論　文

ポリプロセッサ・シミュレーション・システム-PPSS*

古谷　立美**

Abstract

This paper describes a poli-processor simulation system (PPSS) that enables various poli-processor system (PPS) designers easily to construct a simulator of PPS according to their requirements and to evaluate its performance and correctness. In the construction of simulator, user inputs four isolated descriptions (1; resource operation. 2; resource connection. 3; operating system. 4; application program.) corresponding to one's PPS design.

The simulator produced by the PPS is of hierachical and modular structure, and is a structured system, without any special attention given by the users. Of main features are (1) description of PPS is simple compared with general purpose simulation language SIMTRAN; (2) simulator can be modified easily because of the hierachical structure of simulator; (3) simulation level (from μ-instruction to macro-instruction) is determined freely according to user's interests; etc.. Some ideas and methodologies represented here are also useful and applicable to other simulation and system design.

1. まえがき

標準モジュールセットの考え方は，回路素子からゲート，フリップ・フロップを経てレジスタ・トランスファ・モジュールへと発展して来た．一方半導体技術は，めざましく進歩し，豊富な機能を持った CPU チップや大容量 IC メモリが安価に入手出来るようになった．そしてこのような傾向は，LSI を豊富に使って目的のシステムを組立てるという考え方を実現可能にした．この代表的なものがポリプロセッサシステム (PPS) である．しかしこのようにしてシステムを構成する場合，資源の選択，配置・結合，スケジューリング等の決定が性能に重大な影響を与えるばかりか，これを誤ると所定の動作が行えない状況も発生する．そこで予じめシミュレーションを行いそれらの検討を行うことが肝要である．過去マルチプロセッサシステムのシミュレーション・システムに関する報告は二，三[2),3)]あるが，それらはいずれも特殊なハードウェアを前提にしていたり，解析対象を一点にしぼったもので，扱えるプロセッサ数も限られていた．故に PPS を総合的に検討出来るものではなく，当然総合的に検討する系統的手法も示していない．

この論文は，いろいろな立場の PPS 設計者が，自分の要求に合った PPS のシミュレータを PMS*** レベルで容易に構成し，その性能評価と動作の正当性の確認を可能にするシミュレーションシステム (PPSS) に関して述べる．PPS の設計領域には，一般に **Fig. 1** に示すものがあり，設計者の立場もこれらのうち一部を設計する者から，全体を設計するものまで様々である．PPSS は，それらのいかなる設計者でも簡単に目的のシミュレータを構成出来，任意の精度でのシミュレーションが出来るように設計されている．またこのシステムのユーザは，何ら意識することなく，Dijkstra の階層法[5)]と Parnas のモジュラー法[6)]によりシミュ

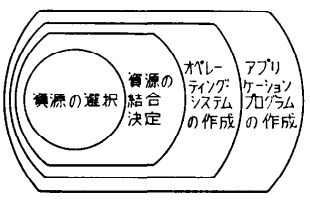


Fig. 1 Design phases of PPS.

* A poli-processor simulation systm-PPSS by Tatsumi FURUYA (Computer Division, Electrotechnical Laboratory).

** 電子技術総合研究所電子計算機部計算機方式研究室

*** C.G. Bell の記法[4)]

レータを構成していく形になっており，出来上ったシミュレータは，構造化されている．ここで示す考え方と手法は，このシステムの初版[1]の巾広い使用から導びかれたもので，PPS のシミュレーションにとって重要な多くの点を含むと共に，この他のシミュレーションや，システムの設計等に応用可能な有用な点を示している．

2. PPSS の概要

PPSS は，種々の PPS のシミュレータを PMS レベルで構成し，設計した PPS の性能測定と動作の確認を行うことを目的に設計したシミュレーション・システムである．**Fig. 2** は，シミュレーションの流れを示している．PPS のシミュレータを作ろうとする人は，Fig. 1 の 4 つの設計フェーズに対応した 4 つの記述を行うと，それぞれに対応したルーチンで処理され，4 つの分離されたサブシミュレータがFORTRAN のソースプログラムの形で生成される．このサブシミュレータ間のインタフェースは，均一化され，単純化されているため，ある設計フェーズの変更に対応した記述の変更は，他の記述と独立に行うことが出来る．またサブシミュレータは，SIMTRAN のイベント管理部を利用しているが，ユーザは，これに関する知識は不要である．4 つのサブシミュレータは，SIMTRAN のプログラムと共にコンパイルされ，リンクされて目的のシミュレータが完成し，シミュレーションが実施される．

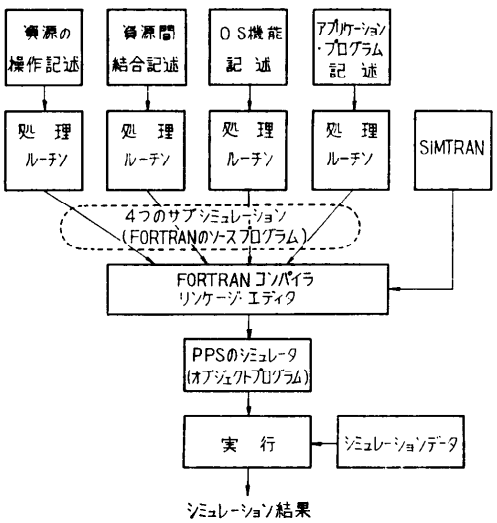


Fig. 2 Flow diagram of simulation.

PPSS は，2つのモードのシミュレーションを行うことが出来る．その1は，各プロセッサの処理内容には関知せず，システム全体の性能を求めるものである．その2は，各プロセッサの動きの命令レベルでシミュレートするもので，システムの性能の他並列処理の処理結果も求まるものである．なお，後者のモードでは，アプリケーションプログラムの記述次第でシミュレートする命令のレベルは，マイクロ命令からマクロ命令まで任意に設定出来る．

性能に関する測定データは，各資源の利用率，待ち時間，使用頻度等であり，各記述中で測定対象を指定するため，任意の項目に関して測定が出来る．また PPS では，各プロセッサの処理過程が固定の場合等に，ある資源がボトルネックとなり，システムが同期動作をする場合があるため，これに関するデータも求まるようになっている．

3. 設計の要点

この章は，PPSS の設計思想と主なアイデアを示す．

（1） PMS[4] レベルでのシステム把握

多数の LSI を使って PPS を構成する場合，資源の配置・結合は，重要な設計ポイントである．このような点を明確に評価するため，PPS を PMS レベルでとらえるようにした．

（2） 設計フェーズによるシミュレータの分割と階層化

Fig. 1 に示した各フェーズの設計者が，自分の設計しているフェーズにのみ着目してシミュレータを構成出来るようにするため，シミュレータはサブシミュレータに対応した以下に示す 3 つのレベル（資源操作部と資源結合部は同一レベル）から構成され，レベル間のインタフェースは原則として 1 つ上のレベルへのサブルーチンコールに限定する．**Fig. 3** は，その様子を示したもので，最上位は SIMTRAN のイベント

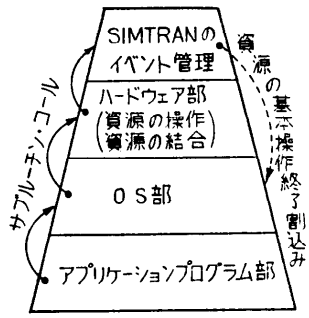


Fig. 3 Hierarchical structure of simulator.

管理部である．Fig. 3 の各レベルの概要は次のとうりである．

（i） ハードウェア部；各資源の基本操作と，資源間結合状態を示すデータベースよりなり，このレベルからのみ SIMTRAN のイベントスケジュール表にアクセス出来る．

（ii） オペレーティングシステム (OS) 部；資源のスケジュール等を主な仕事とし，アプリケーションプログラム部中で使われるプリミティブを解釈し，ハードウェア部の基本操作に起動をかける．

（iii） アプリケーションプログラム部；PPS 上で走るアプリケーションプログラムを FORTRAN 文と OS 部へのアクセスプリミティブで書いたもの．

（3） 資源間結合の可変性

PPS のシミュレーションでは，使用する資源が決定した後で，それらの結合形態をいろいろ変えて性能評価を行う場合が多い．PPSS では，このような結合形態の変更を容易に行えるように，Fig. 3 のハードウェア部を資源の操作を記述する部分と，操作中に参照する他の資源の状態信号が，どの資源のどの信号に相当するかを指定する部分に分離している．

（4） OS の機能をユーザが決定する．

PPS の OS は，システムごとに最適性能をめざして別個に設計されることが多い．そのため PPSS では OS の機能を予じめ固定することをせず，ユーザが自分の好む OS 機能を自由に組立てられる機能を提供している．これは，OS の核[8]に相当するものである．

（5） FORTRAN ベースのモジュール化

ハードウェア部と OS 部は，サブルーチン化された基本機能の集合であり，前節で述べたように 1 レベル上のサブルーチンを FORTRAN プログラム中から呼ぶ形で下位レベルの基本機能が出来ている．これは，構造化プログラミング[7]の概念をインプリメントするもので，シミュレータの構成が明確になると共に，シミュレータの部分的変更が全体に影響を与えないようにする上で重要である．

PPSS の生成するシミュレータは，FORTRAN のソースプログラムであり，システム構成にフレキシビリティを与え，プログラムのチェックも容易に行える．また PPSS 自体も FORTRAN で書かれていることは，多くの計算機システムでシミュレーションを実施することが出来る．

（6） 2つのタイプのシミュレーション

2. で簡単に触れたように，2つのタイプのシミュレーションが可能である．第1のタイプは，PPS 上で走るプログラムを命令レベルでシミュレートするもので，実際のプロセッサで実行される処理がそのままシミュレートされるため，並列処理の演算結果も求まり，並列処理のタイミングに由来するプログラムエラー等のチェックに利用出来る．第2のタイプは，各プロセッサの発生する他の資源へのアクセス頻度の統計データに基づいて，各資源の利用状況等を求めるものである．これは，各プロセッサの処理内容には関知せず，システムの大まかな性能を調べる時に便利である．

（7） アプリケーションプログラムのシミュレート法

（6）で示した第1のタイプのシミュレーションの場合，命令にもマイクロ命令，機械命令，マクロ命令等いろいろのレベルがあり，必要に応じたレベルでのシミュレーションが行える必要がある．PPSS では，ユーザがアプリケーションプログラム部を記述する際に自由に設定出来るようにしてある．PPSS では，PPS

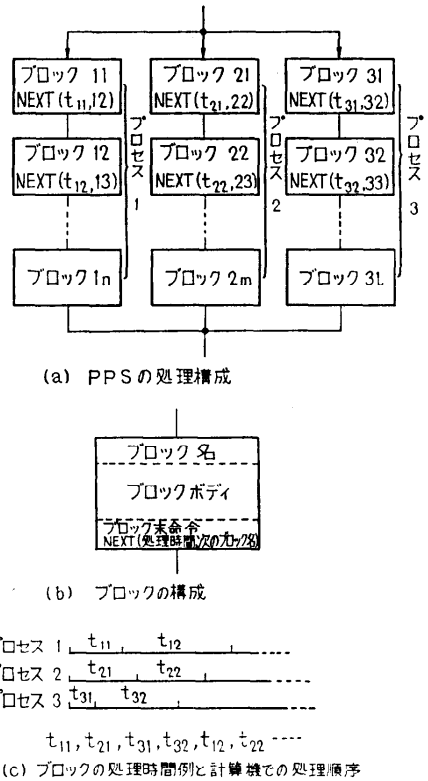
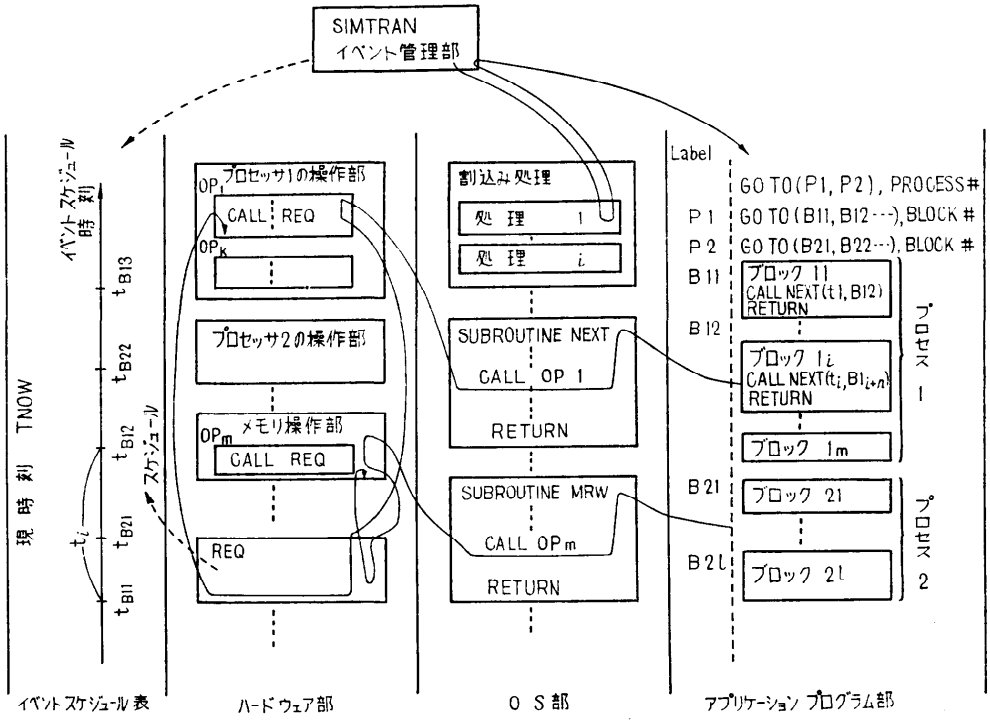


Fig. 4 Simulation method of parallel processing.

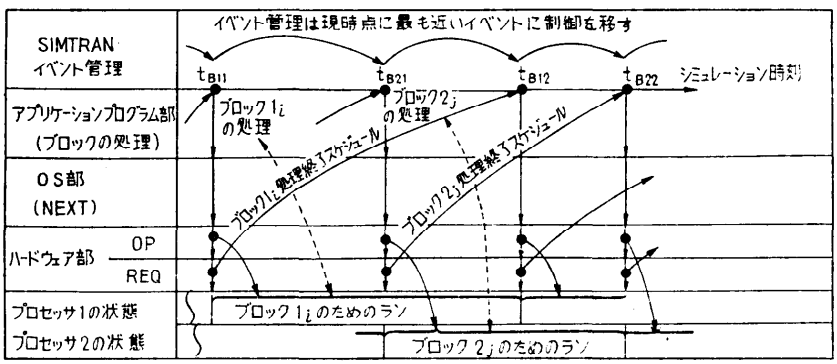
の処理をプロセスの集合ととらえ，各プロセスは，割込み不能なブロックの集合とし，シミュレーションは，ブロック単位に進める．**Fig. 4** は，その様子を示している．図の(a)は，3プロセスの例を示しており，(b)は，ブロックの構成を示している．ブロックは，ブロック名，ブロックボディ，ブロック末命令より成る．ブロックボディは，ブロックでの操作を FORTRAN 文で書くもので，マイクロ命令レベルのシミュレーションを行いたい人はマイクロ命令に相当する操作を，マクロ命令レベルのシミュレーションを行いたい人はマクロ命令に相当する操作をここで指定すればよい．ブロック末命令 (NEXT) は，ブロックの処理時間と，次に実行すべきブロック名を指定する．Fig. 4(c) は，プロセッサ数がプロセス数以上ある場合のブロックの処理時間例と，計算機の処理順序を示している．

4. PPS のシミュレート法とシステム構成

この章は，前章の設計思想を実現する PPS のシミュレート法と，シミュレーションシステムの構成を述べ



(a) Simulation process example



(b) Simulation process model.

Fig. 5 Simulation process.

る.

4.1 PPS のシミュレート法

この節は，1台の計算機で，PPS の処理をどのようにシミュレートするかを，**Fig. 5** の例で説明する．これは，2つのプロセスを2台のプロセッサで実行する単純な場合を示している．

SIMTRAN のイベントスケジュール表には，イベントの発生時刻とイベントの種類が登録され，イベント管理は，現在のイベントの処理が終ると，現在のシミュレーション時刻に最も近い時刻に起こる次のイベント発生時刻にシミュレーション時刻を進め次のイベントの処理に制御を移す．今，シミュレーション時刻が t_{B11} で，これはプロセッサ1がブロック $1i$ の処理を開始するという時刻とすると，制御は，アプリケーションプログラム部のブロック $1i$ の処理に移る．ブロック $1i$ の処理が終った時点では，プロセッサ1が t_{B11}〜t_{B12} の間に行う処理は既に終っているがシミュレーション時刻はまだ t_{B11} であるため，プロセッサ1が実際には t_i 間の処理を行っている様にシミュレートする必要がある．そこで，ブロック末命令は，OP を呼びプロセッサ1をラン状態に登録すると共に REQ 命令を呼びブロックの処理が t_{B12} に終了する旨をイベントスケジュール表に記入する．NEXT 命令は，この他，割込み処理ルーチンと共に，処理ブロックのシーケンス制御も行う．シミュレーションは以上の流れを繰り返すことで進行する．Fig. 5(a) の実線は制御の流れを示し，(b)は，それをモデル化して示したものである．

また，ブロックの処理が共有メモリへのアクセス等，他の資源の状態が関係し，処理時間を予じめ決定出来ない場合には，NEXT 命令を使わない．図のサブルーチン MRW は，メモリ・アクセス用の OS 機能で，あるブロックからこれが呼ばれると，ここから，ハードウェア部のメモリ操作部が呼ばれる．メモリ操作部では，もしメモリが空いていれば，メモリを使用中にし，読出し終了時刻をイベントスケジュール表に登録するし，メモリが使用中なら，待ち行列に入れる．待ちに入ったリクエストは，メモリで現在処理中の仕事が終った時点に，割込み処理から再度ハードウェア部のメモリ操作が呼ばれ，待ちから出される．以上がシミュレート法の基本原理である．

4.2 システム構成

この節は，Fig. 2 に示したシミュレータを構成するための4つの記述と，それらの処理部に関して述べる．

(1) ハードウェアに関する記述とその処理部

ハードウェアに関する記述には，資源の操作記述と，資源間結合記述がある．操作記述は，その例を **Fig. 6** に示すように，資源名(RN)，他の資源からアクセス可能な状態信号(GS)，基本操作(OP1, …, OPn)から成る．基本操作は，資源の動作を FORTRAN 文，システムの提供する基本命令，ユーザがハードウェア部で定義した命令により記述したものである．結合記述は，操作記述内でアクセスする他の資源の状態信号(ST1, ST2, …, ST3 と記述される）が他の資源の GS 部で示されるどの信号に相当するかを指定する．

ハードウェア記述に対する処理部は，記述からソースプログラムを生成するルーチン，システムの提供する基本命令，資源間結合を記憶するデータベースから成る．基本命令には，**Table 1** に示す様な，待ち行列操作命令，イベントスケジュール表の操作命令，統計データを得る命令，乱数発生命令等がある．故にユーザは，SIMTRAN に関する知識は全く無くても，FORTRAN と基本命令さえ知っていれば，この記述を行うことが出来るが，特に細かな制御に SIMTRAN の命令を使いたい場合には，この記述中にそれを用いても一向に差支えない．

(2) OS 部の記述とその処理部

OS 部は，OS のいろいろな機能をユーザが定義する部分で，その記述は，ハードウェア部同様 FO-

操 作 記 述	コ メ ン ト
@RN	リソースネーム.
MEM 1.	メモリ #1.
@GS	グローバルステータス.
.	ナシ.
@OP 1	オペレーション 1.
(WN, NB)	パラメータ.
IF(MS, EQ. 0) GO TO 100	もしメモリが使用中でなければ 100 へ飛べ.
ST 1(ICP)=0	プロセッサを待ちにする.
¥S(ICP+1)	プロセッサの待ち時間の統計を取る.
¥QI(2, 999, WN, NB, 0)	待ち行列に入れる.
¥E	終り.
100 MS=1	メモリを使用中にする.
¥S(1)	メモリの使用時間の統計を取る.
TIME=WN*ACTI	メモリの読出し時間を計算.
¥REQ(TIME, 1, NB, 0, 0)	メモリの読出し終了時刻をスケジュール.
¥E	終り.
@OP 2	オペレーション 2.
基本操作記述 2	
@END	終り.

Fig. 6 Description format of resource operation.

Table 1 Primitives for hardware description.

- REQ($t, i, a_1, a_2, \cdots, a_6$): 資源への操作起動命令 SIMTRAN のイベントスケジュール表に，操作終了時刻を登録.
  - t; 操作の所要時間
  - i. 操作終了時の処理ルーチン番号
  - $a_1, a_2, \cdots a_6$; ユーザの指定する属性
- QI($n, c, a_1, a_2, \cdots a_6$); 待ち行列 n へのデータの加入
  - c; データの挿入位置指定
  - $a_1, a_2, \cdots, a_6$; ユーザの指定する属性

  但し，プロセス名，現ブロック名，現時刻は自動的に属性として登録される.
- QO($n, c, a_1, a_2, \cdots, a_6$); 待ち行列 n からデータを取出す．引数は QI と同じ.
- F(n, i, d, c): 待ち行列 n の i 番目の属性値が d である待ち行列中のエレメント番号を c に求める.
- S(m); この一つ前の命令の変数に関する統計データを集める.
- E; 操作終了記号.
- 各種乱数発生命令.

注) 属性 a_1～a_6 は，6 個以下任意の個数にセット可.

RTRAN プログラム中からハードウェア部で定義した資源の操作や，OS 部でユーザが定義した機能，及び以下に述べるシステムの提供する機能を呼ぶことにより行われる.

この記述に対する処理部は，記述から FORTRAN のソースプログラムを生成するルーチンの他，ユーザの OS 機能定義を容易にするためなどの，次の 3 つの機能を有する.

（i） 資源のスケジュール機能; PPS の OS 中心機能は，資源のスケジューリングである．ここでは，それを容易にするために **Table 1** に示した機能 (REQ 命令は除く) の他，プロセスをプロセッサに割当てる仕事を容易にするための **Fig. 7** に示すデータ構造（プロセステーブル）やプロセッサ管理用待ち行列，及び，それらに対する操作命令を提供する.

（ii） データ転送管理; データ転送には，様々な形態があり，単純なものは，ハードウェアの資源結合記述部で管理できる．ここでのデータ転送管理とは，OS の介入が必要となるような，複雑なデータ転送をサポートするもので，資源の物理的結合関係を示すテーブル，資源間の転送路を決定するルーチン等から成り立っている.

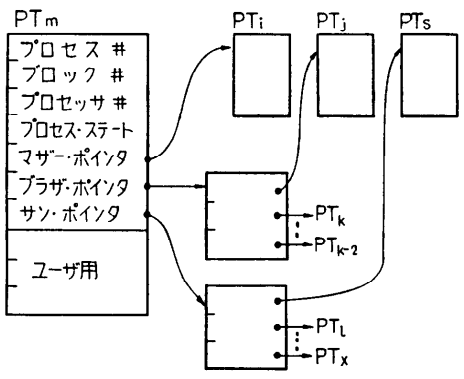


Fig. 7 Process Table (PT)

（iii） 割込み処理; この部分は，ハードウェア操作記述中の REQ 命令により起動をかけられた資源の操作が終了したとき，次にどのような処理に進むかを指定する部分である．即ち REQ 命令の 2 番目の引数 (i) がこの割込み処理ルーチンの番号を示している.

以上が OS の機能記述を容易にするために用意した機能であるが，この他に，アプリケーションプログラムで使用する並列処理用プリミティブや，このシミュレーション特有のプリミティブを解釈，実行するルーチン，即ち予じめ用意された OS 機能がある．**Table 2** は，その主なものである.

（3） アプリケーションプログラムとその処理部

アプリケーションプログラムは，一つのサブルーチンに構成される．その記述法は，次章の例を参照されたい．これに対する処理部は，ユーザの指定をもとに，ブロックを決定すること，及びブロック名と FORTRAN のラベルとの対応付け等である.

Table 2 Provided OS functions.

1. プロセスの生成・消去命令.
2. プロセス間の同期命令.
3. 資源間の通信命令
4. ブロック末命令.
5. 割込み処理.

5. シミュレーション例

Fig. 8 に示すハードウェア構成で高速フーリエ変換[10]を行う場合のアプリケーションプログラムの一部を **Fig. 9** (次頁参照) に示す．ここで共有メモリには，変換されるデータとその中間結果のみを格納し，回転因子その他は各プロセッサがローカルメモリに格納していると仮定している．変換されるデータは，1024 点の単精度で，プロセッサは当研究所で開発中のマイ

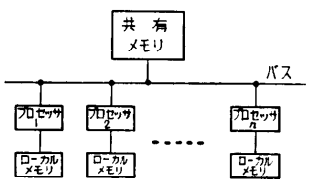


Fig. 8 Hardware structure.

```
B1   〈初 期 設 定〉
B2   FORK(2,B3)
       ⋮
     FORK(16,B3)
B3   KLEFT(PN)=IR1(L,PN)+IR2(L,PN)-1
     KRIGHT(PN)=KLEFT(PN)+DLR
     JLEFT(PN)=JLEFT(PN)+1
     JRIGHT(PN)=JLEFT(PN)+N2
     K(PN)=(IR2(L,PN)-1)*K1+1
     WK(PN)=W(K(PN))
     ICTR(PN)=ICTR(PN)+1
     NEXT(170.0,B4)
B4   W1(PN)=A(JLEFT(PN))
     MRW(4,B5)
B5   W3(PN)=A(JRIGHT(PN))
     MRW(4,B6)
B6   W2(PN)=W3(PN)*WK(PN)
     W3(PN)=W1(PN)+W2(PN)
     NEXT(440.0,B7)
B7   B(KLEFT(PN))=W3(PN)
     MRW(4,B8)
B8   W3(PN)=W1(PN)-W2(PN)
     NEXT(162.0,B9)
B9   B(KRIGHT(PN))=W3(PN)
     MRW(4,B10)
B10  IF(ICTR(PN).EQ.MAX) GO TO 10
     IR2(L,PN)=IR2(L,PN)+1
     IF(IR2(L,PN).GT.DLR) GO TO 20
     NEXT(27.5,B3)
  10 NEXT(12.5,B11)
  20 IR1(L,PN)=IR1(L,PN)+DLR2
     IR2(L,PN)=1
     NEXT(62.5,B3)
B11  JOIN(PN)
B12  DO 100 I=1,N
 100 A(I)=B(I)
     MRW(9192,B13)
B13  IF(L.EQ.10) GO TO 200
     〈新たなFORKのための設定〉
     NEXT(15.0,B2)
 200 RETURN
```

〈注 Bn;ユーザの設定するブロック〉

Fig. 9 An application program (FFT).

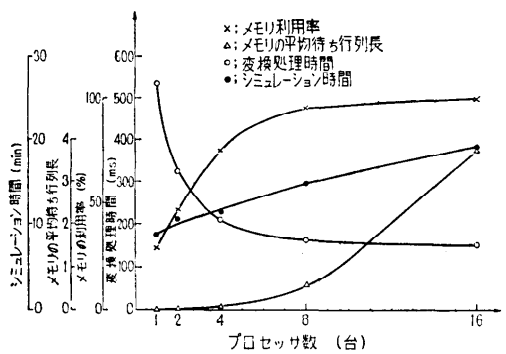


Fig. 10 Results of system performance measurement.

クロプロセッサ[9]，共有メモリのアクセス時間は 1 μs を仮定し，マイクロ命令レベルでシミュレーションを行ったものである．**Fig. 10** は，プロセッサの台数を増やしていったときのシステムの性能とシミュレーション時間を示したものである．

6. むすび

プロセッサの動作を命令レベルでシミュレートするタイプの場合，資源の操作の規模が比較的小さいときに特に取り扱い易い．そのような点からも今後広く行われるであろうポリ・マイクロプロセッサシステムの評価に最適といえよう．現在マイクロプロセッサメーカは，各プロセッサ用シミュレータを FORTRAN で提供する傾向にあり，それらと結合出来れば，より具体的レベルでのシミュレーションが可能となろう．PPSS は，現在 HITAC 8400 (256 k バイト) 上にインプリメントされており，プログラムサイズは，SIMTRAN (35 k バイト) を加えて約 75 k バイトである．この他データ構造としては，128 台までの資源を扱うための 64 k バイトの領域を用意している．シミュレーションの精度は，各プロセッサの通信リクエストをもとにシミュレーションを行うタイプについて解析モデルによる解析[11]との比較を行ったが，差は数%以下であった．

最後に，本システムに関して御検討いただいた飯塚肇室長，藤井狷介の両氏，そして PPSS 1 を御使用いただき有益な御助言をいただいた東芝総研の高橋義造氏他の方々，及び本研究の機会を与えられた黒川一夫電子計算機部長に感謝の意を表します．

参 考 文 献

1) 古谷，飯塚，藤井：ポリプロセッサシミュレーションシステム (PPSS 1)，情報処理学会創立 15 周年記念大会，(1974)
2) J. V. Levy : Computing with multiple microprocessors, Ph. D. Dissertation, Stanford Univ., (1973).
3) M. J. Knudsen : PMSL, An interactive language for system-level description and analysis of computer structures, Carnegie-Mellon Univ., (1973).
4) C. G. Bell and A. Newell : The PMS and ISP descriptive systems for computer structures, SJCC, Vol. 36, (1970).
5) E. W. Dijkstra : The structure of The Multiprogramming System, CACM 11, 5, (April 1968).
6) D. L. Parnas : A Technique for Software Module Specification with Examples, CACM, 15, 5, (May 1972).
7) E. W. Dijkstra, et al. : Structured programming, Academic Press, (1972).
8) W. A. Wulf and Roy Levin : The Hydra Operating System, Carnegie-Mellon Univ. (1975).

9) 飯塚，古谷：マイクロプロセッサアーキテクチャの一設計，電子通信学会論文誌(D)，Vol. 59-D, No. 3, (1976).
10) 吉沢：FFT とは何か，数理科学，1969年6月号.
11) 古谷：バス結合マルチプロセッサシステムの解析モデルと解析，情報処理，Vol. 17, No. 5, (1976).

(昭和51年1月24日受付)
(昭和51年6月7日再受付)