎの読みだしが行われ，これが繰り返される場合がある．ＤＦＣでは不定回の読みだしに対しての実行を保証し，さらに，ループを展開して実行を進める意味であるため，恒に古い配列のコピーを作成しこれを保存する．このコピーの回収は当該プロセスの終了と同時に行われる．自由な配列のアクセスに対してＤＦＣの手法はそれを保証しているが，ｎが定数として与えられているときにはコピーの作成，回収などのオーバヘッドが大きく，回収のタイミングなど微妙な問題を抱える．このような場合に対してのＤＦＣとしての記述法を別途用意し，配列に対しての異なった処理を行う必要がある．

A Proposal for some functional extension of DFC
Satoshi SEKIGUCHI,Toshio SHIMADA and Kei HIRAKI
Electrotechnical Laboratory

実際に，red-black SORの例では，

```
while(収束していないならば){
   赤の格子点の更新 /*古い黒の値を用いる*/
      [同期]
   黒の格子点の更新 /*更新した赤の値を用いる*/
      [同期]
}
```

といった同期をコードとして生成すればよい．

5．統計処理の記述

ＰＩＣで必要な，荷電粒子による場に刻まれた格子点への効果を記述する場合を考える．格子点の立場に立てば自分に効果を及ぼす荷電粒子が何個存在するか不明である．また，荷電粒子の動きにより，個数は不定となる．４で述べた場合とは格子点により読みだし，書き込みの回数が異なることに問題点がある．さらに並列処理を行っていることにより，ひとつのデータに対して読みだし／書き込みのタイミングが重要で，レーシングに対しての留意が必要となる．

実際の演算としては，

```
for(i=0;i(n;i++){ /*全粒子に対して*/
 n_grid = int(position[i]);/*最寄りの格子点*/
 value[n_grid]=value[n_grid]+1.0;/*効果を加える*/
}
```

といった形式で，この形式ではＳＩＧＭＡ－１の場合にvalueをＳＥへ割り当てる．しかし，ＳＥに割り付けた場合，データの読みだしと更新が不可分には行われないため前述の問題点が生じる．これをＳＩＧＭＡ－１で実現するためにＳＥのアドレスに対応してループ番号(I)とカラー(LN)を用いて，マッチングメモリでのアドレシングを行う．命令アドレスは同一にし，そのアドレスにこの場合はFADDをおき，出力を自分自身に返して累積算演算を行う．

このような例はＫＥＮＯ等で現れるヒストグラム作成のためのカウント処理をはじめとする統計処理などでも現れる．

6．各種型の記述

ＤＦＣはＣに較べて型の種類が少なく，Ｃで慣れたユーザが最初に戸惑う点でもあった．純粋なデータフローの枠組みの中ではポインタという概念が存在しないため，ＤＦＣでも対応をとっていなかった．しかし，Ｃとの互換性を追究するにはポインタ型は必須である．しかも実際，ＳＩＧＭＡ－１ではＳＥにおけるアドレスとしてポインタ型を実現できる．すなわち，ポインタ操作を行う変数はＳＥに割付，ポインタ操作を伴わない変数は純粋なデータフロー変数として取り扱う．この区別は，Ｃにおけるregisterとしての記憶クラスに割り付けられた変数はポインタ操作を行えないことに対応している．さらにポインタが利用できることによりchar型も自然に導入できる．

また，ＤＦＣでは大域的変数の存在はなかった．大域変数は並列実行の場合，実行の順序によって値が不定になる．逆に，この不定さがプロセス間の実行状況を記述するに不可欠である．実際に，ＳＥを使って実現ができる．大域変数をＳＥのある領域にアドレス付きで格納すれば十分である．この大域変数に対するアクセスは非同期書き込み命令ASTORE，非同期読みだし命令AFETCHを用いる．

この他，配列の全要素に対して同一の演算を施すようなベクトル命令に対しては配列の記憶クラスとしてhls（高機能構造体）を与える．通常の四則演算，内積演算，マスク演算などが

```
hls float a[],b[],c[],m[];/*高機能構造体宣言*/
      c = (a + b) & m; /*高機能構造体命令*/
```

として記述できれば，ベクトル演算の抽出が容易になる．

7．おわりに

ＳＩＧＭＡ－１用高級言語ＤＦＣは昭和５９年度より開発を進め，現在ほぼ安定状態にある．しかし，ＤＦＣは一般のユーザが容易にＳＩＧＭＡ－１を利用するための言語という設計の立場であった．また，処理系の開発がアーキテクチャの改良による命令セットの改良と同時並行的に行われた．この状況により，ＤＦＣではユーザに対して言語仕様上の制限を与えることにより，アーキテクチャの変更とは独立に処理系が作成できた．また，ＤＦＣでは言語仕様上で記述可能であった内容はシステムがその結果を一意に保証している．しかし，この言語仕様上の強い制約により，並列処理実験システムとしてのＳＩＧＭＡ－１をＤＦＣでは十分に記述できない．すなわち，カラー，メモリなどの計算資源の割付と解放，関数間のデータ通信など細かな並列処理の制御，データの競合，並列処理効率，スケール則，速度等の測定が自由でなかった．今回の考察により，ＤＦＣにおける言語仕様上の制約を取り去り，ハードウェアで実行可能なことは全て言語仕様で記述できることが必要になってきた．この自由を得たことにより，ユーザはデータの競合や多重書き込みなどに対する責任を自らが負うことになる．現在はここにあげた項目を基に新たな言語設計を行っている．

本研究は通産省大型プロジェクト「科学技術用高速計算システムの研究」の一環である．日頃より御指導頂く田村浩一郎電子技術総合研究所電子計算機部長，弓場敏嗣計算機方式研究室長および同僚諸氏に感謝する．

【参考文献】

[1] 島田 他:データフロー言語ＤＦＣの設計と実現 電子情報通信学会論文誌，vol. J71-D, No. 3, pp. 501-508, (1987).