解　説

階層構造の MIMD 型スーパコンピュータ†

——イリノイ大学の Cedar プロジェクトを例として——

小　原　和　博††

1. はじめに

一般的に言って，科学技術用超高速計算機のことをスーパコンピュータと呼んでいる．スーパコンピュータでは $A=B+C$ というようなスカラ演算だけでなく，$A(i)=B(i)+C(i)$ というようなベクトル演算の高速化も重要となる．したがってスーパコンピュータでは，汎用コンピュータのアーキテクチャ（一般的にSISD*[1]）とは異なり，並列処理を可能とするアーキテクチャ(SIMD*[2], MISD*[3], MIMD*[4])が採用されている[1)~4)]．

SIMD の例としてプロセッサアレイ方式がある．プロセッサアレイ方式の代表としてはイリノイ大学がバロース社と '73 年に共同開発した ILLIAC-Ⅳ（最高性能 80 MFLOPS*[5]）が挙げられる[5)]．また，MISD の例としてパイプライン方式がある．パイプライン方式の代表としてはクレイ・リサーチ社が '76 年に開発し，スーパコンピュータのベストセラーとなった CRAY-1（最高性能 160 MFLOPS）がある[6)]．

現在の代表的な商用スーパコンピュータを，開発中のスーパコンピュータとともに**表-1** に示す[7)]．従来の商用スーパコンピュータでは，ほとんどでパイプライン方式が採用されている．パイプライン方式では，パイプラインの本数を増やすことと，サイクルタイムを短縮することによって性能向上が図れる．例えば日本電気の SX-2 では，加減算用 4 本，乗除算用 4 本，論理演算用 4 本，シフト演算用 4 本，合計 16 本のパイプラインがあり，そのサイクルタイムは 6 ns である[7)]．VLSI 技術等の進歩により，パイプライン方式のままで最高性能が 10 GFLOPS*[6] 級のスーパコンピュータを実現することは可能である．しかしスーパコンピュータのユーザにとって問題となるのは，瞬間的な最高性能ではなくて定常的な実効性能の方である．実効性能を高めるためには，ハードウェア面での工夫（演算器の追加，サイクルタイムの短縮など）だけでなく，ソフトウェア技術を含めたシステム全体の工夫(例えば独立動作可能な演算器を多数台用意して，それらを効率よく使用する技術）が重要となる．そこで登場してきたのが，MIMD アーキテクチャのマルチプロセッサ方式である．デネルコ社が '81 年に発表した HEP (Heterogeneous Element Processor) は，商用機として初めてマルチプロセッサ方式（最高 16 台）を採用したスーパコンピュータである[8)]．またクレイ・リサーチ社が '82 年に発表した CRAY X-MP も小規模（2 台）ながらマルチプロセッサ方式となっている[9)]．研究開発中のものではコントロール・データ社の CYBER 2XX [25)]*，ローレンス・リバモア研究所とスタンフォード大学の S-1[11)]，イリノイ大学の Cedar[12)~17)] などがマルチプロセッサ方式のスーパコンピュータである[19),20)]．

筆者は 1982 年 9 月から一年間イリノイ大学の **Cedar**（シーダー）プロジェクトに参加する機会を得た．本プロジェクトの最終目標は 1990 年前後に 1024 台以上のマルチプロセッサ方式により，10 GFLOPS 級のスーパコンピュータを実現することである．この

表-1　現在および将来のスーパコンピュータ

国名	開発者	機種名	最高性能 (FLOPS)	アーキテクチャ	開発年
米国	CDC	CYBER 205	400M	パイプライン 8 本	'80
米国	CRI	CRAY X-MP	630M	2 台マルチプロセッサ	'82
日本	富士通	VP-200	500M	パイプライン 6 本	'82
日本	日立	S-810/20	630M	パイプライン 12 本	'82
日本	日本電気	SX-2	1.3G	パイプライン 16 本	'83
米国	CDC	CYBER 2XX	10G	8 台マルチプロセッサ	('87)
米国	イリノイ大	**Cedar**	10G	1024 台マルチプロセッサ	('90)
日本	通産省	科学技術用高速計算システム	10G	大規模並列処理アーキテクチャ	('90)

† A Hierachically Structured MIMD-type Supercomputer —Referring to the Cedar Project at the University of Illinois as an Example— by Kazuhiro KOHARA (Yokosuka Electrical Communication Laboratory, N. T. T.).

†† 日本電信電話公社横須賀電気通信研究所

* 現在 CYBER 2XX は，CDC の子会社である ETA 社に移管され，名称も "GF-10" と変更された．

性能目標は，通産省のスーパコンピュータ計画「科学技術用高速計算システム」[21] とほとんど同じものである．本稿ではイリノイ大学の全面的支持のもとに Cedar プロジェクトで採用されている技術を引用しながら，マルチプロセッサ方式のスーパコンピュータを実現する上で重要となるいくつかの基幹技術について解説する．

*1) SISD: Single-Instruction stream and Single-Data stream
*2) SIMD: Single-Instruction stream and Multiple-Data stream
*3) MISD: Multiple-Instruction stream and Single-Data stream
*4) MIMD: Multiple-Instruction stream and Multiple-Data stream
*5) MFLOPS: Mega FLoating-point Operations Per Second
*6) GFLOPS: Giga FLoating-point Operations Per Second

2. Cedar プロジェクトの概要

2.1 Cedar のアーキテクチャ

Cedar のハードウェア構成を**図-1** に示す．Cedar は**バックエンド型**スーパコンピュータであり，ホストコンピュータ側でコンパイルされたプログラムを高速に実行する．Cedar の基本ユニットは，グローバルメモリ (GM)，グローバルネットワーク (GN)，グローバルコントロールユニット (GCU)，プロセッサクラスタ (PC)，I/O クラスタ (IOC) である．

アーキテクチャの特長は次のとおりである．(1)**マルチプロセッサ構成**であり，複数台のプロセッサが同一の命令を実行したり個別の命令を実行したりすることができる．(2)クラスタレベルとプロセッサレベルの**2階層構造**になっている．つまり Cedar の演算部は複数台のプロセッサクラスタからなり，各クラスタは複数台のプロセッサから構成される．(3)クラスタレベルの演算制御に**マクロデータフロー制御**を採用している．(4)クラスタレベルの相互結合ネットワークに**マルチパス・オメガネットワーク**を採用している．(5)**拡張可能なアーキテクチャ**となっており，プロセッサ台数やメモリ容量を増加することによって性能を向上できる．(6)GM, GN など各基本ユニット間の通信は**非同期制御**であり，部品技術の変更などに柔軟に対処できる．

2.2 Cedar の開発計画

Cedar プロジェクトの研究開発は**表-2** に示すとおり，試作機 (prototype system) と実用機 (production system) の 2 段階で行われる．当面，試作機の設計を

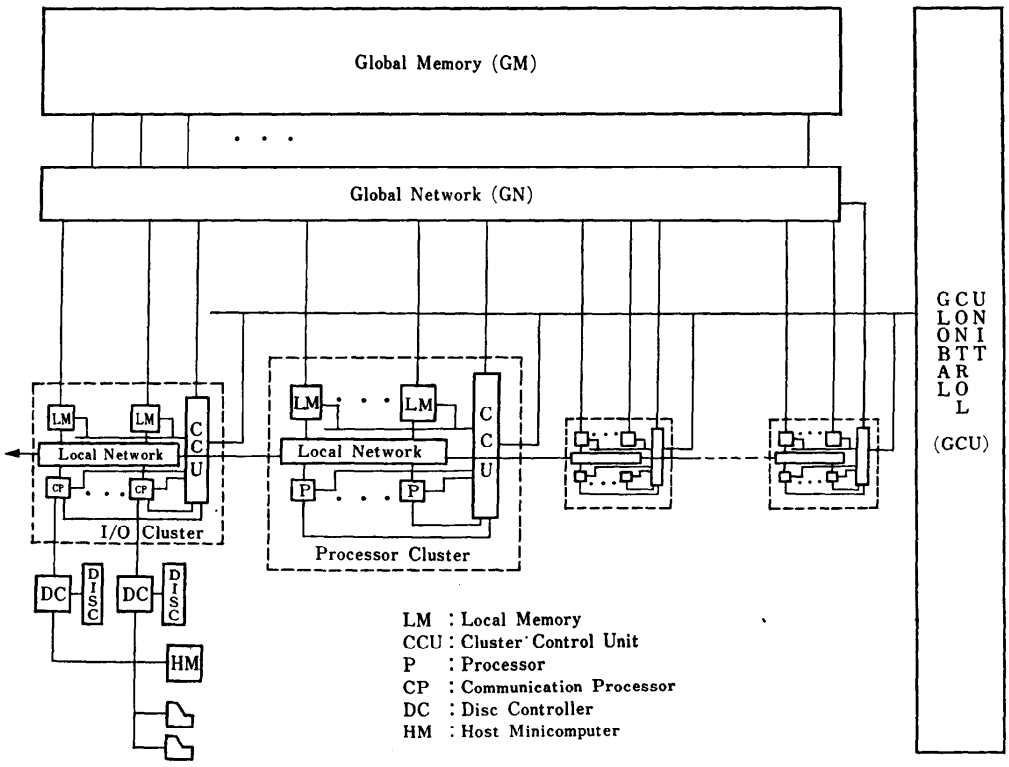


図-1 Cedar のハードウェア構成[4]

表-2 Cedar の開発計画[13]

	試作機	実用機（最上位機種）
開発目標年	1986	1990 前後
開発者	イリノイ大学	企業（未定）
実効性能	CRAY-1 相当	数 GFLOPS
アーキテクチャ	32 台 MIMD	1024-4096 台 MIMD
VLSI 技術	市販 LSI	フルカスタム VLSI

イリノイ大学単独で行い，'86 年までに 32 台のマルチプロセッサで CRAY-1 相当の実効性能を持つコンピュータ (**Cedar-32**) を実現する予定である．Cedar-32 の評価（動作確認，規模拡張性など）をみて実用機の開発が企業参加のもとに行われる予定である．

実用機ではシステム規模の拡張（プロセッサ台数やメモリ容量の増加など）と使用素子の高速化（より高速の論理 LSI とメモリ LSI を使用する）によって，最高性能ならびに実効性能の飛躍的増大を図る．システムの規模拡張により **Cedar-128** (128 台マルチプロセッサ)，**Cedar-512**, **Cedar-1024** などの上位機種が，また素子高速化により **Cedar-32 H**（ゲート遅延 350 ps 程度），**Cedar-32 VH**（同 80 ps 程度）などの上位機種が作られる予定である[16]．

Cedar のアーキテクチャと開発計画に関するさらに詳しい内容は文献 17) を参照されたい．

3. VLSI 技 術

10 GFLOPS 級のスーパコンピュータを実現するためには，バイポーラ，ガリウム ひ素などの超高速 IC 技術と，MOS を中心とする VLSI 技術が不可欠である．論理 LSI の技術動向をゲートアレイを例にとって**図-2** に示す．'83 年で実用段階にあるゲートアレイの中で最大規模のものは 8 キロゲート（ゲート遅延 2.5 ns) 程度であり[22]，最高速のものは 250 ps/ゲート（1 キロゲート）程度である[7]．また '83 年までに発表されたゲートアレイの中で最大規模のものは 20 キロゲート CMOS ゲートアレイ[23]であり，最高速のものは 150 ps/ゲートの 500 ゲート GaAs ゲートアレイ[24]である．そして日米のスーパコンピュータ計画で開発目標としているゲートアレイの性能は，通産省のプロジェクトでは 3 キロゲート以上かつ 30 ps 以下であり[21]，コントロール・データ社の CYBER 2XX では 20〜40 キロゲート (CMOS) である[25]．

Cedar プロジェクトにとっても VLSI 技術は欠くことのできない基幹技術である．しかし必ずしも最先端の VLSI 技術をあてにしているわけではなく，そ

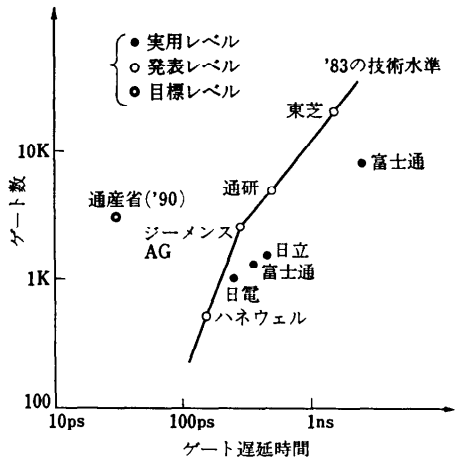


図-2 ゲートアレイの技術動向

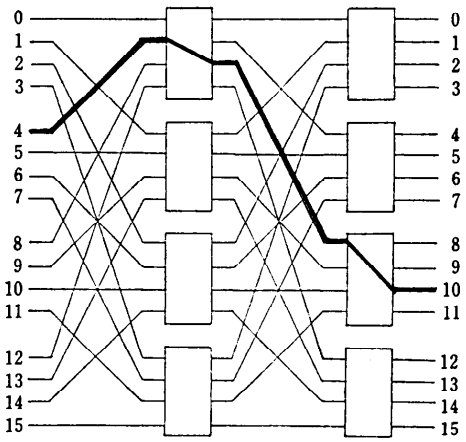
の時点で十分に使いこなされた技術を採用する予定である．ちなみに，Cedar-32 (開発目標年 1986)，Cedar-32 H (同 1989)，Cedar-32 VH (同 1991) で使用される予定のゲートアレイの性能は，それぞれ 800 ゲート (2.5 ns)，2000 ゲート (350 ps)，400 ゲート (80 ps) である[16]．また Cedar-32 では市販の浮動小数点演算用チップとして H-P 社の 2.5 MFLOPS チップセット[26]をプロセッサ内に搭載する予定である．

4. 相互結合ネットワーク

並列処理システムに適した相互結合ネットワーク (interconnection network) として，これまで (1) クロスバ・スイッチ，(2) 多段スイッチ・ネットワーク，(3) プロセッサ・ネットワークなど種々の方式が提案されている[27)-29)]．ここでは数千台のマルチプロセッサに耐える多段スイッチ・ネットワークとして提案された**マルチパス・オメガネットワーク** (multipath omega network)[16] について解説する．

4.1 1パス・オメガネットワーク

従来のオメガネットワーク（1 パス・オメガネットワーク)[30]について，16×16(入力数 16，出力数 16) のものを例にとって説明する．**図-3**(a) に構成を示す．この場合の構成単位は，2 ビット左回転のシャッフル交換 (shuffle exchange) と 4×4 のスイッチである．入力アドレスを (S3, S2, S1, S0)，出力アドレスを (D3, D2, D1, D0) としたときの機能を**図-3**(b) に示す．従来のオメガネットワークでは各入力アドレ



（a） 構成

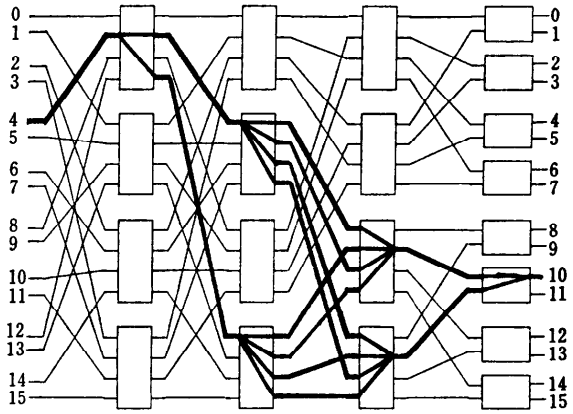
（S3, S2, S1, S0)
→(S1, S0, S3, S2)……2ビット左回転
=→(S1, S0, D3, D2)……S3, S2をD3, D2に置換
→(D3 D2, S1, S0)……2ビット左回転
=→(D3, D2, D1, D0)……S1, S0をD1, D0に置換
（b） 機能（→：シャッフル通過，=→：スイッチ通過）

図-3 16×16 1パス・オメガネットワークの構成と機能[15]

スから各出力アドレスに対してただ1通りのパスが与えられる．例えば4番入力から 10 番出力へのパスは図-3（a）に太線で示したようになる．しかしオメガネットワークはクロスバ・スイッチと異なり多段構成となっているために，他のパスと中間アドレスが一致する場合にはパスは確立されない．例えば4番入力と 10 番出力が通信中に，12 番入力から8番出力へパスを設けようとすると第1段めのスイッチの出口でぶつかってしまう．これは両者の中間アドレス（S1, S0, D3, D2）が共に（0010）となるためである．そこでオメガネットワークに冗長パスを追加し各入力アドレスから各出力アドレスに対して複数のパスを与えることによって，性能と信頼性を高めたのがマルチパス・オメガネットワークである．

4.2 マルチパス・オメガネットワーク

16×16 の8パス・オメガネットワークを例にとって説明する．**図-4**（a）に構成を示す．図-3 に示した1パス・オメガネットワークと同様にシャッフル交換とスイッチから構成されるが，2段構成から4段構成に増設されたのが大きな違いである．またスイッチには必要に応じて代替パスを選定する機能などが追加されている．4×4 のスイッチが3段で，2×2 のスイッチが1段となっている．つまり 16 の出力に対して合計 128 通り（4×4×4×2）のパスが与えられるので，



（a） 構成

（S3, S2 S1S0)
→(S1, S0, S3, S2)……2ビット左回転
=→(S1, S0, X1, D3)……S3, S2をX1, D3に置換
→(X1, D3, S1, S0)……2ビット左回転
=→(X1, D3, X2, X3)……S1, S0をX2, X3に置換
→(D3, X2, X3, X1)……1ビット左回転
=→(D3, X2, D2, D1)……X3, X1をD2, D1に置換
→(D3, D2, D1 X2)……下位3ビットを1ビット左回転
=→(D3, D2, D1, D0)……X2をD0に置換
（b） 機能（→：シャッフル通過，=→：スイッチ通過）

図-4 16×16 8パス・オメガネットワークの構成と機能[14]

各出力に対しては8通りのパスがある．入力アドレスを（S3, S2, S1, S0），出力アドレスを（D3, D2, D1, D0）としたときの機能を**図-4**（b）に示す．但し X1, X2, X3 は複数のパスを与えるのに必要な**冗長ビット**であり，0でも1でもよい．第1番めのスイッチでひとつの冗長ビット X1 が与えられるのでパスは2重化される．第2番めのスイッチでさらにふたつの冗長ビット X2, X3 が与えられるのでこの時点でパスは8重化される．第3番めのスイッチではふたつの冗長ビット X3, X1 が除かれるのでパスは2重に絞られる．第4番めのスイッチではさらにひとつの冗長ビットが除かれ目的の出力アドレスとなる．例えば4番入力から 10 番出力へのパスは太線で示したように8通りある．また4番入力と 10 番出力が通信中でも，12 番入力と8番出力の間にパスを設けることができる．これは中間アドレス（S1, S0, X1, D3）に冗長ビットが入っているために（0001）と（0011）の2通りのアドレスをとれるからである（ただし，さらに8番入力から 15 番出力にパスを設けようとするとぶつかりが生じてしまう）．

5. コンパイラ

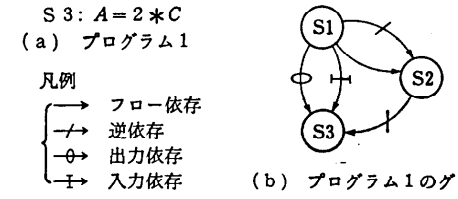
ベクトルプロセッサと呼ばれるスーパコンピュータでは，複数台の演算器が同時に同一の命令（ベクトル命令という）を実行することによって高速化を図る．これに対しマルチプロセッサ方式のスーパコンピュータでは，複数台の演算器が個別の命令を実行できるのでさらに実効性能を上げることが可能である．ここではベクトルプロセッサだけでなくマルチプロセッサにも適用できるコンパイラとして開発された **Parafrase**（パラフレイズ）を例にとって，並列処理のためのコンパイラ技術について解説する．ベクトルプロセッサで採用されているコンパイラ技術（自動ベクトル化技術）については文献 34）を参照されたい．

Parafrase はイリノイ大学が開発したスーパコンピュータ用コンパイラであり，その機能は概略次のとおりである[31)-33)]．（1）まず元のプログラムを解析して**依存関係グラフ** (dependence graph) を作成する．（2）次に冗長な依存関係 (dependence) を除去するなどして並列性の高いグラフに変換する．これを**グラフ変換** (graph transformation) という．具体例を用いて依存関係グラフとグラフ変換について説明する．

5.1 依存関係グラフ

最初に**図-5**（a）に示すプログラム 1 を例にとって，4 つのデータ依存関係 (data dependence) と依存関係グラフについて説明する．プログラム 1 で S1 の出力 A が S2 の入力となっているため，S2 は S1 に対して**フロー依存** (flow dependent) であるという（S1⟶S2 と印す）．この場合，S2 は S1 の後に実行しなければならない．S2 の出力 B が S1 の入力となっているため，S2 は S1 に対して**逆依存** (anti-dependent) であるという（S1⇢S2）．この場合，S1 の入力 B を変更しないように S2 は S1 の後に実行しなければならない．S3 の出力 A が S1 の出力 A と同じであるため，S3 は S1 に対して**出力依存** (output dependent) であるという（S1⇢S3）．この場合，S1 の出力 A が S2 の入力となっているので S3 は S2 の後に実行しなければならない．S3 の入力 C が S1 の入力 C と同じであるため，S3 は S1 に対して**入力依存** (input dependent) であるという（S1⟶S3）．入力依存だけの場合，S3 は S1 と同時に実行できる．プログラム 1 に対応する依存関係グラフを**図-5**（b）に示す．グラフからフロー依存，逆依存，出力依存を削減することによって並列性を高めることができる．

S1：$A=B+C$
S2：$B=A*3$
S3：$A=2*C$
（a）プログラム 1



（b）プログラム 1 のグラフ

図-5 依存関係グラフ[33)]

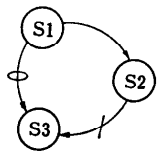
5.2 グラフ変換

グラフ変換には**アーク変換** (arc transformation) と**抽象変換** (abstract transformation) がある．アーク変換はグラフから依存関係をできるかぎり減らす方法であり，変数名変更 (renaming)，変数拡張 (expansion)，ノード分割 (node splitting)，先行代入 (forward substitution) などの手法がある．抽象変換はグラフ内のループを分散 (loop distribution) したり，統合 (loop fusion) したりすることによってプログラム処理の高速化を図る方法である．以上の中からアーク変換の例として変数名変更と変数拡張について，また抽象変換の例としてループ分散について具体的に説明する．

（a）　変数名変更

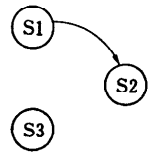
変数名変更によって逆依存と出力依存を除去できることを示す．変換前のプログラム 2 を**図-6**（a）に，これに対応する依存関係グラフを**図-6**（b）に示す．プログラム 2 の S3 は変数 A のために，S1 に対して出力依存の関係にあり，また S2 に対し逆依存の関係にある．したがってこのままでは S3 は S1 と S2 の後でなければ実行できない．そこで S3 の変数 A を変数 C

S1：$A=D+1$
S2：$B=A+1$
S3：$A=0$
（a）プログラム 2：変換前



（b）プログラム 2 のグラフ

S1：$A=D+1$
S2：$B=A+1$
S3：$C=0$
（c）プログラム 3：変換後



（d）プログラム 3 のグラフ

図-6 変数名変更[37)]

```
      DO  I=1, N
S1:   X=A(I)+1
S2:   B(I)=X**2
      END
```

（a） プログラム4：変換前　（b） プログラム4のグラフ

```
      DO  I=1, N
S1:   X(I)=A(I)+1
S2:   B(I)=X(I)**2
      END
```

（c） プログラム5：変換後　（d） プログラム5のグラフ

図-7 変数拡張[31]

に変更してみる．変換後のプログラム3を**図-6**(c)に，これに対応する依存関係グラフを**図-6**(d)に示す．変数名変更によって逆依存と出力依存が除去できたことがわかる．プログラム3ではS3はS1，S2と並列に実行できる．

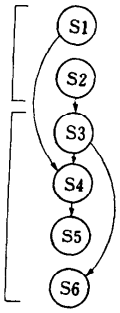
（b） 変数拡張

サイクル内にある逆依存関係を変数拡張によって除去できることを示す．変換前のプログラム4を**図-7**(a)に，これに対応する依存関係グラフを**図-7**(b)に示す．プログラム4のS2は変数XのためにS1に対してフロー依存である．くり返し演算なので，S1は変数XのためにS2に対して逆依存の関係にある．したがってグラフにはサイクルができる．各くり返し演算の間で依存関係をなくすために，変数Xを変数$X(I)$に拡張してみる．変換後のプログラム5を**図-7**(c)に，これに対応する依存関係グラフを**図-7**(d)に示す．変数拡張によって逆依存が除去されサイクルがなくなったことがわかる．プログラム5では各くり返し演算の間に依存関係がないのでベクトル処理が可能である．

（c） ループ分散

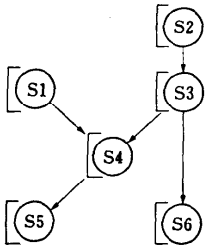
ループ制御を分散させることによって並列性を高められることを示す．変換前のプログラム6を**図-8**(a)に，これに対応する依存関係グラフを**図-8**(b)に示す．ふたつのDOループの間にフロー依存の関係があるので各ループは順番に実行される．またループ内にある各ステートメントも順番に実行される．このままではベクトルプロセッサやマルチプロセッサの効果が出ないので，各ステートメントごとにループ制御を分散させてみる．変換後のプログラム7を**図-8**(c)に，これに対応する依存関係グラフを**図-8**(d)に示す．ループ分散によって並列性が高まったことがわかる．例えばプログラム7ではS1のくり返し演算とS2のくり返し演算を並列に実行できる．したがってベクトルプロセッサならびにマルチプロセッサの効果

```
      DO  I=1, N+1
S1:   A(I)=2*I+3
S2:   READ  B(I)
      END
      DO  I=1, N
S3:   C(I)=B(I)*B(I+1)
S4:   D(I)=C(I)**2/A(I)
S5:   WRITE  D(I)
S6:   WRITE  C(I)
```



（a） プログラム6：変換前　（b） プログラム6のグラフ

```
      DO  I=1, N+1
S1:   A(I)=2*I+3
      END
      DO  I=1, N+1
S2:   READ B(I)
      END
      DO  I=1, N
S3:   C(I)=B(I)*B(I+1)
      END
      DO  I=1, N
S4:   D(I)=C(I)**2/A(I)
      END
      DO  I=1, N
S5:   WRITE  D(I)
      END
      DO  I=1, N
S6:   WRITE C(I)
      END
```



（c） プログラム7：変換後　（d） プログラム7のグラフ

図-8 ループ分散[31]

を出せるようになった．

Cedar 用のコンパイラとして Parafrase をさらに強化したものを開発中である．Cedar のコンパイラは最終的にマクロデータフロー・グラフという極めて並列性の高いプログラム・コードを生成する．マクロデータフローについては第7章で説明する．

6. アルゴリズム

並列処理システムではハードウェア資源を有効に利用し演算速度を高めるために，演算アルゴリズムとアーキテクチャの整合を図る必要がある．ここではマルチプロセッサに適した並列アルゴリズムについて，Wave-Front アルゴリズムを例にとって説明する[33]．並列処理システムに適した数値演算アルゴリズム全般の研究動向については文献 35)を参照されたい．

図-9(a)に示すプログラム8には偏微分方程式の解法でよく使われる計算式が含まれており，全体で$N\times N$回のくり返し計算を行う．各計算式を実行する時間

```
DO I=1, N
  DO J=1, N
    X(I, J)=X(I-1, J)+X(I, J-1)+C
  END
END
```

(a) プログラム8：変換前

```
DO I=1, N
  X(I,1)=X(I-1,1)+X(I,0)+C
END
DO I=1, N
  X(I,2)=X(I-1,2)+X(I,1)+C
END
  ⋮
DO I=1, N
  X(I,N)=X(I-1,N)+X(I,N-1)+C
END
```

(c) プログラム9：変換後

時間	プロセッサ1での処理
1	$X(1,1)=X(0,1)+X(1,0)+C$
2	$X(1,2)=X(0,2)+X(1,1)+C$
⋮	⋮
N	$X(1,N)=X(0,N)+X(1,N-1)+C$
$N+1$	$X(2,1)=X(1,1)+X(2,0)+C$
$N+2$	$X(2,2)=X(1,2)+X(2,1)+C$
⋮	⋮
$2N$	$X(2,N)=X(1,N)+X(2,N-1)+C$
⋮	⋮
$(N-1)N$	$X(N,1)=X(N-1,1)+X(N,0)+C$
$(N-1)N+1$	$X(N,2)=X(N-1,2)+X(N,1)+C$
⋮	⋮
$N\times N$	$X(N,N)=X(N-1,N)+X(N,N-1)+C$

(b) 逐次アリゴリズム（プログラム8に対応）

時間	プロセッサ1での処理	プロセッサ2での処理	…プロセッサNでの処理
1	$X(1,1)=X(0,1)+X(1,0)+C$		
2	$X(2,1)=X(1,1)+X(2,0)+C$	$X(1,2)=X(0,2)+X(1,1)+C$	
3	$X(3,1)=X(2,1)+X(3,0)+C$	$X(2,2)=X(1,2)+X(2,1)+C$	
⋮	⋮	⋮	
N	$X(N,1)=X(N-1,1)+X(N,0)+C$	$X(N-1,2)=X(N-1,2)+X(N-1,1)+C$	$\cdots X(1,N)=X(0,N)+(1,N-1)+C$
$N+1$		$X(N,2)=X(N-1,2)+X(N,1)+C$	$\cdots X(2,N)=X(1,N)+X(2,N-1)+C$
⋮			⋮
$2N-1$			$\cdots X(N,N)=X(N-1,N)+X(N,N-1)+C$

(d) Wave-Front アルゴリズム（プログラム9に対応）

図-9 並列アルゴリズム[33]

を1とすると，図-9(b)に示すようにプログラム8を1台のプロセッサで順番に計算する場合には $N\times N$ の時間がかかる．これに対しN台のプロセッサで図-9(d)に示すように並列計算する場合には $(2N-1)$ 時間で実行可能である．コンパイラがプログラム8を図-9(c)に示すプログラム9のように変換（ループ分散）すれば，このような並列アルゴリズム（Wave-Front アルゴリズム）を実現できる．プログラム9で2番めの DO ループの $I=1$ の計算は，1番めの DO ループの $I=1$ の計算が終了した後に実行可能となる（変数 $X(I, 1)$ で，フロー依存にあるため）．以下同様．

したがってプログラム9の i 番めの DO ループをプロセッサ i に割り付ければ，N台のマルチプロセッサにより図-9(d)に示すとおり $(2N-1)$ 時間で実行される．ただし，Wave-Front アルゴリズムでは他のプロセッサにデータを転送する必要があるために，データ転送のオーバヘッドを含めた計算式あたりの演算時間が，逐次アルゴリズムのそれに比べて大きくなる可能性があり，この点については十分考慮する必要がある．

7. 演算制御

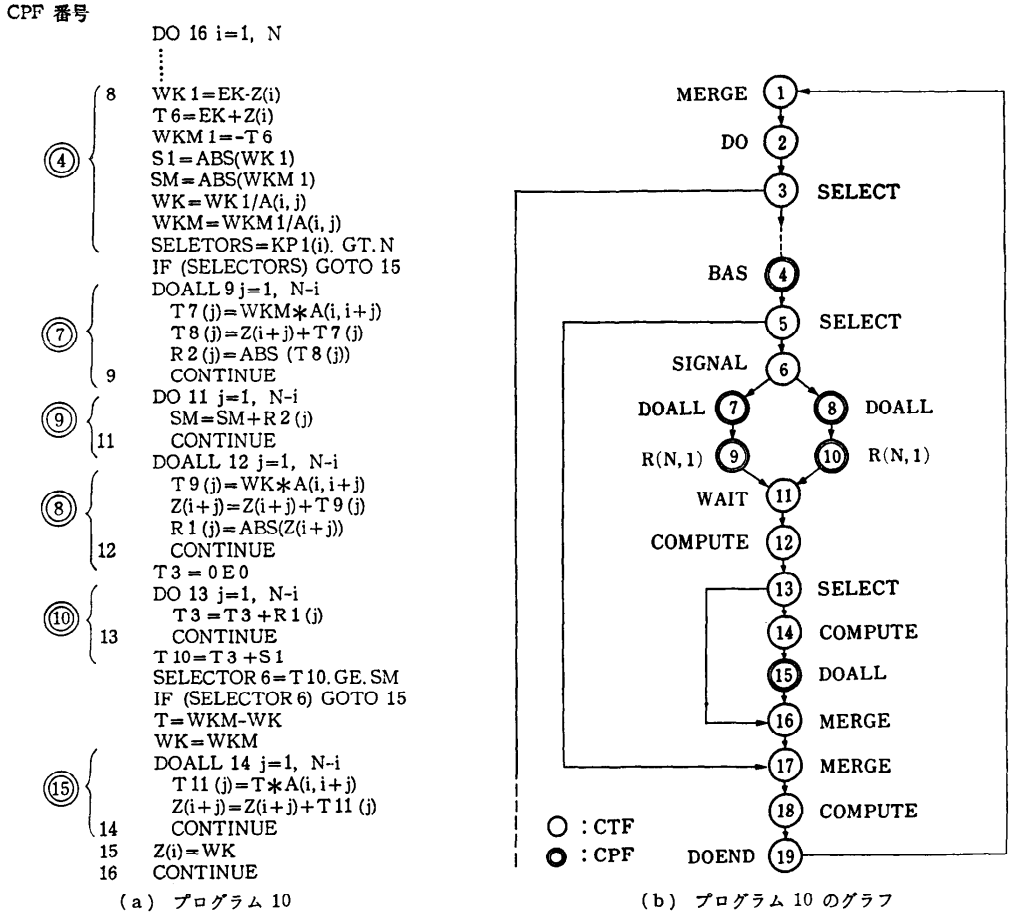
コンパイラが生成した並列性の高いプログラムをできるだけその並列性を生かしながら実行するためには相応の演算制御技術が必要となる．ここでは大規模な並列処理システムに適した演算制御技術について，**マクロデータフロー** (macro dataflow) 制御を例にとって解説する．

7.1 マクロデータフロー制御の特長

マクロデータフロー制御と従来のデータフロー制御（例えば MIT 提案のもの[18]）との比較を表-3 に示す[36),37)]．(1)演算制御の基本はどちらもデータ駆動型である．すなわちある演算について必要なデータが揃った時点でその演算は実行可能となる．(2)演算単位に関しては，従来のデータフローでは加減乗除のような比較的小さい演算を実行単位としているのに対し，マクロデータフローでは DO ループのような比較的大きな演算を実行単位としている．(3)プログラミング言語に関しては，従来のデータフローでは新たにデータフロー言語を開発する必要があるのに対し，マクロデータフローでは基本的に LISP，データ

表-3 マクロデータフロー制御の特長[37]

	マクロデータフロー（イリノイ大学）	従来のデータフロー (MIT)
演算制御	データ駆動型	データ駆動型
演算単位	大きい（例 DO ループ）	小さい（例 +，−，×，÷）
言語	主に FORTRAN	データフロー言語
ハードウェア構造	ノイマン型（プログラム・カウンタあり）	非ノイマン型（プログラム・カウンタなし）
プログラム互換性	あり（FORTRAN のとき）	なし
テスタビリティ	良い（プログラム・カウンタがあるため）	悪い（プログラム・カウンタがないため）



（a） プログラム 10　　（b） プログラム 10 のグラフ

図-10 マクロデータフロー・グラフ[13]

フロー言語を含むいかなるプログラミング言語を採用することができる．既存の応用プログラムを利用可能とするために FORTRAN が特に重要である．（4）ハードウェア構造に関しては，従来のデータフローではプログラムカウンタを持たない非ノイマン型となっているのに対し，マクロデータフローではプログラムカウンタを持つノイマン型となっている．プログラムカウンタを持っているので，プログラムカウンタを持たない従来のデータフロー制御に比べテスタビリティにすぐれている．

7.2 マクロデータフロー制御の実際

マクロデータフロー制御について具体的に説明する[12),13)]．

（a） マクロデータフロー・グラフ

ある応用プログラムを Parafrase でコンパイルしたプログラム 10 を図-10（a）に示す．プログラム 10 は FORTRAN を並列処理向きに拡張した言語で記述されている．これをもとに生成したマクロデータフ

ロー・グラフを図-10(b)に示す．マクロデータフロー・グラフはファンクション・レベルの依存関係グラフである．グラフのノードは **CF** (compound function) と呼ばれる演算単位であり，矢印の向きが CF の実行順序を規定している．CF には**計算用 CF(CPF)** と**制御用 CF (CTF)** がある．すべての CPF にはひとつの前任 CPF (predecessor) とひとつの後継 CPF (successor) が対応している．図-10(a)の左側に CPF の番号とその範囲が示してある．図-10(b)のノード内の番号がこれに対応している．図-10(b)で，BAS, DOALL, $R(N,1)$ が CPF であり，MERGE, DO, SELECT, SIGNAL, WAIT, COMPUTE, DOEND が CTF である．CF の一覧を**表-4** に示す．

マクロデータフロー・グラフは GCU によって解読される．各ノードでは **CF の演算**と**グラフ更新**というふたつの動作が行われる．CF の演算に関しては，CTF は GCU によって実行され CPF はひとつ以上のクラスタで実行される．グラフ更新とは，あるCF の演算が終了したことを後継 CF に知らせる動作に相当している．グラフ更新はすべて GCU で実行される．

(b) プログラム処理の手順

マクロデータフロー制御によるプログラム処理の手順について簡単に説明する．(1)I/O クラスタ配下のディスク内にあるコンパイル済のプログラムが GN 経由で GM にロードされる．(2)GCU コードが GN 経由で GCU にロードされる．(3)GCU は GCU コードで表現されたマクロデータフロー・グラフを解

表-4 CF (Compound Function) 一覧[13]

分類	名称	内容
CPF	DOALL	各回の演算が並列に実行できるDOループ
	R(N, 1)	1次のリニア・リカレンス
	BAS	Block of Assignment Statements
	SERIAL DO	各回の演算を順番に実行すべきDO ループ
	I/O	I/O クラスタが実行する CF
CTF	PROGRAM	プログラムの実行開始を示す
	END	プログラムの実行終了を示す
	CALL	サブルーチンを呼び出す
	SUBROUTINE	サブルーチンの先頭を示す
	RETURN	サブルーチンの終りを示す
	DO	DO ループの開始を示す
	DOEND	DO ループの終了を示す
	COMPUTE	GCU が実行する計算を示す
	SELECT	複数の後継 CF のうち一つを実行可能とする
	MERGE	複数の前任 CF のどれか一つが終了するのを待つ
	WAIT	複数の前任 CF がすべて終了するのを待つ
	SIGNAL	複数の後継 CF すべてを同時に実行可能とする

CPF: Computational Function (PC が実行する計算用 CF)
CTF: Control Function (GCU が実行する制御用 CF)

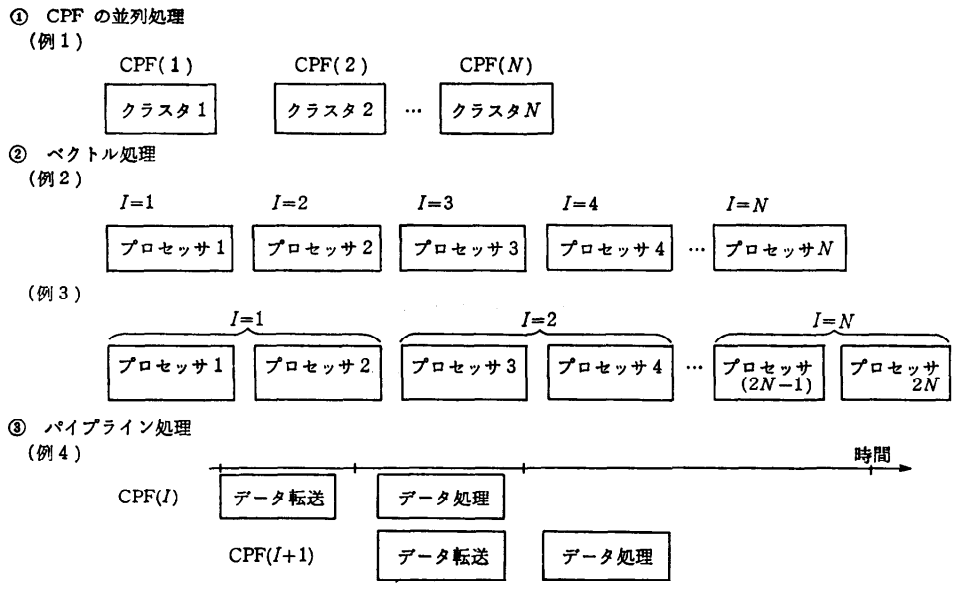


図-11 マクロデータフロー制御で期待される並列効果

読し，CTF の場合には自分自身で実行し CPF の場合には一つ以上のクラスタにその実行を割り付ける (CCU に GO 信号を送る)．(4)クラスタは GM 内にあるデータを読み書きしながら CPF を実行する．(5)CPF の演算が終了したことを GCU に知らせる (GCU に DONE 信号を送る)．(6)GM 内にある演算結果を GN 経由で I/O クラスタ配下のディスクに転送する．

(c) 期待される並列効果

マクロデータフロー制御によって期待される並列効果を次に示す．(1)複数のクラスタに割り付けられた CPF は並列に実行される (**図-11** の例1)．(2)ベクトル処理が可能な CPF (例えば DOALL) は複数台のプロセッサで並列に実行される (**図-11** の例2)．必要に応じて各回の演算を複数台のプロセッサで実行できる (**図-11** の例3)．(3)CPF の演算をデータ転送とデータ処理の2段階に分けてパイプライン処理すると，ある CPF のデータ転送とほかの CPF のデータ処理が並列に実行される (**図-11** の例4)．

8. おわりに

筆者が参加したイリノイ大学の Cedar プロジェクトを例にとって，MIMD 型スーパコンピュータの主要技術について解説した．数千台の大規模マルチプロセッサからなるスーパコンピュータを実現するためには，本稿でとりあげた技術のほかにも高密度実装技術や冷却技術が極めて重要である[10]．本資料がスーパコンピュータに興味をもつ読者にとって多少なりとも参考になれば幸いである．

謝辞

本解説で Cedar プロジェクトを引用することに対し積極的なご支持をいただいた Kuck 教授をはじめとするイリノイ大学関係者に感謝致します．また日頃ご指導いただく横須賀通研データ通信研究部新井克彦部長，苗村憲司統括役，同通信制御研究室平松琢弥室長，大島一純調査役はじめ関係各位に感謝致します．

参考文献

1) 元岡 達：スーパコンピュータの現状と展望，情報処理，Vol. 22, No. 12, pp. 1103-1110 (1981).
2) 山田 博：スーパコンピュータ，信学誌，Vol. 64, No. 5, pp. 514-516 (1981).
3) 村岡洋一，坂間保雄：並列処理技術，信学誌，Vol. 63, No. 10, pp. 1064-1071 (1980).
4) 小高俊彦，河辺 峻：超高速演算の動向，情報処理，Vol. 21, No. 9, pp. 927-937 (1980).
5) 加藤満左夫，苗村憲司：並列処理計算機，オーム社 (1976).
6) Russell, R.: The CRAY-1 Computer System, Communications of ACM, Vol. 21, No. 1, pp. 63-72 (Jan. 1978).
7) 日経エレクトロニクス, pp. 74-76 (May. 9, 1983).
8) Electronics, pp. 161-163 (Feb. 24, 1982).
9) 500 MFLOPS 級の商用スーパコンピュータが稼働へ，日経エレクトロニクス，pp. 108-126 (Apr. 11, 1983.).
10) Lyman, J.: Supercomputers Demand Innovation in Packaging and Cooling, Electronics, pp. 136-143 (Sep. 22, 1982).
11) Widdoes, L.: The S-1 Project: Developing High-Performance Digital Computers, COMPCON Spring '80, pp. 282-291 (1980).
12) Gajski, D. et al.: Construction of a Large Scale Multiprocessor, Univ. of Illinois at Urbana-Champaign, DCS Report No. 83-1123 (May 1983).
13) Gajski, D. et al.: Second Preliminary Specification of Cedar, Cedar Document No. 8, Univ. of Illinois at Urbana-Champaign, Department of Computer Science (Feb. 1983).
14) Gajski, D. et al.: Cedar-A Large Scale Multiprocessor, Proc. of the 1983 Int'l Conf. on Parallel Processing, pp. 524-529 (Aug. 1983).
15) Padmanabhan, K. et al.: Fault Tolerance Schemes in Shuffle-Exchange Type Interconnection Networks, Proc. of the 1983 Int'l Conf. on Parallel Processing, pp. 71-75 (Aug. 1983).
16) Kuck, D. et al.: Transparency Copies, Parallel Architecture Workshop 1983, Boulder, CO, Cedar Document No. 12, Univ. of Illinois at Urbana-Champaign, Department of Computer Science (Jan. 1983).
17) 小原和博：Cedar プロジェクト—イリノイ大学の新しいスーパコンピュータ計画—，信学研資，EC 83-47 (Jan. 25, 1984).
18) Dennis, J.: Data Flow Supercomputers, Computer, pp. 48-56 (Nov. 1980).
19) Manuel, T.: Advanced Parallel Architectures Get Attention as Way to Faster Computing, Electronics, pp. 105-114 (June 16, 1983).
20) Behnhard, R.: Computing at the Speed Limit, IEEE Spectrum, pp. 26-31 (July 1982).
21) 柏木 寛：スーパコンピュータ，Computrol, No. 1, pp. 131-137 (Jan. 1983).
22) 市場が急成長し始めたゲート・アレイ，日経エレクトロニクス，pp. 111-122 (Feb. 28, 1983).
23) Saigo, T. et al.: A 20 K-Gate CMOS Gate Array, Proc. of the 1983 IEEE Int'l Solid-

State Circuits Conf., pp. 156-157 (Feb. 1983).
24) Vu, T. et al.: A 500-Gate GaAs SDFL Gate Array with On-Chip RAM, Proc. of the 1983 Custom Integrated Circuits Conf., pp. 32-36 (May 1983).
25) Electronics, p. 50 (June 30, 1983).
26) Ware, F. et al.: C-MOS Chip Set Streamlines Floating-point Processing, Electronics, pp. 149-152 (Feb. 10, 1982).
27) 高橋義造：並列処理のためのプロセッサ結合方式，情報処理，Vol. 23, No. 3, pp. 201-209 (1982).
28) 並列処理システムの性能を左右する相互結合ネットワーク，日経エレクトロニクス，pp. 88-108 (Dec. 21, 1981).
29) Feng, T-Y.: A Survey of Interconnection Networks, Computer, Vol. 14, No. 12, pp. 12-27 (Dec. 1981).
30) Lawrie, D.: Access and Alignment of Data in an Array Processor, IEEE Trans. on Comp., C-24, No. 12, pp. 1145-1155 (Dec. 1975).
31) Kuck, D.: The Structure of Computers and Computations, Vol. 1, John Wiley & Sons (1978).
32) Kuck, D. et al.: Dependence Graphs and Compiler Optimizations, Proc. of the 8th ACM Symp. on Principles of Programming Languages, pp. 207-218 (Jan. 1981).
33) Kuck, D.: High-speed Machines and their Compilers, Proc. of the CREST Parallel Processing Systems Course, Evans D., Ed., pp. 193-214, Cambridge Univ. Press (Sep. 1980).
34) 梅谷征雄，高貫隆司，安村通晃：技術計算プログラムの自動ベクトル化技術，情報処理，Vol. 23, No. 1, pp. 29-40 (1982).
35) Sameh, A.: Parallel Algorithms in Numerical Linear Algebra, First Int'l Colloquium on Vector and Parallel Computing in Scientific Applications (1983).
36) Gajski, D. et al.: Dependence Driven Computation, Proc. of the COMPCON '81 Spring Computer Conf., pp. 168-172 (Feb. 1981).
37) Gajski, D. et al.: A Second Opinion on Data Flow Machines and Languages, Invited paper, Special Dataflow Issue, IEEE Computer, Vol. 15, No. 2, pp. 58-69 (Feb. 1982).

(昭和59年2月28日受付)