High Level Data Flow Machine (TOPSTAR) のシステムプログラム
System Program of a High Level Data Flow Machine

栗原　謙　鈴木達郎　元岡　達
Ken KURIHARA　Tatsuo SUZUKI　Tohru MOTO-OKA
東京大学工学部
Faculty of Engineering, University of Tokyo

1. はじめに

TOPSTARは，プロシージャレベルのデータフローによって分散制御される，マルチマイクロプロセッサシステムである。

データフローによる制御は，高度の並列処理性を可能にするのみならず，データ駆動による制御の非同期性と局所性の観点からも，大規模システムの分散制御方式として適している。

J.B.Dennisら[5]のシステムは，命令レベルのデータフローマシンであるが，本研究は，従来のノイマン型マイクロプロセッサ多数より成るシステムの制御に，プロシージャレベルのデータフローを用い，分散制御の下で高度の並列処理性の実現を目指したものである。

ここでは，TOPSTARの制御方式と，それを実現するシステムプログラムについて報告する。

2. TOPSTARのアーキテクチャ

TOPSTARのアーキテクチャの特徴は，以下の3点にある。(詳細は参考文献[1]~[4]を参照されたい。)

1) C-Module (Communication / Control Module) とP-Module (Processing Module) の2種類のモジュール多数より成り，それぞれがペトリネットの "place" と "transition" に対応する。

2) C-ModuleとP-Moduleの結合は，大規模システムの実現のため，部分結合とする。この際，データフローネットワークとの対応を柔軟に行なうために，この結合をオーバラップさせ，可変構造を可能にしている。

3) プロシージャレベルのデータフロー処理においては，比較的多量のデータを扱うので，DMAによってメモリ対メモリの高速転送を行なう。

図1にシステムの構成を示す。

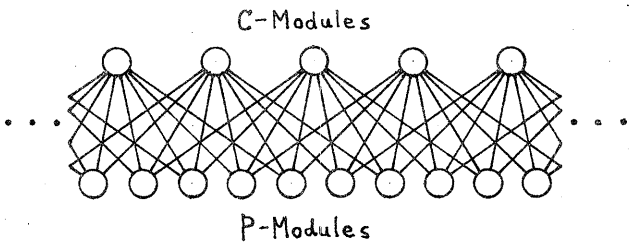


図1　TOPSTARシステムの構成

(注)

1) 以前の文献で用いていたSAMD処理 (Single Algorithm-stream Multiple Data-stream) は，一般のデータフロー処理に包含されるので，ここでは，SAMDという名称は用いない。

2) C-Moduleは，以前はM-Module (Memory Module) と呼ばれていた。

3. プロシージャレベルデータフロープログラム

TOPSTAR の処理対象は，プロシージャをノードとする，データフロープログラムである。図 2 (a) にその一例を示す。一方，このデータフロープログラムは，同図 (b) のペトリネットと等価である。

TOPSTAR においては，このデータフロープログラムは，同図 (c) の様に実行される。ペトリネットの "place" は，C-Module の中のバッファに対応し，"transition" は，P-Module が "雇われて仕事を行なう" ことに対応する。ここでは，"bar" は，その中に P-Module が "入り込む" という意味で，むしろ "box" と呼ぶことにする。

4. TOPSTAR の制御方式

TOPSTAR の制御方式の特徴としては，以下の 4点が挙げられる。

1) データ駆動による分散制御
2) P-Module の自由競争による負荷分散
3) 追い越しを許すパイプライン処理
4) ループ，選択，再帰呼出し等のプログラム構造が使用可能

本章では，これらの特徴を中心に，制御方式を解説する。

4.1 システムの動作の概要

各 P-Module は，(結合している範囲の) 1つの C-Module に割り込みをかけ，(注1) DEQ (DEQueue) コマンドを送ることにより，"仕事" を要求する。各 C-Module には，いくつかのノードが割り当てられており，DEQ コマンドを受けた C-Module は，管理テーブルを調べて，実行可能な "仕事" があれば，そのプロシージャ (注2) とデータ，及び結果の送り先の情報を P-Module に送る。実行可能な "仕事" がない場合には，その旨知らせる。

P-Module は，"仕事" がもらえなかった場合には，他の C-Module に尋ねて回る。"仕事" をもらった P-Module は，その "仕事" を実行し，結果の送り先のノードが割り当てられている C-Module に割り込みをかけ，(注1) ENQ (ENQueue) コマンドを送り，(注3) ひきつづいて，結果のデータを送る。idle になった P-Module は，再び "仕事" を求めて C-Module に割り込みをかける。

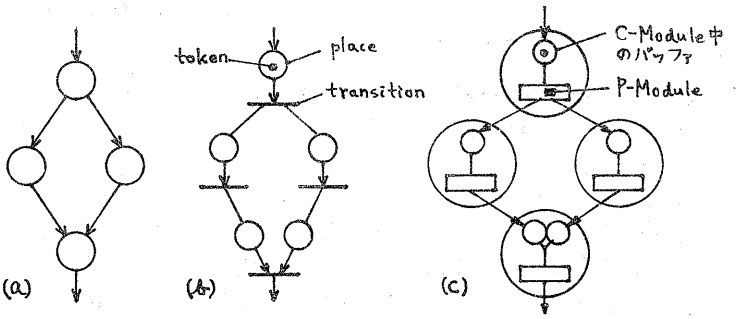
以上の様に，各 P-Module が "仕事" を求めて (オーバラップした部分結合の範囲で) 自由競争を行なうことにより，"忙しい" (実行可能な "仕事" が多く存在する) C-Module に，自然に多くの P-Module が集まり，適切な負荷分散が行なわれる。

ここで言う "実行可能" とは，そのノードが必要とするすべての入力データがそろっているということを意味する。即ち，データ依存性 (data dependency) のみによって制御が行なわれる，データ駆動方式が用いられている。また，データ依存性の管理は，各々の C-Module が単独に行なえ，集中的な制御は不要である。

(注1) この際の競合は，PIC (Programmable Interrupt Controller) によって解決される。
(注2) P-Module が前回実行したプロシージャと同一ならば，送る必要はない。P-Module は，そのプロシージャを優先指定する。
(注3) これにより，C-Module は，データの格納場所の計算等を行なう。

図 2

(a) データフロープログラムの例
(b) 対応するペトリネット
(c) TOPSTAR における処理



4.2 フローコントロール

データフロー処理において，C-Module が行なう制御は，原理的には極めて簡単である。即ち，必要な入力データがそろった"仕事"を，次々とP-Module にDEQしてやればよい。

しかし，実際には，結果の送り先のC-Module 中のバッファは有限であり，これがオーバフローしない様に制御してやらねばならない。これがフローコントロールである。

TOPSTAR の場合，P-Module が，C-Module のバッファ空きを待つ様にすると，デッドロックを生じる恐れがあり，また，C-Module が全ての管理を行なうという立場をとっているので，各C-Module は，結果の送り先のC-Module のバッファに空きがある時にのみDEQ を行なう。

このために，C-Module は各ノード毎にセマフォを持っている。セマフォは，0から次段のノードのバッファサイズまでの値をとる変数であって，DEQ 時に1減じられ，P-Module からのV-OP（V-OPeration）コマンドを受けた時に1ふやされる。V-OP コマンドは，次段のノード（を管理しているC-Module）からDEQ をうけたP-Module が，前段のノード（を管理しているC-Module）に対して送るものである。

セマフォは，次段のノードのバッファの空き数を示しており，C-Module は，セマフォの値が0の時には，DEQ を行なわない。

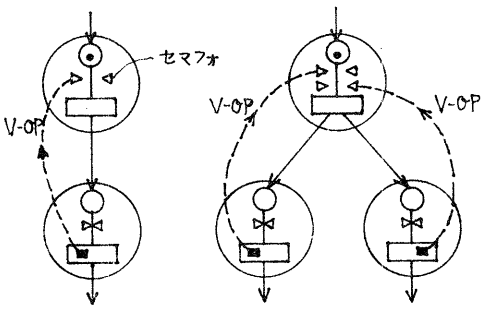


図3　セマフォによるフローコントロール

4.3 パイプライン処理と追い越し問題

マルチプロセッサシステムにおける並列処理性には，パイプライン処理による，いわば縦方向の並列処理性，及び，独立した演算が並行して行なわれる，いわば横方向の並列処理性がある。

一般に，パイプライン処理の制御方式としては，FIFOが用いられている。しかし，TOPSTAR においては，C-Module と P-Module に分かれていることにより，複数の P-Module に順次DEQ を行なっても，その処理結果が，DEQ した順に次段のノードにENQ されるとは限らない。即ち，追い越しが起こってしまう。

データによって処理時間が大きく異なる様な場合，この様な追い越しを許すことにより，並列処理性が向上すると考えられる。

以上の理由から，TOPSTAR においては，追い越しを許す制御方式を採用している。

FIFO以外の制御方式においては，各データには，Serial Number（SN）を施して識別する必要がある。これは，いわゆる colored token の考え方である。

しかし，バッファ数は有限であるので，追い越しを無制限に許すと，FORK-JOIN の際にデッドロックを生じる。（図4 参照）

この様なデッドロックを生じないフローコントロール方式として，"Reserve one Buffer for the Most Delayed serial number"（RBMD）方式を用いている。これは，あるノードのバッファ数が k 個あるとき，通常は k-1 個のみを使用し，残りの1個は，最も遅れているSNのトークンのために取っておくと

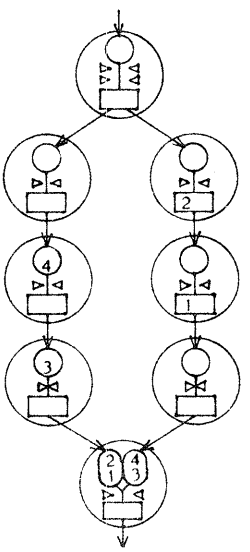


図4
追い越しによる FORK-JOIN の際のデッドロックの例
（バッファサイズ＝2）

いう方式である。これにより，そのトークンは，必ず先へ進んで行けるので，JOINノードで待っている，同じSNを持つトークンとJOINすることができる。

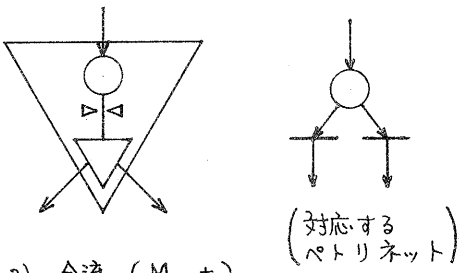
アルゴリズムを以下に示す。

```
var Semaphore : 0..NextQsize ;
    MostDelayedSN : integer ;
    Gone : packed array [0..PassingLimit]
           of Boolean ; (* bit pattern *)
if ( (Semaphore > 0)
     and
    (OldestFireableSN = MostDelayedSN) )
or
   ( (Semaphore > 1)
     and
    (OldestFireableSN - MostDelayedSN
                      <= PassingLimit) )
then begin
   Gone [OldestFireableSN - MostDelayedSN]
     := true ;
   if Gone [0]
   then repeat
     shiftleft( Gone ) ;
     inc( MostDelayedSN )
   until not Gone [0] ;
   dec( Semaphore ) ;
   dequeue( OldestFireableSN )
end ;
```

4.4 プログラム構造

データフロープログラムにおいても，選択，ループ，再帰呼出しなどのプログラム構造を使用できることが望まれる。このために，以下の2種類の制御ノードを設ける。

1) 条件分岐 (Cond. Branch)



2) 合流 (Meet)

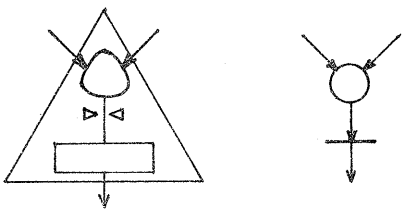
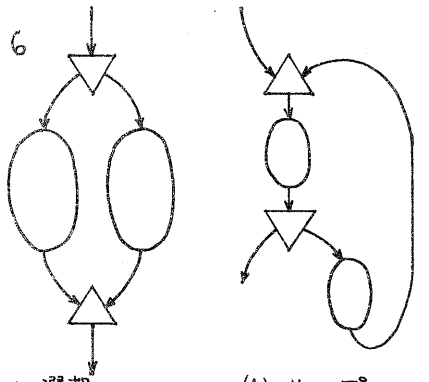


図5　条件分岐ノード及び合流ノード

これらのノードを用いると，選択及びループは，図6の様に実現される。

尚，ここで言うループとは，繰り返しを本質的にシーケンシャルに実行しなければならないもののことである。

図6



4.5 選択，ループにおけるフローコントロール

選択やループの構造の中では，RBMD方式のフローコントロールを用いることはできない。なぜならば，選択においては，一方のパスを通過するSNはとびとびになり，ループにおいては，同一のSNが何度もやって来るからである。

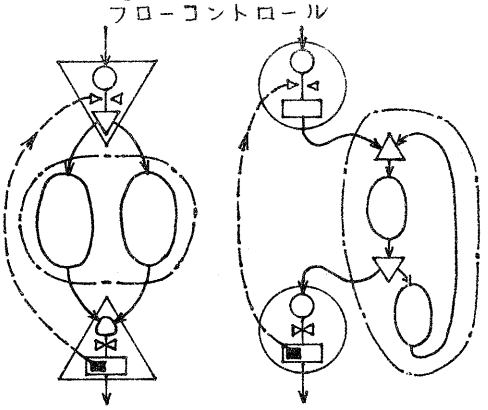
選択において，選択されたパス以外のパスに，そのパスにはそのSNが来ないという情報を流してやれば，RBMD方式が使えるが，それでは実質的に，FORK-JOINと変わらなくなってしまう。

ここでは，図7の様に，"出口"のノードから"入口"のノードへV-OPを行ない，"内部"に入るトークンの数を制限してしまうという方法を用いている。

"入口"と"出口"のノードは，通常のRBMD方式のフローコントロールを行なっていればよい。"内部"の各ノードは，"出口"のノードのバッファ数と同じだけのバッファを備えておけばよく，そうすれば，"内部"のノード間でV-OPを行なう必要はない。

FIFO制御においては，"内部"にはトークンが1個しか入れないことに比べれば，並列処理性は向上している。

図7　選択及びループにおける
フローコントロール



4.6　再帰呼出し

本システムにおいて，再帰呼出しを実現するためには，ノードにスタックを持たせることも考えられるが，ここでは，データ自体に履歴を持たせる方法を採用している。

再帰的なプロシージャ呼出しを含むデータフロープログラムは，以下の規則に従い変換する。

1) "入口" と "出口" にそれぞれPUSH，POPノードを設ける。

2) 再帰的にプロシージャへ入っていくリンクを，代わりに，PUSHノードへ接続する。

3) 再帰的にプロシージャから帰ってくるリンクを，代わりに，POPノードから出てくることにする。

図8に示す様に，データそれ自体がスタックを持っており，ノードは履歴を持たない。PUSHノードでは，戻り先情報をPUSHし，POPノードでは，POPした戻り先情報に従って分岐する。

図9にアッカーマン関数のプログラムを示す。同図において，JOINノードでは，図8のSNとSPを合わせたものを新たなSNとみなして，それが等しいもの同志をJOINすればよい。一般に，POPノードの出力が入っていくノードでは，大量のデータがJOINの相手を待つことになるので，充分な大きさのバッファを用意しなければならない。

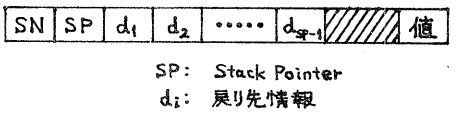
尚，再帰呼出しにおいても，RBMD方式のフローコントロールは使用出来ないので，"出口" から "入口" へのV-OPを行なわねばならない。

4.7　並列再帰呼出し

前節の再帰呼出しの実行はシーケンシャルであった。しかし，ツリー状のデータ構造の処理など，再帰呼出しが並列に行なわれる場合がある。

この場合，JOINは，SN及びSPの値が等しいというだけで行なうことはできず，図8のd_1からd_{sp-1}までの履歴が全て等しいか否かを見なければならない。

また，ツリー状にひろがる再帰呼出しは，Width First に実行したのでは，たちまちバッファがオーバフローしてしまう。Depth 方向とWidth 方向のひろがりを適当に制御することによって，バッファがオーバフローしない範囲で並列処理性を最大限にとり出し得ることが望まれるが，今のところ，その制御は実装されていない。



SP:　Stack Pointer
d_i:　戻り先情報

図8　再帰呼出しにおけるデータフォーマット

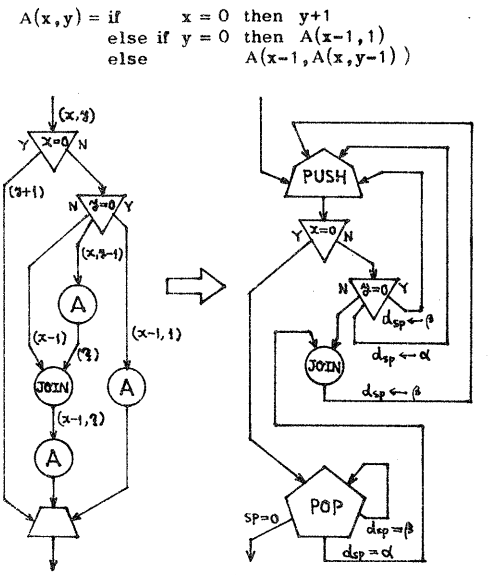


図9　アッカーマン関数のデータフロープログラム

4.8 部分結合に関する問題

システムが大規模になると，多数のC-ModuleとP-Moduleの間を完全結合することは事実上不可能となる。よって，本システムでは，データフローネットワークの局所性を仮定して，部分結合を採用している。（具体的には，1個のC-Moduleに8～16個のP-Moduleが結合）

しかし，ネットワークの局所性の範囲が，ハードウェアの部分結合の範囲を越える場合がある。また，P-Moduleの自由競争による問題もある。これらを以下の様に解決している。

1) "仕事"は，任意のP-ModuleにDEQしてよい訳ではなく，ENQ及びV-OPが届くもののみにDEQしなければならない。

P-Moduleを選択するために，MASKというビットパターンを用いる。これは，I/O等が接続されているP-Moduleの選択にも使用される。

2) リンクでつながっている2つのノードを，どうしても，遠く離れた2つのC-Moduleに割り当てなければならない場合がある。

この場合には，Relayノードを作りつけて中継させる。

3) 選択やループなどでは，遠くのノードへV-OPを行なわなければならない。

このためには，V-OP Relayノードを作りつけて，V-OPコマンドを中継させる。

効率の観点から，多くの中継は好ましくない。ノードのC-Moduleへの最適な割り当て手法は今後の研究課題である。

5. システムプログラムの実装

現在，前章で述べた制御方式に基くシステムプログラム Version 2 をTOPSTAR-1に実装中である。本章では，その概要を，主に図を用いて説明する。

（注）

TOPSTAR-1は，C-Module 2台，P-Module 3台より成る，プロトタイプシステムである。

また，システムプログラムVersion 1 は，シングルバッファによるFIFO制御であり，ループ等のプログラム構造も許していない。

システムプログラム Version 2 の制御テーブルの構成を図11 に示す。

データフロープログラムの各ノードには，ユニークなノード番号が付けられる。また，1つのノードの入出力ポートには，それぞれ，一連の番号が付けられる。

各ノードは，それぞれのNode Control Block（NCB）によって管理される。但し，V-OP Relayノードは，V-OP Relay Node Control Block（VRNCB）によって管理される。

NCB及びVRNCBの構成を図12，13に示す。

図14に，データに付されるヘッダの形式を示す。ヘッダは，P-ModuleがV-OP及びENQを行なうために必要な情報から成る。

図15は，NCB内のENQTBとSYNCTRの働きを例示したものである。最初にやって来たデータの入力ポート番号によって，caseを知ることができ，それに従ってSYNCTRのBITPAT を設定する。

P-Module及びC-Moduleの動作のフローチャートをそれぞれ図16，17 に示す。

図10 は，P-ModuleがC-Moduleに送るコマンドの形式を示したものである。

ENQ コマンドのENQER NDNOは，デバッグ用に設けたものである。また，V-OP コマンドのVOPOR NDNO もなくして，SEMTBのBITPAT を単なる減算カウンタに代えてもよい。

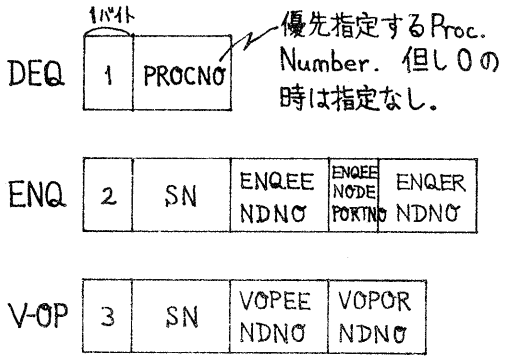


図10 コマンドの形式

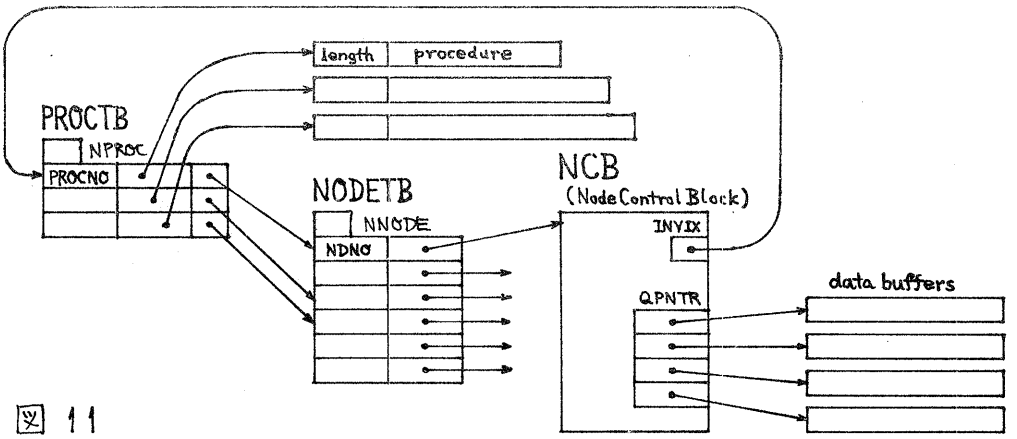


図 11
制御テーブルの構成

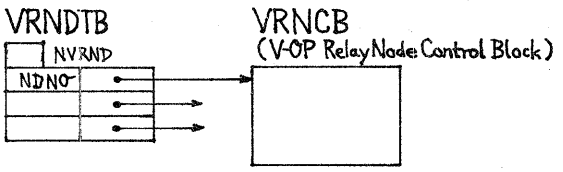


図 12 NCBの構成

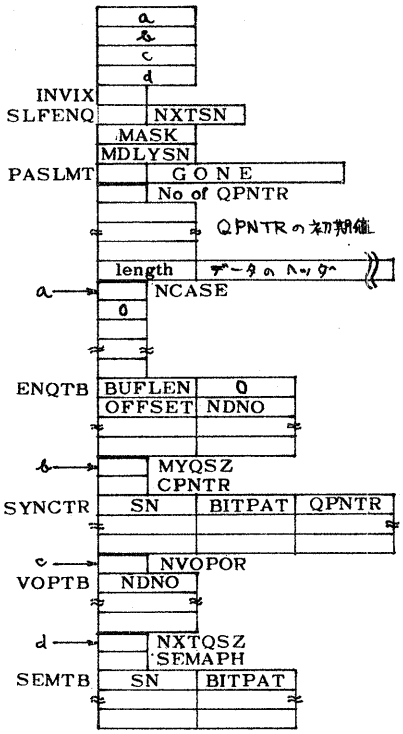


図 13 VRNCBの構成

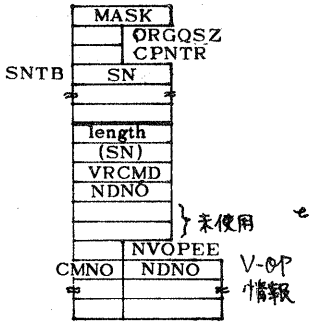
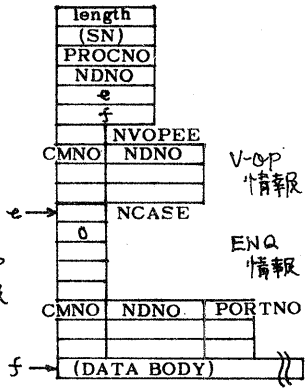


図 14 ヘッダの構成



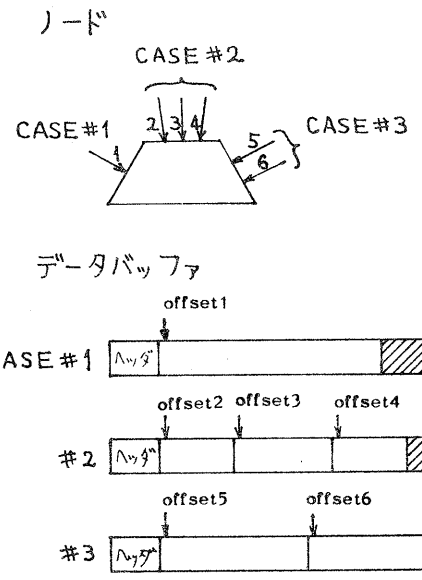


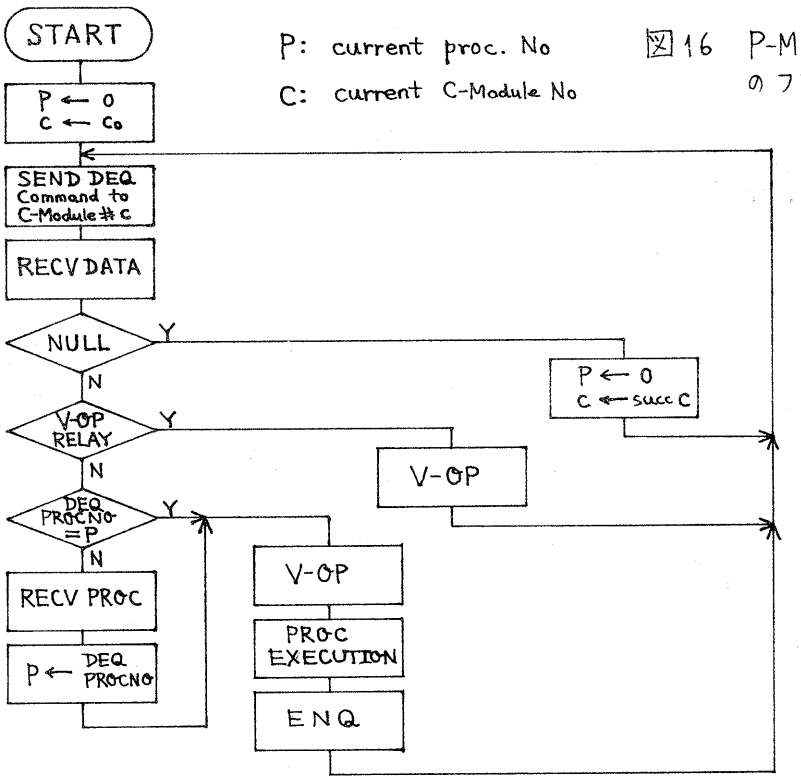
図15 C-Moduleにおける合流及びJOINの制御の例

ENQTB

NCASE	3	
	0	
	1	
	4	
	6	
	buflen	0
PORT #1	offset1	NDNO
2	offset2	NDNO
3	offset3	NDNO
4	offset4	NDNO
5	offset5	NDNO
6	offset6	NDNO

SYNCTRのBITPATの初期値

CASE#1	1	0			
#2	0	1	1	1	0
#3	0	1	1	0	



P: current proc. No
C: current C-Module No

図16 P-Moduleの動作のフローチャート

図17　C-Moduleの動作のフローチャート

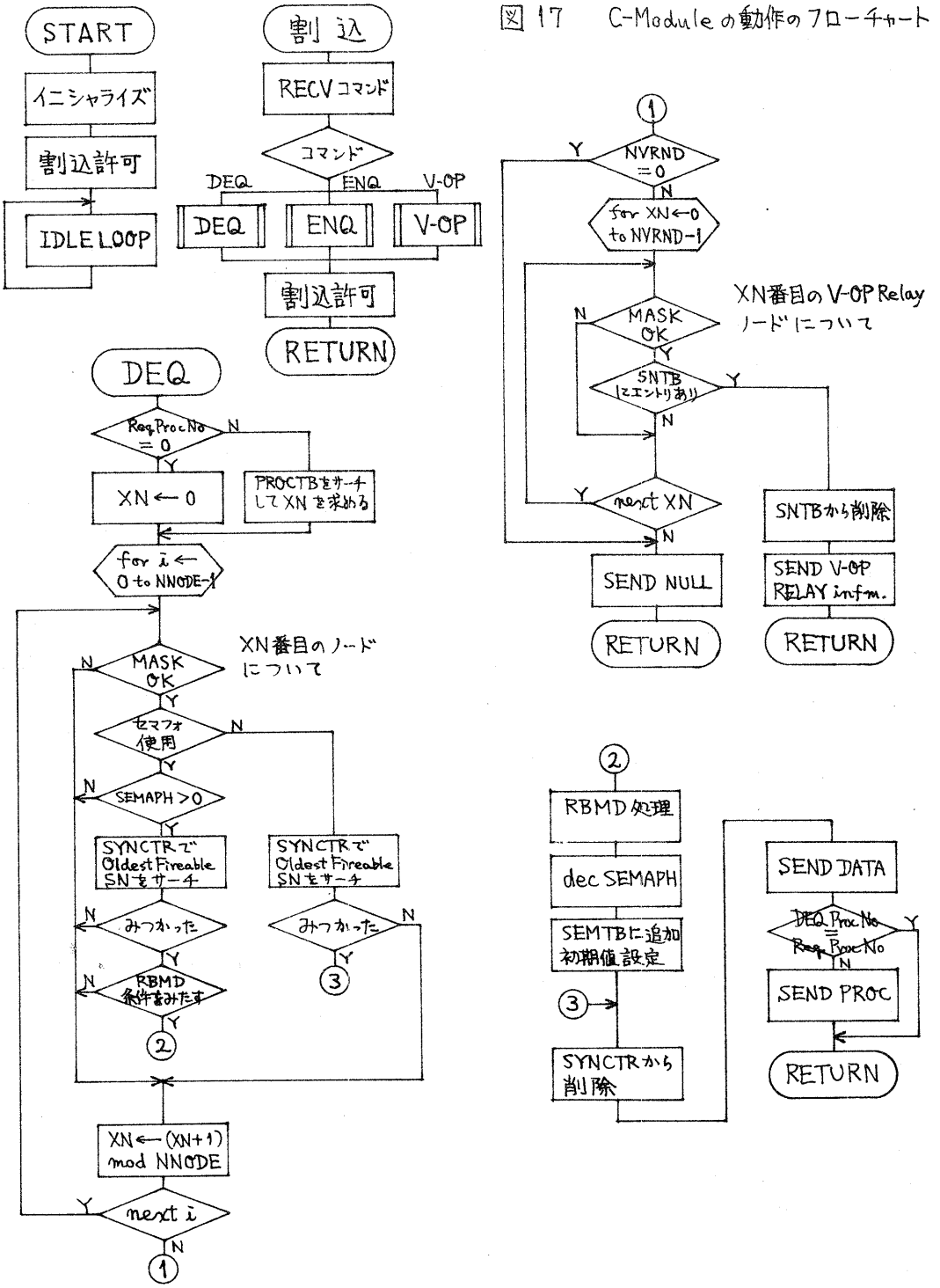
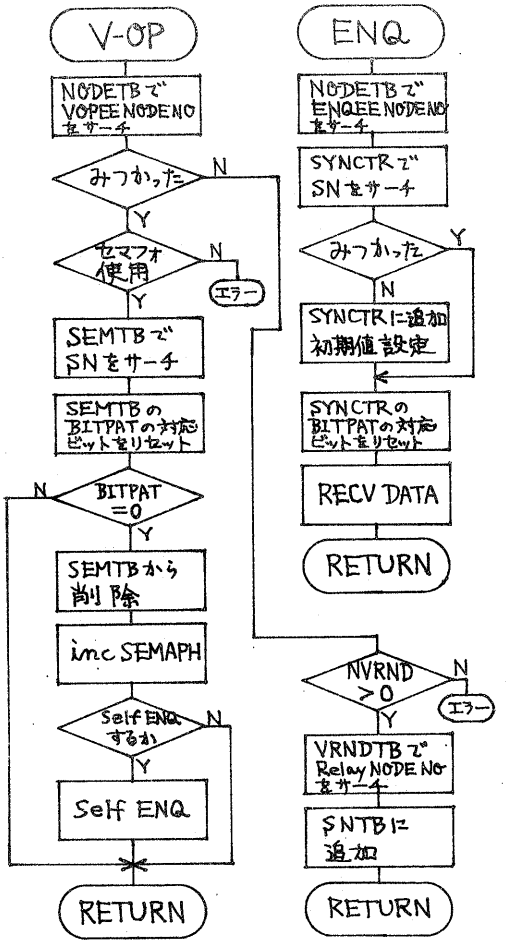


図 17 （つづき）



(注) Self ENQ とは，入力ポートのないノードにおいて，V-OP によって，新しいSNのトークンを生成すること。

6. おわりに

プロシージャレベル・データフローマシン TOPSTAR の制御方式とシステムプログラムについて述べた。

本システムプログラムにおいては，パイプライン処理での追い越しを許す等，データ依存性以外の制約を大部分排除した。

さらに，プログラム構造として，選択，ループ，再帰呼出しの使用を可能にし，プログラム作成時の制約を緩和した。

今後の課題としては，

1) 複数バッファを使用し，追い越しを認めることによる，効率向上の評価，測定，GPSS によるシミュレーション

2) 選択，ループ，再帰呼出し等を活用した，アプリケーション・プログラムの開発

3) ノード間の結合関係を記述する言語の設計及び，それによって，各ノードの C-Module への割り当て，制御テーブルの生成を行なうコンパイラの開発

等が挙げられる。

尚，TOPSTAR システムにおける，他のプロジェクトとして，要求駆動 (demand driven) による pure Lisp の実装を行なっている。

また，C-Module 8台，P-Module 16 台から成る TOPSTAR-2 の設計製作も行なっている。

参考文献

[1] 元岡，鈴木，「SAMD」，昭54電気学会全国大会シンポジウム，pp. S13-13～S13-16.

[2] 元岡，鈴木，喜連川，新岡，「SAMD 計算機～A High Level Data Flow Machine～」，情報処理学会アーキテクチャ研究会資料34-1，'79年5月。

[3] 元岡，喜連川，鈴木，「A High Level Data Flow Machine ～ hardware ～」，昭54 情報処理学会全国大会講演論文集，pp. 11-12.

[4] 元岡，鈴木，喜連川，「A High Level Data Flow Machine ～ software ～」，同上，pp. 13-14.

[5] J.B. Dennis and D.P. Misnus, "A Computer Architecture for Highly Parallel Signal Processing," Proc. ACM Annual Conf., 1974, pp. 402-409.