北海道大学工学部研究報告
第119号（昭和59年）

Bulletin of the Faculty of Engineering,
Hokkaido University, No. 119 (1984)

離散型システムシミュレーションの並列処理手法

後藤　環*　土肥　俊**　石動　善久**　牧野　圭二***

（昭和58年9月30日受理）

A Parallel Processing Technique for a Discrete System Simulation

Tamaki GOTOH, Shun DOI, Yoshihisa ISURUGI and Keiji MAKINO

(Received September 30, 1983)

Abstract

This paper is a study of a parallel processing technique for a discrete system simulation of GPSS type using a parallel processing system which has a one dimensional array of processors.

In a discrete system simulation, transactions, which reprenent temporary entities of a model, move through a network of blocks, which represent permanent entities of the model and perform given functions to transactions. Paying special attention to the fact that the activation of a block is caused by the arrival of a transaction at the block independent of others, we propose a parallel processing method, in which we assign each block to an appropriate processor to form the network of blocks on the array of processors and make each processor execute its assigned block in a parallel manner. We also propose a preprocessing algorithm, which computes the assignment of blocks to processors suitable for the above parallel processing.

A practical goal of our study is to improve the performance of the Realtime Interactive System Simulator (RISS) using the parallel processor Array (PPA) ;both are available at the Simulation Center of Hokkaido University.

1．はじめに

計算機シミュレーションはわれわれをとりまく自然・社会における各種現象の解析に用いられ，多くの成果をあげている[1]。シミュレーションの対象を記述するシミュレーション言語は，対象の状態変化を時間軸に対して連続的なものとしてとらえるか離散的なものとしてとらえるかによって，連続型と離散型に大別されるが，後者の代表とも言えるGPSS(General Purpose Simulation System)は待ち行列で特徴付けられる現象の解析に用いられ，世界中で最も広く使われている。

本学の共同利用施設である北海道大学汎用シミュレータ施設には，このGPSS系統の離散型システムシミュレータRISS (Realtime Interactive System Simulator) [2]があり，各種離散型システムのシミュレーションに利用できる。RISSはGPSSと同様の言語仕様を持っており，プログ

*㈱北海道新聞社勤務
**精密工学科 自動制御工学講座
***共同利用施設 汎用シミュレータ施設

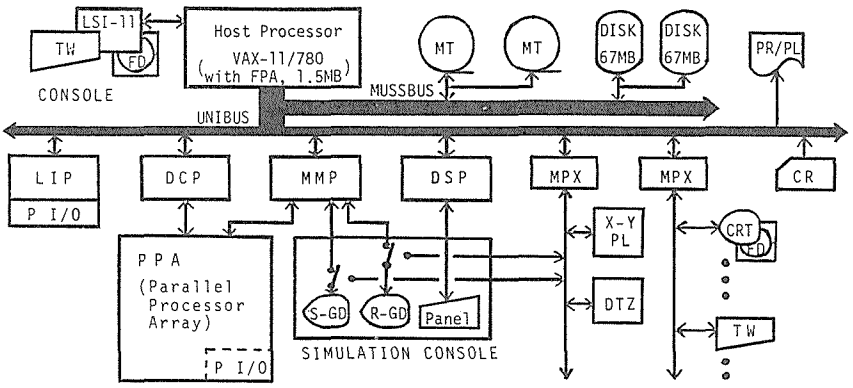
ラミングが容易であるが，さらに専用のエディタを持つ等，プログラム作成からシミュレーション実行，解表示に至るまで一貫して会話形式で作業を進めることができるよう配慮されている。しかしながら，システムシミュレータ全般にあてはまることではあるが，シミュレーションの対象が大規模になるとシミュレーション実行に莫大な計算時間がかかるため計算の高速化が強く望まれる。

一方，計算の高速化の一手法として多数のプロセッサを用いて計算を行う並列処理があり，性能価格比の点における優位性もあいまって，高速演算の有効な手段となりつつある。実際，汎用シミュレータ施設に設置されている並列処理システム PPA (Parallel Processor Array) [3] は上述の連続型システムシミュレーション[4] 及び有限要素シミュレーション[5] に利用され，その優秀性が実証されている。そこで本稿では PPA のような一次元プロセッサアレイ方式の並列処理システムを用いて離散型システムシミュレーションを行うための並行処理方式について検討を加え，さらに PPA 上での実現について述べる。

本稿の 2 章では使用する計算機システム HOSS (Hokkaido University High-Speed System Simulator) とそのサブシステムである並列処理システム PPA の概要を述べる。PPA は連続型システムシミュレーションの高速化を主目的として開発された並列処理システムであるが，有限要素法への応用や本研究等，新しい分野への応用について研究が進められている。3 章では RISS についてその概要を紹介する。4 章では RISS 等，単一計算機上で動作する離散型システムシミュレータの基本的な実行制御方式を概説し，その並列化の方針を示す。さらに並列化に伴って生じる計算負荷のプロセッサへの割当ての問題について，簡単かつ実用的解法を提案する。最後に，これらの並列処理方式を用い，PPA 上に現在試作中の離散型システムシミュレータの概要を紹介する。

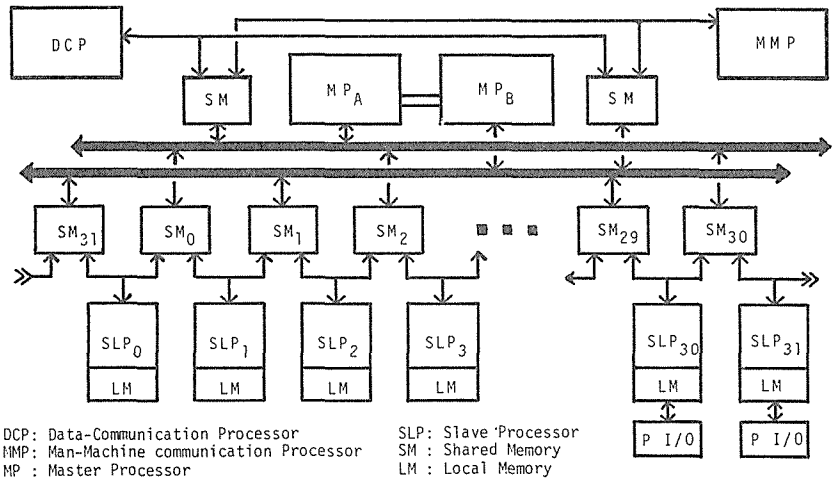
2．高速システムシミュレーション装置 HOSS とその並列処理サブシステム PPA

HOSS は DEC 社の VAX-11/780 をホスト計算機とし，その他約 40 台のミニ計算機 (DEC 社の PDP-11/34 相当) から成る計算機複合体である (図 1)。現在これらのミニ計算機のうち 1 台が離散系プロセッサ DSP (Discrete System Simulation Processor) として離散型システムシミュレータ RISS のシミュレーション実行専用に使われている。一方，34 台のミニ計算機が 1 次元プロ



LIP: Laboratory Interface Processor
DCP: Data-Communication Processor
MMP: Man-Machine communication Processor
DSP: Discrete-system Simulation Processor

図－1 HOSS のハードウエア構成



図－2 PPA のハードウエア構成

セッサアレイ方式の並列処理システム PPA を構成し，連続型システムシミュレータ ICOSS (Interactive Continuous System Simulator) により使われているが，その高速演算能力は他の様々な問題に利用できる。

ホスト計算機は DSP や PPA を使用するさい必要なプログラム，データの入力及び実行制御をはじめとした，HOSS 全般にわたる監視や制御を行う。以下に PPA の主な特徴を示す (図 2)。

(1) PPA は 32 台のスレーブプロセッサ SLP (Slave Processor) とそれらを制御する 2 台のマスタプロセッサ MP (Master Processor) から成り，各プロセッサが独自にプログラムを実行する MIMD (Multiple Instruction Stream Multiple Data Stream) 型の並列処理システムである。

(2) 32 台のスレーブプロセッサ SLP_i ($i=0$, …, 31) は 1 次元の循環アレイを構成する。各 SLP_i は共有メモリ SM_{i-1}, SM_i (Shared Memory) をアクセスすることができ，それによって 2 台の隣接するスレーブプロセッサ SLP_{i-1}, SLP_{i+1} と直接交信する。

(3) マスタプロセッサ MP-A, MP-B はそれぞれ偶数番，奇数番の SM を自分のメモリとしてアクセスすることができる。従って相隔たった SLP 間のデータ転送は MP によるメモリ上のデータの移動により容易に行われる。また各 MP は SM を介して偶数番，奇数番の SLP を制御する。マスタプロセッサ同士は専用のレジスタを介して交信する。.

(4) 各 MP はそれぞれ支配下にある任意複数台の SLP に同時に起動をかけることができる。一方それら任意複数台の SLP 全てがあるレジスタに書き込みを行ったときそのことが MP に通知され，それと同時に，書き込まれた内容の論理和が通知される。また各 MP は支配下の全 SM に一斉に同一のデータを送るブロードキャスト機能を持つ。

PPA に関する詳しい説明は文献 3) に述べられている。

3．離散型システムシミュレータ RISS

RISS は本学汎用シミュミレータ施設に設置されている高速システムシミュレーション装置 HOSS 上で稼動する離散型システムシミュレータであり，GPSS 系統のモデル構成概念と言語仕様を持っている。即ち，シミュレーション対象のモデルを構成する要素は，他からサービスを受ける客等を表す一時要素（トランザクション）と，この一時要素に対してサービスを行う設備等を表す恒久要素から成り，恒久要素はさらに複数個のブロック命令（あるいは単に，ブロック）

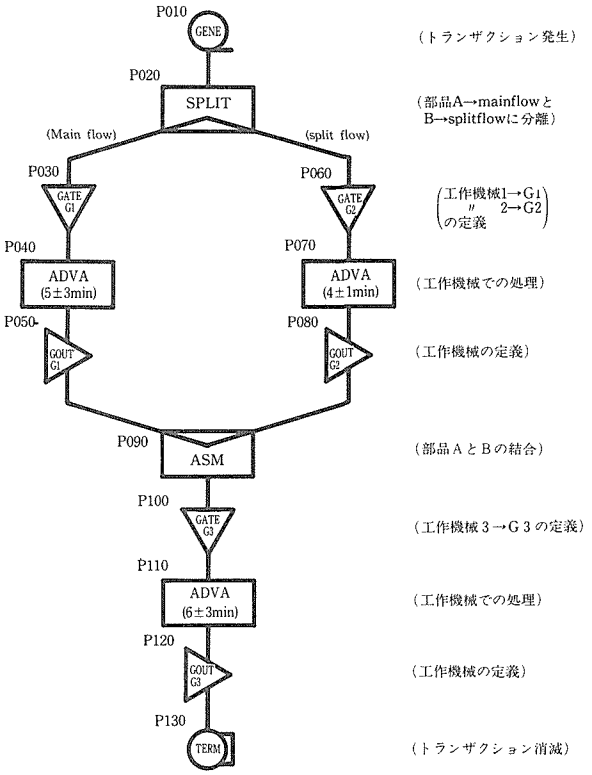
で記述される。シミュレーションは複数個のブロックで構成されるネットワーク上をその流れに沿ってトランザクションが動きまわることで行われる。シミュレーションプログラムはネットワークを構成する各ブロックの属性とそれらの接続関係を規定するブロック文の集りである。その基本的な書式は次のように表される。

ブロック文番号　ブロック命令　属性値，…，次のブロック文番号

ここでRISSのモデル化を簡単な生産工場を例にとり示す。

「工場に製品を作るための部品A，Bが6±4分間隔で一緒に送られてくる。部品Aは工作機械1で5±3分の処理を受け，Bは工作機械2で4±1分の処理を受けた後，両者は結合されて，さらに工作機械3で6±3分の処理を受けて製品となる。」

この例題のブロック図を図3に，またRISSプログラムを図4に示す。



図－3　シミュレーションモデル例のブロック図

さて図4のようにコーディングされたプログラムは，ホスト計算機上でRISS専用エディタにより入力され，RISSトランスレータにより中間コードに変換された後，離散系プロセッサDSP上で実行制御プログラムにより実行にうつされる。一般にシミュレーションの過程には多くの試行錯誤が含まれるが，RISSではこのような試行錯誤を伴った一連の作業が一貫して会話形式で行えるよう配慮されている。またシミュレーション実行を独立したプロセッサで行うことにより，実時間との対応が良い。しかしながら，シミュレーションモデルが大規模になると莫大な計算時間がかかり，計算の高速化が望まれる。次章では，計算の高速化を目的とした離散型システムシミュレーションの並列処理方式について述べる。

4．離散型システムシミュレーションの並列化

4.1　単一計算機におけるシミュレーションの実行制御方式

シミュレーションモデル内に存在するトランザクションはそれぞれ独立にサービスを受け，ブロックで構成されるネットワーク上を動きまわる。このような同時並行現象を処理するためにシミュレータは，シミュレーションの時刻を管理するシミュレーションクロックを一旦停止し，その時刻に動きうる全てのトランザクションがADVANCEブロック（付録参照）による時間消費またはGATEブロック（同参照）における待ち等によって動けなくなるまでネットワーク内を移動させる。そして動きうるトランザクションがなくなったとき，シミュレーションクロックを先に

進める。更新された時刻で時間消費を終えたトランザクションは「動きうるトランザクション」となり，再びネットワーク上を移動する。これが待ち状態にあるトランザクションの動きを促し，シミュレーションを進行させる。シミュレータはトランザクションの各状態，即ちGATEブロックにおける待ちの状態，ADVANCEブロックにおける時間消費の状態及び動き得る状態等に対応したリスト（図5）を持ち，それら各リスト内のトランザクションに対して図6の流れ図で示される処理を行う。

```
P010   GENE   1, U, 0, P020, 2, 10
P020   SPLIT  P060, P030
P030   GATE   1, G1, F, 1, P040
P040   ADVA   1, U, 1, P050, 2, 8
P050   GOUT   G1, P090
P060   GATE   1, G2, F, 1, P070
P070   ADVA   1, U, 2, P080, 3, 5
P080   GOUT   G2, P090
P090   ASM    P100
P100   GATE   1, G3, F, 1, P110
P110   ADVA   1, U, 3, P120, 3, 9
P120   GOUT   G3, P130
P130   TERM
```

図－4 シミュレーションモデル例のRISSプログラム

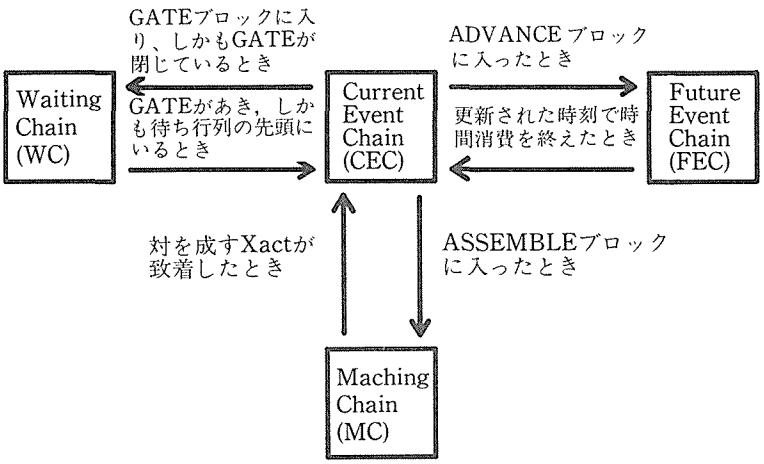
各トランザクションは次に処理を受けるべきブロック文番号等の情報を持っている（図7）。シミュレータは動きうる状態にある各トランザクションに対してその優先順及び致着順にそのブロック文番号に対応するブロック命令の処理を行い，次に処理を受けるブロック文番号を与える。

シミュレータは以上のようなシミュレーションクロック，トランザクション及びリストの管理を行うと同時に，GATEブロックの待ち行列の長さ等のデータ収集及びそれらの統計計算を行う。

4.2 シミュレーション実行制御の並列化

離散型システムシミュレーションでは，各ブロックにおける処理(サービス)がトランザクションの到着と共に開始され，他と独立に実行される。従って各ブロックの処理を最小単位とした並列化が可能である。即ち各ブロックにプロセッサを割当てる。このとき，トランザクションはこれらプロセッサで構成されるネットワーク上をシミュレーションプログラムの流れに従って移動する。

PPAではスレーブプロセッサが共有メモリを介した1次元（リング状）のネットワークを構成しており，ネットワークに沿った隣接プロセッサ間のデータ転送が共有メモリを介して高速かつ



図－5 トランザクション（Xact）の状態リストの相互関係

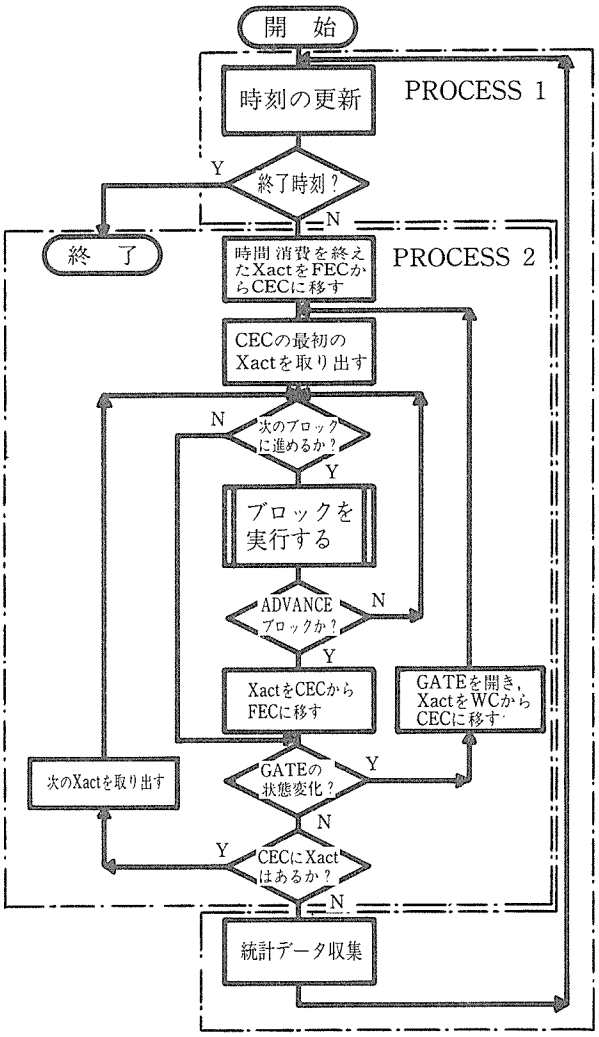
容易に行われる。そこで本稿では，分岐やサイクルを含むシミュレーションモデルの有向ネットワークをPPAの1次元の単純なネットワーク上に図8のように写像し，対応するブロック命令の処理をプロセッサに割当てる並列化を採用する。トランザクションは各共有メモリ SM_i ($i=0$, 1, …)上に作られたリスト上を移動する。各スレーブプロセッサ SLP_i ($i=0$, 1, …) は共有メモリ SM_{i-1}上のリストに入っているトランザクションに対して，割当てられたブロック文を実行する。トランザクションに対して処理を行うべきブロック文がSLP_iに割当てられていないとき，そのトランザクションをSM_iのリストに入れる。一方マスタプロセッサはシミュレーションクロックやシミュレーションモデルの全体的な状態の管理を行う。図6の上では大雑把に言って，MPがPROCESS 1を行い，各SLP_iがPROCESS 2を行う。

なお，統計計算には比較的多くの計算時間を必要とするため，それはホスト計算機上で行うこととする。

4.3 スレーブプロセッサへのブロック文の割当てのアルゴリズム

前掲の図8に示した写像による割当ては一意ではなく，任意性を持つ。割当ての良否はシミュレーション実行時のプロセッサの利用効率に影響を及ぼすが，一方割当ての計算に時間をかけることは並列計算のオーバヘッドの増大を意味する。そこで本稿では以下に述べる単純かつ実用的な割当てのアルゴリズムを用いる。

ここではスレーブプロセッサSLPの台数を一般的に m 台とし，与えられるシミュレーションモデルの各ブロック文を B_i ($i=1$, …, n)で表す。



図－6　単一計算機によるシミュレーションのゼネラルフロー

	0	15
0	トランザクションの状態	優　先　順　位
1	トランザクション番号	
2	発　生　時　刻	
3	後続トランザクション番号	
4	ブロック文番号	
5	属　性　値　1	属　性　値　2
6	属　性　値　3	属　性　値　4
7	トランザクションが動き出す時刻	
8	待ち行列番号	
9	次のブロック文番号	

図－7　トランザクションのデータ構造

通常は $n>m$ である。

STEP 1　各ブロック文の負荷と各SLPに割当てる理想負荷の計算

各ブロック文 B_i の計算負荷 W (B_i) をそのブロック命令の実行に要する時間とする。理想負荷量 w を

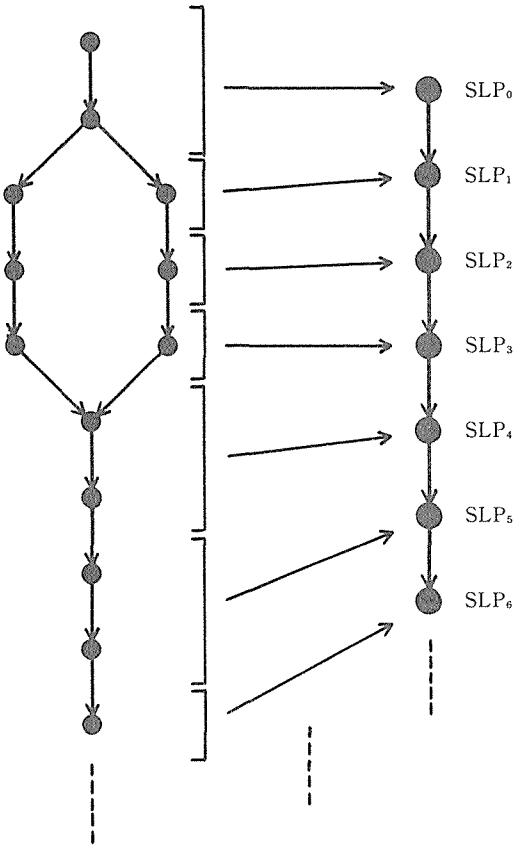
$$w=\sum_{i=1}^{n} W(B_i)/m$$

とする。

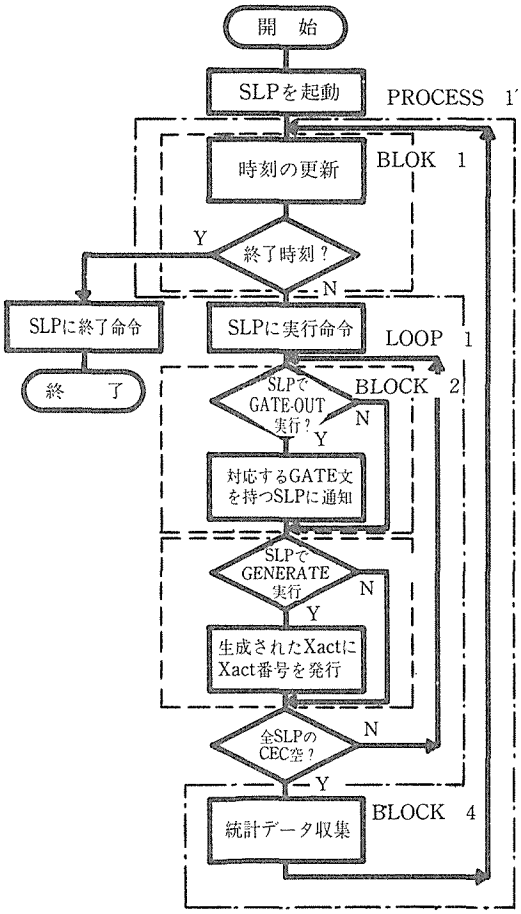
STEP 2　ブロック文のソーティング

与えられるシミュレーションプログラムの各ブロック文を頂点* としブロック文間の結合を弧* としたとき得られる有向グラフ* をGとする。Gがサイクリック* な場合，サイクル* 上の適当な弧を取り除く。このとき，Gから図8のような単純連鎖* への写像はGの階層を考えることで求めることができる。具体的には次のようにする。

(1)　与えられるシミュレーションモデルの有向グラフGの全ての弧の向きを逆にして得られる



図－8　シミュレーションモデルのネットワークと1次元プロセッサアレイとの対応(図3の例題)



図－9　マスタプロセッサMPでの処理のゼネラルフロー

＊　文献6)参照

有向グラフ G_1 の各始点，即ち TERMINATE ブロック(付録参照)より弧を逐次追跡することでサイクルを検出する。例えばブロック B_1，B_2，…，B_l がサイクルを構成するとき，弧 $a \simeq (B_l, B_1)$ をグラフより削除する。

(2) 各ブロック B_i の階層 L (B_i) を次のようにして求める。

a．各ブロックの階層 L $(B_i)=0$ とする。

b．各初点(TERMINATE ブロック)より出発し，全てのパス* について以下の計算を行う。頂点 B_i の階層 L $(B_i)=l$ で，頂点 B_j が B_i を初頂点* とする弧の終頂点* であるとき，B_j の階層 L (B_j) を新たに

L $(B_j)=\max$ $(L$ (B_j), $l+1)$

とする。

c．ブロック文を階層の大きい順に並べる。同一の階層を持つブロックは STEP 1 で求めた負荷の大きい順に並べる。

STEP 3 ブロック文の SLP への割当て

STEP 2 でソーティングされたブロック文を先頭より，それぞれ負荷の和が理想負荷 w となるようなグループ $G_0, \cdots, G_{m-1}$ に分割する。各 G_i $(i=0, \cdots, m-1)$ をそれぞれ SLP_i に割当てる。

5．並列処理方式の離散型システムシミュレータにおける処理の概要

4 章で述べた並列処理方式を用いた離散型システムシミュレータを試作中である。そこでは，前掲の図 7 のデータ構造を持つトランザクションがシミュレーション実行に伴って各共有メモリ SM_i $(i=0, \cdots, 31)$ それぞれに作られたリスト（図 5）上をプログラムの流れに従って移動する。このときマスタプロセッサ及び各スレーブプロセッサが行う処理は図 9，10 のフローチャートで示される。以下に，マスタプロセッサ，スレーブプロセッサにおける処理の概要を述べる。

5.1 マスタプロセッサにおける処理

マスタプロセッサは主にシミュレーションクロックの管理，GATE, GATE-OUT ブロック(付録参照）の状態変化に対する処理，各トランザクションの識別及び統計データのホスト計算機への転送を行う。図 9 の PROCESS $\hat{1}$ は図 6 の PROCESS 1 にほぼ等しい。図 9 の BLOCK 1 から BLOCK 4 までの各処理ブロックの機能を解説する。

BLOCK 1 シミュレーションクロックの管理

MP は図の LOOP 1 で個々の SLP の状態を監視する。BLOCK 1 では全 SLP において処理を要求するトランザクションがなくなったとき，シミュレーション時刻を更新する。なおシミュレーションモデル内に異なる優先順位を持つトランザクションが存在するときは，各優先順位毎に LOOP 1 をくり返す。

BLOOK 2 GATE, GATE-OUT ブロックの状態変化に対する処理

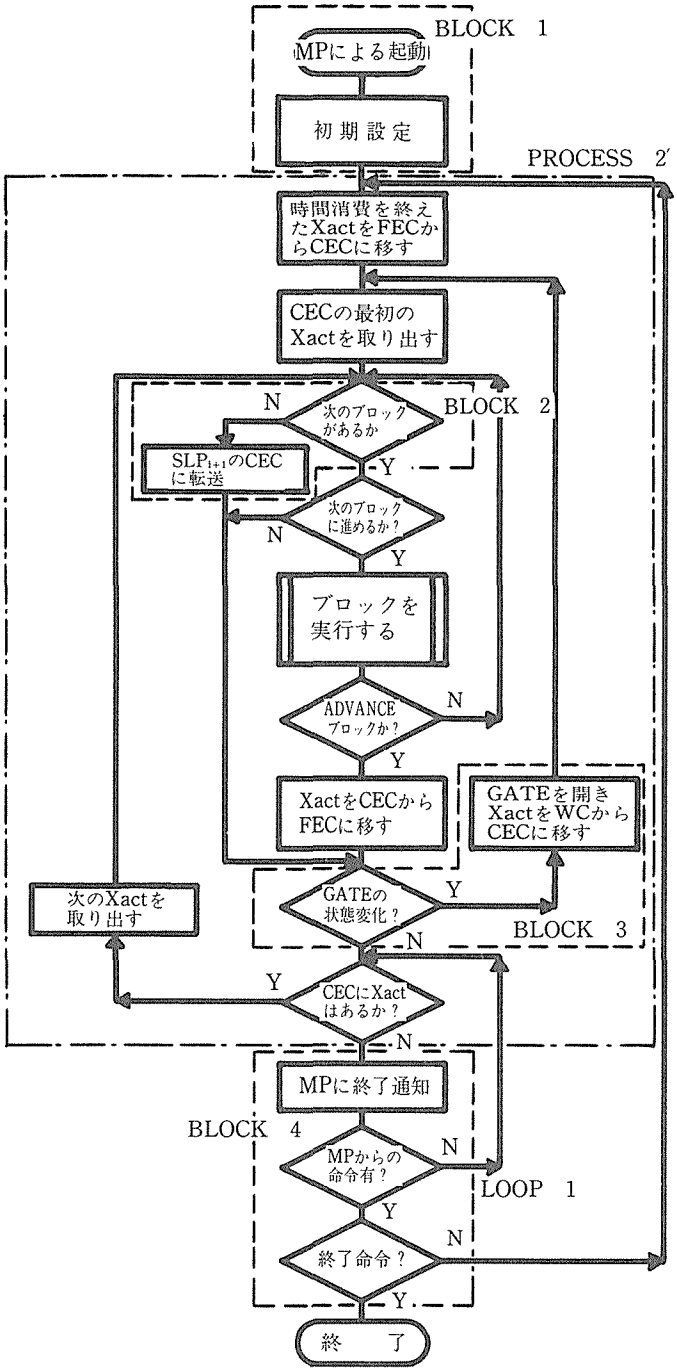
GATE, GATE-OUT ブロックは容量を持つ設備の入口，出口として対をなして使われ，通常はその間に一つ以上のブロックが介在している(図 3 参照)。GATE-OUT ブロックはトランザクションがそこを通過したときそのことを対応する GATE ブロックに知らせる働きをする。一方 GATE ブロックはゲートを開き，待ち行列があればその先頭のトランザクションを通過させる（図 10, BLOCK 3 参照)。

本シミュレータでは MP が BLOCK 2 で GATE-OUT ブロックの通過状況を対応する GATE ブロックに通知する。

BLOCK 3 トランザクションの識別

シミュレーションモデル内に滞在するトランザクションはそれぞれ固有のトランザクション番号（図7参照）を持ち，他と区別される。MPはBLOCK 3でシミュレーションモデル内で新たに発生するトランザクションに対してこの番号を発行し，それらを管理，識別する。

BLOCK 4　統計データの集収



図－10　スレーブプロセッサ SLP$_i$ での処理のゼネラルフロー

BLOCK 4では各シミュレーション時刻の終りに，その時刻における各種データを収集し，ホスト計算機に送る。

5.2 スレーブプロセッサにおける処理

各SLP_i ($i=0$, …, 31) は共有メモリSM_{i-1}内に作られたリスト上のトランザクションに対して図10の処理を他と並行して行う。図10のPROCESS $\hat{2}$ はBLOCK 2を除くと図6のPROCESS 2とほぼ等しい。以下に，図10のBLOCK 1からBLOCK 4までの各処理ブロックの機能を解説する。

BLOCK 1 MPによる起動と初期設定

各SLP_iはMPから起動をかけられると，SM_{i-1}上のリストの初期化を行う。最初のトランザクションを発生すべきGENERATEブロックを割当てられているSLPはそのGENERATEブロックを実行する。

BLOCK 2 トランザクションの転送

SLP_iは取り出したトランザクションに対して次に処理を行うべきブロック文が自分自身に割当てられていない場合，そのトランザクションをSM_iに転送し，その処理をSLP_{i+1}にまかせる。一方処理を行うべきブロック文がSLP_iに割当てられている場合には，図6の単一計算機の場合と同様の処理を行う。トランザクションに対して次に処理を行うべきブロック文が離れたスレーブプロセッサSLP_j ($j>i+1$) にあるとき，そのトランザクションはSLP_i, …, SLP_{j-1}による上述の転送をくり返し受け，SLP_jに到達する。

BLOCK 3 GATEブロックの状態変化に対する処理

図9，BLOCK 2でMPからGATE-OUTブロックの実行を通知されたとき，SLP_iは対応するGATEブロックのゲートを開き，待ち行列リストにトランザクションが入っていればその先頭の

付録 RISSのブロック命令一覧

ブロック名		機能
ADVA	advance	トランザクションを指定分布に従って滞在させる
ASM	assemble	splitしたトランザクションを集め一つにまとめる
ASGN	assign	トランザクションの属性値を指定する
CIN	check in	セクションの入口を定義する
CNTR	counter	カウンタの内容を更新する
COMP	compare	2つの指定項目を比較し分岐先を決定する
COUT	check out	セクションの出口を定義する
DBW	data block	トランザクション発生時間分布のテーブルを定義する
GATE	gate	ファシリィティの入口を定義する
GENE	generate	トランザクションを指定分布に従って発生させる
GOUT	gate out	ファシリィティの出口を定義する
GSW	gate switch	ゲートの開閉を行う
GTST	gate test	ゲートの開閉状態によってトランザクションを分岐させる
HALT	halt	シミュレーションを一時停止する
NOP	no _ operation	何もしない。分岐先変更に使用する
PORT	portion	指定確率に従って分岐させる
PRIO	priority	トランザクションの優先順位を指定する
SPLIT	split	トランザクションのコピーを行ない並行するフローを作る
SRCH	search	指定した条件を満たすトランザクションを探す
STOP	stop	シミュレーションを終了する
TERM	terminate	トランザクションを消滅させる

トランザクションを動きうるトランザクションのリストに入れる。

BLOCK 4 シミュレーション時刻の更新に対する処理

SLP_i は SM_{i-1} のリスト上に動きうるトランザクションが一時的になくなっても，それに対する処理は停止しない(図 10, LOOP 1)。MP が全 SM 上に動きうるトランザクションがないことを確認し SLP に命令を発行したとき，SLP はその処理を停止する。

6. むすび

本稿では 1 次元プロセッサアレイ方式の並列処理システムを用いて GPSS 系統の離散型システムシミュレーションを高速に行うための並列処理方式について検討した。GPSS系統の離散型システムシミュレーションでは各ブロックがそこに致着したトランザクションに対して行う処理が本質的に他と独立であることに着目し，シミュレーションモデルの各ブロックが構成するネットワークを 1 次元プロセッサアレイ上に再現し，その上を動くトランザクションに対する処理を各プロセッサで並行して行う並列処理手法を提案した。また，この手法による並列演算の前処理としてシミュレーションプログラムの各ブロック文を 1 次元アレイ状のプロセッサに割当てる割当てのアルゴリズムを提案した。

本研究は本学汎用シミュレータ施設において単一のミニ計算機上で現在既に稼動している離散型システムシミュレータ RISS のシミュレーション実行部分の機能を同施設の並列処理システム PPA 上に実現し，その高速化を計ることを具体的目標としている。このような目標のもとに現在試作中の並列処理方式による離散型システムシミュレータについて，その処理の概要を紹介した。

謝辞

本研究においてその動機付けと有益なご指導を与えて下さった故小山昭一先生，並びにその後種々のご助言を下さった島公脩教授に心から感謝致します。

参考文献

1) 中西俊男：コンピュータシミュレーション (昭 57)，近代科学社
2) 牧野圭二他：電気四学会北海道支部連合大会講演論文集，(昭 56)，p.219
3) 牧野圭二他：電子通信学会技術研究報告，82 (昭 57)，16,p.57－68
4) 小山昭一他：シミュレーション，1 (昭 56)，1，p.42－49
5) 土肥　俊：情報処理学会論文誌，25 (昭 59)，3
6) R.G.バサッカー他：グラフ理論とネットワーク/基礎と応用 (昭 45)，培風館
7) 後藤　環他：SICE 北海道支部学術講演会論文集，(昭 58)，p. 7