密結合マルチプロセッサシステムシミュレータの開発†

山本 登††

処理装置群と主記憶装置群とをマトリクス・スイッチで密に結合したマルチプロセッサシステムの実用化には，高集積回路と実装技術などのテクノロジの進歩，並列処理算法や並列処理言語などのソフトウェア科学の進歩のほか，スイッチ部の構成方式，キャッシュと共有メモリの不一致解消方式，システムに適した仮想記憶方式などの方式技術を確立する必要がある．なかでもマトリクス・スイッチに多量の回路が必要なことは実用化の大きな隘路である．筆者はスイッチ装置の各主記憶インタフェース対応にキャッシュを設け，実効メモリ参照速度を向上することで主記憶インタフェースの組数を削減し，必要なスイッチの回路量を減らす方式に着目している．このキャッシュ記憶系がシステムの性能に効果的に寄与するには，処理装置のキャッシュとスイッチ装置のキャッシュとの機能分担や，処理方式，構成方式などが適切でなければならない．そのためには方式設計の段階で種々の評価に利用できるシミュレータが不可欠である．この目的にかない，実現性も高いシミュレータの基本的な特性と構成を検討した結果，機械命令の解読・実行ができ，メモリ参照単位に論理回路を模擬するシミュレータをソフトウェアで実現する方式が適当なことがわかった．本論文では上記の考察をもとに開発した密結合マルチプロセッサシステムシミュレータの原理と構成を述べる．

1. はじめに

処理装置群と主記憶装置群とを一段のマトリクス・スイッチで密に接続したマルチプロセッサシステムは，スイッチ部の高速伝送能力を優先するため，数百台もの処理装置の接続は難しい．しかしどの処理装置からも主記憶装置群を等速かつ高速に参照できるという他のシステムにない利点があり，実現上の課題が解決されれば広い利用分野で受け入れられると考える．

このシステムの実用化には高集積回路とその実装技術の進歩に加え，多くの並列処理算法の開発と整備，使いやすい並列処理言語の開発などソフトウェア側で解決する課題[3]が多々ある．しかし以下の条件を満たすハードウェアアーキテクチャの確立なしに実用システムは存在しないという意味で，下記の項目は基本的な課題と言える．

■回路の量と接続点の数が膨大[1),2)]なマトリクス・スイッチ装置を優れた価格性能比で実現する方式．

■ある処理装置のキャッシュに写しのある共有メモリの領域を，他の処理装置が書き換える場合に必要な両メモリ間の不一致解消（バッファ合わせ）方式．

■この形式のシステムに適した仮想記憶方式．

なかでも実用化の最大の隘路はマトリクス・スイッチに多量の接続点と回路が必要なことにあるため，筆者はスイッチ部に多くの回路を要しないアーキテクチャの検討をすすめている．その一案として，スイッチ装置の各主記憶インタフェース（接続路と略称）対応にキャッシュ（共有キャッシュ）を設け，実効メモリ参照速度を向上することにより主記憶接続路数の増大を抑止する，多層キャッシュ方式に着目している．

各処理装置に内蔵のキャッシュ（固有キャッシュ）は共用情報の変更を他の処理装置群に迅速に伝達するためストアスルー方式を，共有キャッシュは書き込みを短時間に終了するためストアイン方式を用いる．この場合，共有キャッシュのブロック単位にどの固有キャッシュに複写されているかを示す情報（複写表示子）をもたせると，書き換えに伴うバッファ合わせが容易なこと[4]，両キャッシュを論理アドレスで参照する方が性能面で優れ，その場合アドレス変換機構はスイッチ装置に設けるのが自然[5]なことがわかっている．

図1はこれまでの検討から望ましいと考えるシステムの基本的な構成を示したもので，上述の2種のキャッシュのほか，共有キャッシュの後方，主記憶インタフェース（接続路と略称）対応に命令コードやデータを先読みする後置キャッシュ[7]の設置を考えている．

システムの性能に効果的に寄与する多層キャッシュ記憶の実現には，それらの間の機能分担，処理方式，構成方式などが適切でなければならない．そのためには上述の各方式設計とその評価に有効なシミュレータが設計の各段階で利用できる必要がある．そこで以下では，上述の目的にかない，実現性も高いシミュレー

† The Development of a Tightly Coupled Multi-Processor System Simulator by NOBORU YAMAMOTO (Department of Electrical Engineering, Faculty of Engineering, Nihon University).

†† 日本大学工学部電気工学科

タの特性と基本的な構成法について考察する．

シミュレータは構成手段の違いから，ハードウェアシミュレータとソフトウェアシミュレータがある．前者は対象を模擬する機構をハードウェアで実現するため，複数の設計項目（例えばキャッシュの置換方式，ストア方式，バッファ合わせ方式，1ブロックの容量など）の種々の処理方式を評価できるシミュレータを実現するには，それらに必要な技術的問題の解決に加え，設計・製作に多くの時間と費用を要する．そのうえ複雑な機能の実装が困難なこと，設計変更への柔軟性のなさなどハードウェア固有の特性もあり，評価項目を絞ったものを除き適用は困難である．

一方，ソフトウェアシミュレータは，計算機中に対象システムを模擬するプログラムシステムを構築し，評価に適した条件で動作させ必要な情報を収集する．この方法によっても大規模シミュレータの開発工数は莫大となろうが，実現手段がプログラミングのみですむため，論理設計に加え，専用の回路素子やLSIの開発，印刷回路基板などの実装設計が必要なハードウェアシミュレータに比べ，開発の時間や費用は少ない．そのうえ評価に本質的でない機能要素は抽象化の水準が高いまま設計できること，ハードウェアに比べ仕様や設計の変更が容易なことなどの利点もある．

なお動作単位に高速で並列に動作可能な電子回路を

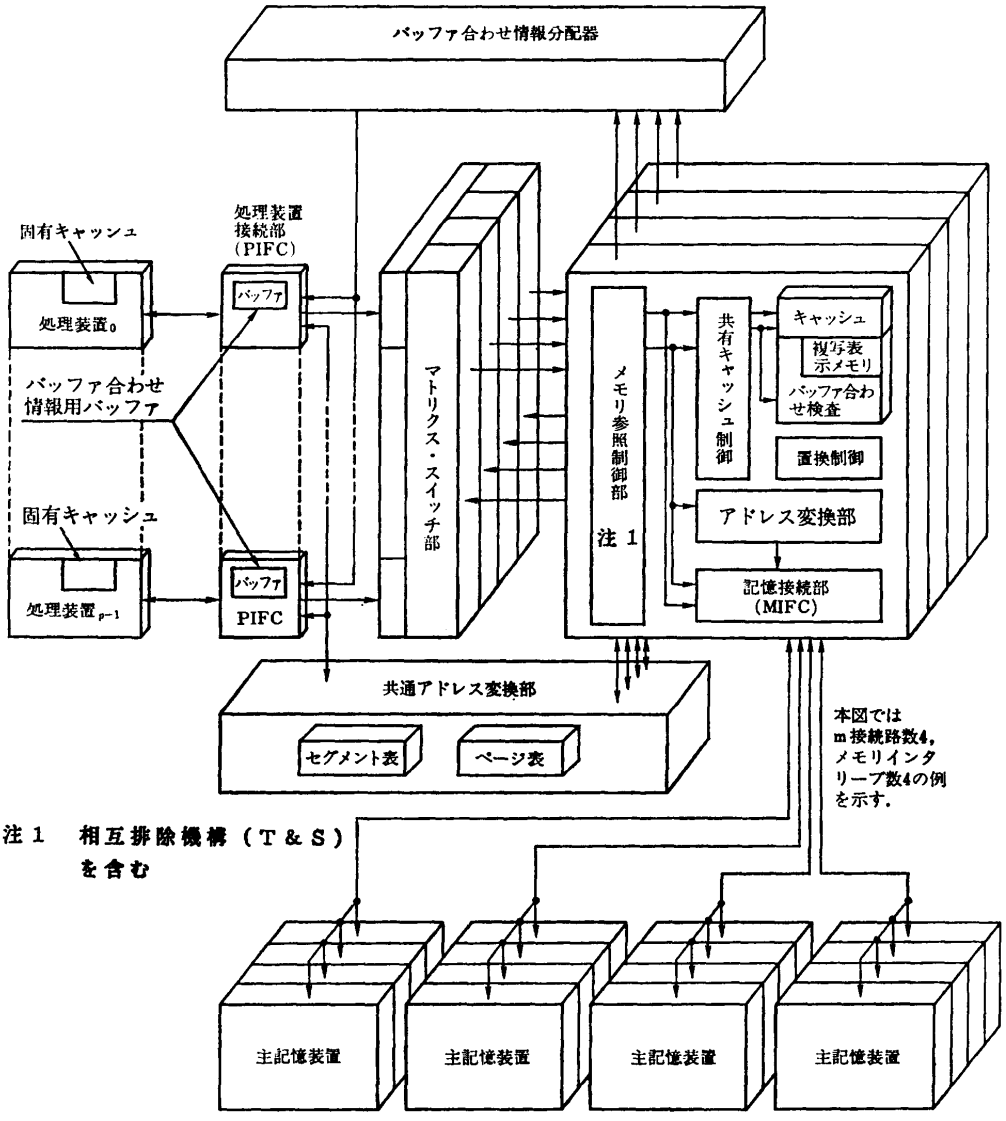

図1 密結合マルチプロセッサシステムの構成
Fig. 1 System configuration of a tightly coupled multi-processor system.

用いるハードウェアシミュレータに対し，動作単位に逐次処理を基本とする機械命令を用いるため，所要時間[6]が大きくなる．しかし，その進歩の程度に緩急の変化はあっても，電子計算機の処理速度は今後も向上の期待はでき，それは現時点で適用困難な模擬対象を可能な対象へと変えていくと考える．

以上の検討から，ソフトウェア方式のシミュレータが今後の本命と考えられ，本論文が対象とするシミュレータもソフトウェア方式により実現する．

シミュレータが模擬する動作の粗さ（粒度）は，その目的により異なる．キャッシュの処理方式や構成方式を評価する場合，ヒットの有無の検査やキャッシュへの参照，ミスヒット時の主記憶からのブロック転送などを，メモリ参照の都度把握する必要がある．それにはメモリ参照単位に模擬しなければならない．

また，キャッシュや仮想記憶では，過去のメモリ参照の履歴が後の参照動作に影響を及ぼす．このような特性の機構をもつシステムを模擬する場合，参照アドレスや読み書きの別，書き込み情報などの時系列データは，現実のジョブの実行過程で生起するものと同じものをシミュレータに与える必要がある[6]．その一般的な方法として，実システム中でジョブを実行し，その過程で生成されるメモリ参照情報を時系列的に収集し利用する方法[8),9)]がある．しかし並列に動作する複数の処理装置からなるシステムで，メモリ参照単位に動作を模擬する場合，評価対象のシステムとデータを収集するシステムとはハードウェア構成まで同じでないと，例えばメモリ参照要求の競合の程度が異なるため，評価結果の信頼性が低くなる．まして1台の処理装置で動作させたジョブから収集した情報を，複数の処理装置が並列に動作するシステムのメモリ参照情報に変換することは困難と考える．したがって筆者が検討中のシステムのように，現実に稼働するシステムがない場合，先に述べたデータ収集方式は使用できない．この問題は並列に動作するタスク群をその発生から消滅まで制御でき，しかも各タスクが実行する機械命令の解読と実行を模擬可能なシミュレータがあれば解決できる．

以上の検討を通し，密結合マルチプロセッサシステムのキャッシュ記憶の評価を目的としたシミュレータは，メモリ参照動作を論理回路の水準で模擬し，機械命令の実行機能をもたせ，プログラムをタスク単位に並列に実行できる必要があることがわかった．

本論文は上記の考察をもとに開発したシミュレータの原理と構成を述べ，このシミュレータが実行時間や記憶域の制約下で実現可能でかつ有益なことを示す．

なお論文中ではビットをb，バイトをB，ソートは整列，マージは併合と表現するが，熟語中の一部にソートまたはマージが使われることもある．

2. シミュレータの要件

2.1 対象システムの構成と動作概要

（1） 処理装置の数（p）は32程度で，専用の接続路でスイッチ装置に接続する．また主記憶装置群はインタリーブ機構をもつ独立したm組の主記憶接続路でスイッチ装置に接続する．mは共有キャッシュの存在を考慮し，pと同じかその1/2を想定している．なお処理装置群と主記憶装置群はそれぞれ処理装置接続制御部，主記憶接続制御部（MIFCと略記）を介しスイッチ装置の内部と接続される．

（2） スイッチ装置には主記憶接続路対応に記憶参照制御部（MAC）を設け，共有キャッシュのヒットの検査やヒット時の参照動作，ミスした場合の主記憶装置からの読み出しやブロック転送などを指示させる．

（3） 固有キャッシュや共有キャッシュには演算数も収容する．共有キャッシュを書き換える時，その写しをもつ固有キャッシュの該当ブロック内の書き換え対象域を，書き換えられた内容で書き直す．このためストア動作に並行して，複写表示子により該当ブロックの写しをもつ処理装置の有無を調べ，あれば番地と書き換えデータを送り，バッファ合わせを指示する[4]．

（4） 後置キャッシュの先取り方式については今後本格的に検討する予定である．以下ではこれまでに検討しシミュレータに実装した基本的な先取り方式を述べる．

（a） i番の記憶制御部（MAC (i)）で共有キャッシュのミスヒットが発生した時，主記憶接続制御部(MIFC (i))は隣接の上位アドレス域を担当するMIFC (i+1) に参照アドレスを送り，ミスヒットとなったブロックに上位隣接のブロックを後置キャッシュへ読み込むことを指示する．MIFC (i+1) は該当ブロックが後置キャッシュになければ，主記憶から読み後置キャッシュに収容する．なお演算数参照時は下位隣接のブロックも先取るため，MIFC (i−1) にも先取りを指示する．

（b） 先取ったブロックが参照されると共有キャッシュに転送されると共に，後置キャッシュからは抹消される．この場合も（a）と同様の先取りを指示する．

（c） 共有キャッシュから置換されたブロックは，後の参照に備え後置キャッシュに収容する．

（5） ミスヒットの場合は後置キャッシュや主記憶を実アドレスで参照する．これに備え，共有キャッシュのヒットの検査と並行にアドレス変換を行い，論理アドレスから実アドレスを得ておく．しかし仮想記憶機構を実装すると，シミュレータに要する仮想記憶域が使用する計算設備の限界（8 MB）を超える恐れがあり，実装していないのでその記述は省略する．

2.2 シミュレータに必要な機能

1章でシミュレータの大枠の特性を検討したが，ここでは目的の達成に必要な具体的な特性をあげる．

（1）検討対象の密結合マルチプロセッサシステムでは，バッファ合わせ方式の確認，後置キャッシュの先取りアルゴリズムの検討，主記憶接続路の組数とシステム性能の関係など，対象とするマルチプロセッサシステム特有の事項の評価のほか，例えばセットアソシアテブ方式のキャッシュの行と列の数や1ブロックの容量の大小が性能に及ぼす影響など，単一プロセッサシステムでは既知の事項であっても，マルチプロセッサの環境下では再吟味を要するものもある．したがってシミュレータは下記の要求に容易に対応できる必要がある．

- ■キャッシュの行数や列数が可変なこと．
- ■1ブロックの容量が可変なこと．
- ■キャッシュのアクセス速度が可変なこと．
- ■主記憶接続路の組数が可変なこと．
- ■バッファ合わせ方式や後置キャッシュの先取りアルゴリズムなどの処理方式の変更や確認が容易なこと．
- ■ヒット率の算出に必要なデータが得られること．

（2）論理回路レベルで模擬されるタスク群が並列に動作するので，異常の場合その原因調査を容易にする手段(メモリの参照履歴の採取など)を装備のこと．

（3）並列に動作するプログラムを模擬対象システムの主記憶へ容易にローデングする方式を設けること．

3. シミュレータの構成原理

対象システムのアーキテクチャの検討に多くの時間をかけたい．したがって安易に専用シミュレーションシステム[6]を用いるアプローチをとり，その開発に精力を費やすことは避けねばならない．そこで代表的な汎用離散型シミュレーションシステムであるGPSSについて，使用可能な命令やデータの種別，実行制御方式などを調べた．その結果，対象システムを論理回路レベルで記述できること，記述の規模に制約がなく，大きなシステムの模擬も可能なことがわかり採用を決めた．なお調査段階では必要な仮想記憶の量や実行時間の予測は困難なので，シミュレータの機能は順次拡張することにし，シミュレーションジョブの走行条件も漸次拡大することにした．

なお GPSS では配列の宣言や対象への作用の記述を『ブロック』と言うが，以下では一般の計算機言語にならい，『文』あるいは『命令』と称する．

3.1 GPSS によるモデルの記述の原理

（1） 一般的な記述の原理

対象となるモデルは設備などの定義の記述と，それらへの作用条件と作用内容を記述した命令系列で表される．前者は通常のプログラムの変数や定数の宣言に相当し，後者は手続きの記述に相当する．シミュレーションの実行過程は，前記した設備への作用の記述をトランザクション（以後 XAC と略記）が逐次たどっていくことにほかならない．これらはレジスタ，フリップフロップ，演算器などの論理回路要素の記述と，それらへの作用条件を定義したゲート回路群からなる論理回路において，信号の入力によりある論理回路の出力が活性化され，その出力が別の論理回路（群）の入力となって出力（群）を活性化していく状況を表している．

（2） 並列性の記述

GPSS で並列動作を記述するには，まず並行動作の数だけ XAC を生成する必要がある．その方法として，発生時刻や発生間隔などを指定可能な generate 命令と，任意の XAC を任意の数だけ複製する split 命令が使用できる．GPSS は生成した XAC を指定されたそれぞれの並列処理記述部へ分岐させ，各 XAC が命令シーケンスを実行するのを，時刻（クロック）の経過順に制御することで並列性を実現する．

なお独立に動作する論理ユニットの場合はgenerate命令によりユニット単位に XAC を生成し，キャッシュのヒット検査を実装された行数分並列に行う場合のように，一連の動作の一部だけ並列に動作させ他は一つの XAC で動作させる場合，split 命令で XAC を複製し，assemble 命令で単一の XAC に統合する．

（3） モデルの記述要素

GPSS のモデルの記述要素と密結合マルチプロセッサシステムの各構成要素との対応を**表1**に示す．

（a） 設備の記述

情報を記憶する設備にはXAC間で共通に参照可能な『savevalue』と，それを二次元の配列に構造化した『matrix savevalue (M配列と略記)』，各XACに付随し当該XACだけが参照できる『パラメータ』と呼ぶ記憶器などがある．情報を記憶しない設備には『Facility』と呼ぶ排他利用の設備のほか，『Pool』と呼ぶ容量を定義可能な蓄積器，XAC間でオン／オフの制御とその変化を検出可能な『論理スイッチ』などがある．なおM配列には整数型と実数型があるが，本シミュレータでは命令コードも演算数も整数として扱う関係で整数型M配列のみを用いる．このため以下でM配列という場合は整数型のM配列を指す．

（b） 手続きの記述

手続きの記述には設備の占有や返却を宣言する命令，論理スイッチをオンまたはオフする命令，記憶場所との情報の転送のための命令などがある．また，設備の使用状況，スイッチのオン・オフ状態，記憶場所の内容，待ち行列の数などを検査し，XACの移動を制御する状態判定命令もある．

（c） 時間遅延要素の記述

メモリの参照や種々の演算，情報の転送などで回路が動作するには時間がかかる．GPSSではこの記述のためadvance命令（以後遅延命令と略称）がある．シミュレータでは時間の経過あるいは遅延の記述にこの命令を用いる．

3.2 GPSSの記述要素と回路機能素子との対応

ハードウェアの基本的な機能回路を以下に示す．

- ●主記憶やキャッシュなど多量の情報の記憶回路．
- ●レジスタ，ラッチ，フリップフロップ，カウンタなどの順序回路．
- ●信号群に対し論理積，論理和，否定，排他的論理和などの論理操作をするゲート回路素子．
- ●演算回路，大小比較回路，一致検出回路，デコーダ，エンコーダ，セレクタ／マルチプレクサなどまとまった論理機能をもつゲート回路の集合．
- ●アドレスやデータなどの情報転送バス．
- ●信号波形の整形あるいは増幅回路．
- ●機器間インタフェース信号用回路とケーブル．

以下では前述のGPSSの記述要素を用いて上述の回路素子の機能を記述する方法を述べる．

（1）セットアソシアテブ方式の固有キャッシュ，共有キャッシュ，後置キャッシュには，それぞれ処理装置あるいは主記憶接続路単位にM配列を割り当てる．その行指標にはキャッシュの行番を，列指標にはセット番号を対応させる．

（2）主記憶装置の容量は下記の項目の積で表される．

表1 GPSSの記述要素（エンティティ）と回路機能要素との対応
Table 1 The entity being used to describe the model by GPSS.

種 別	GPSSのエンティティ	機能の概要	回路記述の利用例
設備の記述	facility	排他的に利用可能な設備	排他的利用の制御部の宣言
	matrix savevalue	共用可能な配列状の記憶装置	レジスタ，キャッシュ，主記憶の宣言
	savevalue	共用可能な記憶装置	シミュレーション条件設定用レジスタ
	parameter	トランザクションに付属したレジスタ	特定トランザクション専用のレジスタ
	pool	容量を指定可能な共用設備	使用せず
	queue	待ち行列	待ち合わせ数の計測用回路の記述に利用
	logic switch	オン／オフの変化をするスイッチ信号，任意のトランザクションが検出可能	他の回路と授受する信号中，受信により相手が動作を再開する信号の記述に利用
作用の記述	seize / release	排他的に利用する設備の取得と解放	排他的利用の制御部の動作開始と終了
	enter / leave	pool への到着と退出	使用せず
	assign	parameter への式の計算結果の代入	演算とその結果の専用レジスタへの転送
	msavevalue	matrix savevalue への式の計算結果の代入	演算とその結果のレジスタへの書き込み
	savevalue	savevalue への式の計算結果の代入	使用せず
	Logic s / Logic r	スイッチ信号のオン／オフ	信号の送出開始と停止回路の記述
	advance	時間遅延	回路動作の所要時間や伝送時間の記述
	split / assemble	XAC の複製と集合	回路の並行動作の開始準備
	gate ls / gate lr	論理スイッチの待機命令	待ち合わせ回路の記述
	test	条件判定命令	条件判定用組合わせ回路
	loop	ループの制御命令	繰り返し動作終了判定回路
	transfer	条件判定分岐命令，無条件分岐命令，サブルーチンへの移行と復帰命令	サブルーチン移行・復帰命令としてのほか，無条件分岐命令として利用

（a）同じ主記憶接続路上にある記憶装置の数.

（b）一つの装置の1回の動作で参照される量（語）を単位に数えた装置当たりの語数.

（c）主記憶接続路の数.

三次元のM配列が許されれば主記憶を一つのM配列で表現できるが，GPSS では二次元までのM配列しか許さないので，（b）項をM配列の行に対応させ，（c）項をM配列の列数に対応させたM配列を（a）項で示す主記憶装置の数だけ設けることで記述する.

（3）レジスタやカウンタが記憶する情報のうち，当該論理ユニットで作業用記憶として用いる情報は『パラメータ』に，他はM配列に収容する.

（4）機器間あるいは異なる論理ユニット間で授受する信号はすべての処理装置または主記憶接続路に共通な1組のM配列に収容する．その列指標には処理装置や主記憶接続路の番号を対応させ，行要素にはアドレスやデータ，動作指令，処理装置の機番などの信号を割り当てる.

また他の論理ユニットがその信号を待機し，その受信により何らかの動作を開始する信号（以下ではストローブまたはタグ信号と略称）には論理スイッチを用い，そのオン／オフの検出には GATE LS 命令／GATE LR 命令を用いる.

（5）信号群に対し論理積，論理和，否定，排他的論理和などの論理操作をするゲート回路素子は検査命令（TEST）で表す.

（6）演算（回路）は，加減算については表1の代入命令がもつ演算機能を，複雑な演算式は利用者が定義する関数を演算数とした代入命令を用いる．また大小比較や一致検出回路などは検査命令に付属した演算機能を用いて記述するほか，デコーダ，エンコーダなども検査命令を組み合わせて表現する.

（7）単なる信号波形の整形あるいは増幅回路は，回路による遅延時間のみを遅延命令で表現する.

（8）回路や伝送路による信号の遅延，メモリのアクセス時間，サイクル時間などは遅延命令で記述する.

3.3 シミュレータの論理ユニット分割

模擬対象の密結合マルチプロセッサシステムは，処理装置群，スイッチ装置，主記憶装置群などの構成要素からなる．これらの装置はそれぞれ機能的にまとまった複数の論理ユニットから構成される．対象を忠実に模擬するため，シミュレータは模擬対象システムと同じ論理ユニットに分割し，各論理ユニット単位に生成された XAC が，下記の①項と②項を繰り返し実行する方式とした.

①他の論理ユニットからの処理関始要求信号の待機.

②他のユニットと信号を授受し課された機能を実行.

4. シミュレータの構成

用いる計算機の関係からシミュレータには IBM 社のシステム 370 の命令セットの部分集合を実装した．また本シミュレータはメモリ参照動作の解析に重点を置くため，これに関する回路動作は厳密に記述するが，記述量の削減と実行時間の短縮のため，並列動作単位に分割されたタスク群が複数の処理装置で並行に

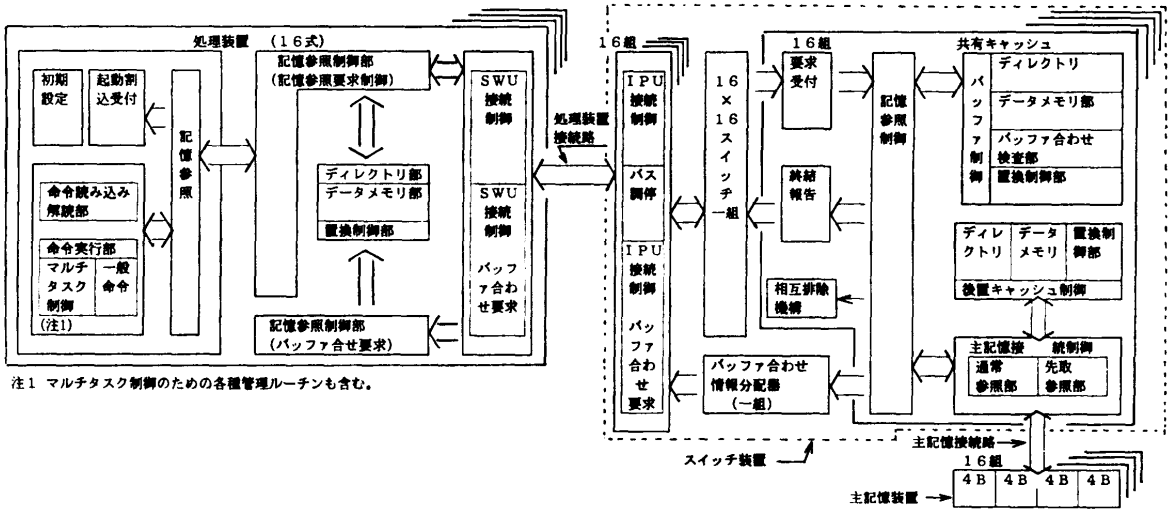

図 2 シミュレータの構成
Fig. 2 Software system structure of the simulator.

動作する環境が実現される範囲で，命令実行部の記述を簡素化した．このため実装する命令も当面必要なものに絞り，他は必要に応じ追加することにした．同じ理由で先行制御やパイプライン処理機構も実装しない．

表 2 論理ユニットの機能の概要とその記述規模
Table 2 The outline of the function and size of each logical unit.

論理ユニットの名称	機 能 概 要	規 模
初期設定	管理表類を初期化し，プログラムをローデング後ジョブを始動する．	60
起動割込み受付	割込みを検出すると割込み連絡域から TCB ポインタを得，TCB から汎用レジスタをロードし，起動（再起動）アドレスへ分岐する．	90
命令読み込み解読	命令の読み出しと解読を行う．	350
命令実行		
一般命令	解読された命令を実行する．	220
マルチタスク命令	マルチプロセッサ用マルチタスク制御命令を実行する．	1,400
記憶参照	命令の読み出しや演算数の参照のため記憶参照制御部と交信する．	90
記憶参照		
記憶参照要求制御	命令実行部からのメモリ参照要求を処理する制御部．	350
バッファ合わせ要求	バッファ合わせのため固有キャッシュを参照し更新データで書き直す．	130
固有キャッシュ		
ディレクトリ	固有キャッシュのディレクトリ情報を記憶する．	200
データメモリ	固有キャッシュのデータメモリ部（標準構成は 4 行，64 列）．	160
置換制御	新しいブロックを収容するため既使用のブロックを空ける．	90
swu 接続制御	スイッチ装置との交信を制御する接続用論理部．	250
ipu 接続制御	処理装置との交信動作を制御する接続用論理部．	630
バス調停	処理装置とスイッチ装置間の接続路の使用要求とその解除を制御．	80
16×16 スイッチ	ipu 接続制御部と記憶参照制御部を接続するマトリクス・スイッチ．	160
記憶参照制御		
要求受付	メモリ参照要求を FIFO 方式で受け付ける制御部．	200
終結報告	メモリ参照動作の結果を報告する終了処理部．	150
記憶参照制御	スイッチ装置中の中核となる制御部で，共有キャッシュのヒットの検査とヒット時の参照，ミス時の主記憶からの読み出しを制御する．	440
相互排除	複数の処理装置が同時に参照できない情報の排他的制御を行う．	60
共有キャッシュ		
制御部	共有キャッシュの下記の構成要素を制御する監視部．	270
ディレクトリ	共有キャッシュのヒットの有無を検査する．	180
データメモリ	共有キャッシュのデータメモリ部（標準構成は 16 行，256 列）．	200
置換制御	新しいブロックを収容するため既存のブロックを空ける．	100
バッファ合わせ検査	対象ブロックの写しをもつ処理装置の有無を検査する．	170
バッファ合わせ情報分配	バッファ合わせ情報を必要な ipu 接続制御部に伝送する．	90
後置キャッシュ		
キャッシュ制御	後置キャッシュの下記の構成要素を制御する監視部．	170
ディレクトリ	後置キャッシュのヒットの有無を検査する．	100
置換制御	新しいブロックを収容するため既存のブロックを空ける．	80
データメモリ	データメモリ部（標準構成は 8 行，256 列）．	120
主記憶接続制御		
通常参照部	ブロック単位の読み出しや書き込み動作のため主記憶と交信を行う．	290
先取り制御	後置キャッシュに先取るため，後置キャッシュや主記憶と交信する．	200
主記憶装置	スイッチ装置の指令により記憶回路部の読み出しや書き込みをする．	120

注） 規模欄は実行文の総計で宣言文は含まれてない．

4.1 シミュレータの構成概要

図2に GPSS で記述した処理装置，スイッチ装置，主記憶装置の主要な論理ユニットの分割とその相互関係を，**表2**に各要素の機能の概要を示した．

（1）記憶参照制御部，SWU 接続制御部，IPU 接続制御部はそれぞれ非同期に発生するメモリ参照要求とバッファ合わせ要求を処理しなければならない．そのためにはこれらの要求信号を模擬する二つの論理スイッチの変化を遅滞なく検出する必要がある．しかし GPSS には，指定した複数の論理スイッチの任意の一つが変化（オンからオフまたはオフからオン）するまで待機状態を持続可能な命令はない．そこでメモリ参照要求を模擬した論理スイッチの変化を検出し，メモリ参照に関わる処理をする部分と，バッファ合わせ要求を模擬した論理スイッチの変化を検出し，バッファ合わせに関わる処理をする部分に分けて記述した．

ただし両者は同じ回路資源を用いるため同時に動作することは許されない．そこでこれらの制御部を同じ『Facility』とみなし，使用に先立ち seize 命令で占有し，使用後 release 命令で解除することで排他的な動作機構を実現した．

MIFC においても，共有キャッシュがミスヒットの時，後置キャッシュまたは主記憶装置から読み出してブロック転送する論理部と，隣接の MIFC から先取り要求を受ける論理部の間に同様の状況があり，この場合も二つの論理部に分けて記述した．

（2）処理装置と主記憶接続路の接続数は共に 16 で，各主記憶接続路には1語 32 b の主記憶装置を4台接続し，4ウェイのインタリーブ動作をする．

なお1章に述べたように，スイッチ部の回路規模を実現可能な範囲に保つ施策の一案として，キャッシュの積極的な導入により主記憶接続路の数を削減する方式を考えている．この方式の良否の判定には主記憶接続路の数とシステム性能の関係を十分に評価する必要がある．このためシミュレータは主記憶接続路の数が 2，4，8 の場合にも動作するよう設計した．

（3）処理装置の命令実行部にデータバス幅 32 b のプロセッサを想定した関係で，メモリ参照に関係するデータバスの幅はすべて 32 b とした．このため1ブロックの容量が 16 B のキャッシュへのブロック転送は 4 B ずつ4回に分けてなされる．なお GPSS の代入系の命令は 32 b までの情報を扱えるためこの動作は無理なく記述できる．また共有キャッシュと後置キャッシュの1ブロックの容量の標準値は固有キャッシュと同じく 16 B だが，32 B と 64 B に拡張して動作させることもできる．

（4）16 組の IPU 接続制御部から発出されるメモリ参照要求は，要求受付部で到着時刻順に記憶され，FIFO 方式で受け付けられる．

4.2 相互排除機構

例えば後述のマルチタスク制御機構では**表3**に示す管理情報群を共用可能な記憶装置に設ける．ある処理装置がこれらの情報を読み，その内容を書き換えることがある．この場合，書き換えが終わるまで他の処理装置の参照を禁ずる相互排除機構が必要となる．

システム 370 では対象情報が参照可能なことを，T &S 命令を繰り返すことにより検知するが，検討対象のシステムでは命令を繰り返す方式は避けたい．しかし一般にこの待機時間は短いことが推定されるので，割込みによる検知は適当でない．

そこで対象が使用中なら待機し，参照可能になったら再開する相互排除機構をハードウェアで実現することにした．その設置場所は参照要求の集中を避けるため主記憶接続路単位に分散させることにし，後述の待ちリストを含め MAC に実装した（図2参照）．

相互排除の対象となる情報には一意な番号を割り当てる．T & S は相互排除資源の占有に，T & R (test

表3 マルチタスク制御用各種資源管理表
Table 3 Control tables for multi-task control.

名　称	機　能
タスク管理表	システムに登録されているタスクの状態(起動待ち，事象待機中，実行中）を管理する．
タスク待ち行列表	事象の発生を待つタスクを発生順に表形式で管理するもので，種別ごと（タスク終了待ち，指令発行待ち，応答発行待ち）に設けられる．
待機逆リスト	事象を待っているタスク群を事象別（タスク終了待ち，指令発行待ち，応答発行待ち）に表形式でまとめた管理表である．あるタスクである事象が発生した時，その事象を待機しているタスクを迅速に検索するため用いる．
TCB 領域管理表	TCB 用の主記憶領域を管理する．
待ち情報領域管理表	主記憶にある待機情報領域を管理する．
処理装置管理表	処理装置が動作中か遊休状態かを表示するほか動作中の場合は実行中のタスク番号を表示する．
処理装置待ち行列	処理装置が遊休状態となるのを待っているタスクを登録する．

& reset) はその解除に用いるが，対象の指定にこの番号を用いる．なお T & S で要求した資源を他の処理装置が使用中の場合，番号単位に設けた待ちリストに追加される．ある処理装置が占有を解除する時，その解放を待っている処理装置を先入れ先出し方式で選び，中断している T & S 操作を再開する．表3の管理情報は相互排除対象の一つのため，これらを参照する表4のタスク制御命令は，その開始に当たり T & S で占有権を獲得し，終了点では T & R で占有権を返却している．

4.3 処理装置接続路の調停

処理装置とスイッチ装置間の接続路は，処理装置側からはアドレスや書き込み情報の転送に，スイッチ装置側からは読み出し情報の転送に用いられるが，バッファ合わせのためアドレスや書き込み情報を処理装置へ転送する場合も用いられる．しかしメモリ参照とバッファ合わせは非同期に発生するため，メモリ参照動作が終了するまで接続路を占有させると下記の不具合が生ずる．すなわちメモリ参照の処理が終了しない間は接続路が占有状態にあるため，この間にバッファ合わせ要求が発生しても必要な情報を処理装置に転送できない．この状態はバッファ合わせ用緩衝器の設置で緩和されるが，頻発すればこれも満杯となり，MAC 部が新しいメモリ参照要求を受け付け不能な閉塞状態に陥る．この事態を避けるため，接続路を使用する時はその都度占有使用を要求し，許可された後に使用し，使用し終わると占有使用の解除を報告することを義務づけ，その調停回路を IPU 接続制御部に設けた．

4.4 命令の読み出しと解読・実行

（1）シミュレータにはロード，ストア，条件分岐，ロードアドレス，加減算，比較などの固定小数点演算命令を中心に，サブルーチン分岐やシフト命令，後述のマルチタスク制御命令群などを実装した．また汎用レジスタや条件コードレジスタは処理装置番号を列要素指標としたM配列に収容した．

（2）命令の読み出しとその解読

命令読み出し・解読部は，命令レジスタ，プログラム状態語 (PSW)，命令先取りレジスタなどの構成要素を用いて命令の読み出しと解読を行い，命令語の各フィールドを操作コードレジスタ，R_1, R_2, B_2, X_2, D_2 などのレジスタへ設定し命令実行部に渡す．

すなわち PSW の内容をアドレスバスに出力し，図2のメモリ参照制御部に命令語の読み出しを要求する．読み出し終了信号を受けると命令レジスタにラッチし，その解読を始め，命令の形式に応じた命令語のサブフィールドを操作コードレジスタ，R_1, R_2, B_2, X_2, D_2 レジスタなどへ設定する．なおさらに命令語の読み出しが必要な場合，先のアドレスに4を加え，再度，命令の読み出しを行う．命令の種別に応じ必要な命令語のフィールドが読み出されたら，命令コードに対応した処理ルーチンに分岐する．なお読み出しは 4B 単位になされるが，語長が 2B とか 6B の命令があるため 2B 余ることがある．この余りは先取りレジスタに保存し次の命令語の一部として利用する．

表 4 マルチタスク制御命令
Table 4 Multi-task control instructions.

名 称	機 能
タスク生成	子タスクを生成する場合，起動するタスクの番号，実行開始アドレス，実行時パラメータへのポインタなどを用意し，この命令を発する．
待機後起動タスク生成	指定した事象が発生した後実行を開始するタスクを生成する．タスク生成命令に必要な情報のほか，待機情報（待機する事象の種別と待機対象のタスクの番号群）を用意し，この命令を発する．
タスク終結	タスクの終了をシステムに通知する．このタスクの終了を待っているタスクがあれば再開される．
待 機	事象の発生を待って休止することをシステムに通知する．前記の待機情報を主記憶に用意し，この命令を発する．
指 令	主記憶上の指令内容を相手タスクに通知する．指令命令の発行を待っているタスクは再開される．
応 答	主記憶上の応答内容を相手タスクに通知する．応答命令の発行を待っているタスクは再開される．
ジョブ終了待ち	指定したジョブの終了を待ち休止する．
ジョブ終結	ジョブの終了をシステムに連絡する．ジョブ終了待ち命令で待機しているタスクがあれば再開する．

（3）命令の実行

実効アドレスの計算，演算数の読み出しのためのメモリ参照要求の発出と演算数の取り込み，条件コードレジスタへの条件コードの設定，分岐命令の PSW の更新処理など，複数の命令で共通に使用する機能はサブルーチンまたは共用ルーチンで記述し，加減算，シフト，分岐条件の判定など命令により異なる機能は命令単位に記述した．なおシミュレータには浮動小数点

演算や十進演算命令は実装していない．実装する場合，原始プログラム中で実定数または十進定数と定義された演算数も，シミュレータでは主記憶やキャッシュを模擬する整数型のM配列に格納されているため，その扱いに注意を要する．

4.5 マルチタスク制御

4.5.1 前提条件

（1）一つのジョブに属するタスク群の並列実行に焦点を絞り，マルチジョブ機能は最小限に止める．

（2）タスクは任意の処理装置で実行される．また事象の発生を待機して休止中のタスクを再開する場合も休止中の任意の処理装置に割り当てられる．

（3）タスクが生成する子タスクの数は，資源の制約(後述のTCBの数など)による場合を除き制限を設けない．子タスクも同様に孫タスクを生成できる．

（4）任意のタスク間で情報の授受ができる．連絡の主体となるタスクは『指令』を発し，客体のタスクは『応答』で答える．なお指令または応答に先立ち，連絡する情報を主記憶に書き込む．

（5）タスクの消滅は自身からのみ可能とし，他のタスクを強制的に消滅させる機能はもたせない．なお上述の指令・応答による連絡方式を用い，両タスク間で合意した消滅制御は可能である．

（6）事象の発生を待機して休止中のタスクは当該事象の発生により再開させられる．また指定の事象が発生するまで始動しないタスクを前もって生成できる．

（7）通常の処理装置では，各命令の実行終了後割込みの有無を検査し，あれば割込み処理シーケンスを，なければ次の命令の読み出しを始める．シミュレータにこの機能を組み込むと，シミュレーション時間が膨大になることがわかり（4.8節参照）その組込みは断念した．したがって処理装置が動作している間は割込みの検出ができない．このため低位のタスクを強制的に停止し，高位のタスクに処理装置を割り付けるタスクの優先処理は実現できない．したがって生成されたタスクの起動や，待機していた事象が発生し再開可能となったタスクの再起動は遊休状態の処理装置があれば即座になされるが，全処理装置が動作中の時は待たされる．

4.5.2 マルチタスクの実現方式

（1） 構成要素

（a）タスクは主記憶域にタスク固有の制御情報を記憶したタスク制御ブロック（TCB）をもつ．ここには起動・再開番地や汎用レジスタの写しなどタスクの実行制御に必要な情報が格納される．なお応用プログラムにはTCB領域の参照は許さない．

（b）処理装置などの各種資源や待ち行列の管理のため表3の資源管理表群を，タスク制御用に表4の機械命令群を設けた．なお資源管理表群は任意の処理装置の参照を許すため，共有可能な主記憶域に設けるのが自然である．しかしこれらは表4のタスク制御命令でのみ参照され一般の命令では参照しないこと，シミュレータの目的の充足に必要な『メモリ参照要求が競合する環境』は，並列に動作するタスク群で実現することにし，資源管理表群は主記憶とは別のM配列に記憶することにした．

（2） 起動割込み連絡域

起動割込み連絡域は主記憶域にあり，割込みを発生する処理装置が割込みを受ける処理装置にタスクに関する情報を中継する場所として用いる．その場所は両処理装置の番号で一意に決まる方式をとるため，p台の処理装置からなるシステムではp^2組（A_{11}，A_{12}～A_{pp}）の要素を必要とする．

処理装置（Pi）上で動作中のタスクが，あるタスクを起動する場合，まずTCBを動的に確保し，起動するタスクに関する情報を書き込む．次に遊休状態の処理装置（Pj）を探し，処理装置番号（iとj）できまる割込み連絡域（A(i,j)）にTCBへのポインタを書き，j番の処理装置に割込みをかける．起動の指示を受けた処理装置は，相手と自分の番号から割込み連絡域の番地を求め，そこから得たポインタによりTCB中にあるタスクの起動に必要な情報を得る．

（3） タスク起動割込み

起動（再開）を指示する処理装置は割込み要求信号を模擬する論理スイッチをオンにする．図2の起動割込み受付部で処理装置単位に生成され，この論理スイッチのオンを待っているXACがこれを検出すると，割込み連絡域からTCBポインタを得，TCB中の汎用レジスタ退避域の内容を汎用レジスタに復元し，TCB中の起動／再開アドレス部が示す場所へ分岐する．なお実行時パラメータなどの連絡情報は，そこへのポインタを当該タスクのTCBの1番の汎用レジスタの退避域に書き間接的に渡される．

（4） タスク間の連絡

受信側のタスクが連絡を待ってすでに休止状態の場合，相手タスクの指令（または応答）命令の実行過程で再開の処置がとられる．一方，受信側のタスクがま

だ動作中に相手タスクが指令（または応答）命令を発した場合，指令（または応答）命令が発せられたことをタスク管理表の該当タスク欄に記憶する．後に受信側のタスクが待機命令を発した時この欄を調べ，すでに連絡を受けたことを認知すると休止せず処理を続行する．

4.6 プログラムのローデング方式

シミュレータ上で実行される並列プログラムは，実行に先立ち，主記憶装置用M配列にロードされねばならない．プログラムの規模が小さい場合，GPSS の初期値設定機能（initial 命令）を用い，主記憶装置用M配列にプログラムを1語ずつ設定しても手間はかからない．しかしプログラムの規模が大きくなると，M 配列への設定を initial 命令で1語ずつ記述するのは非能率的である．

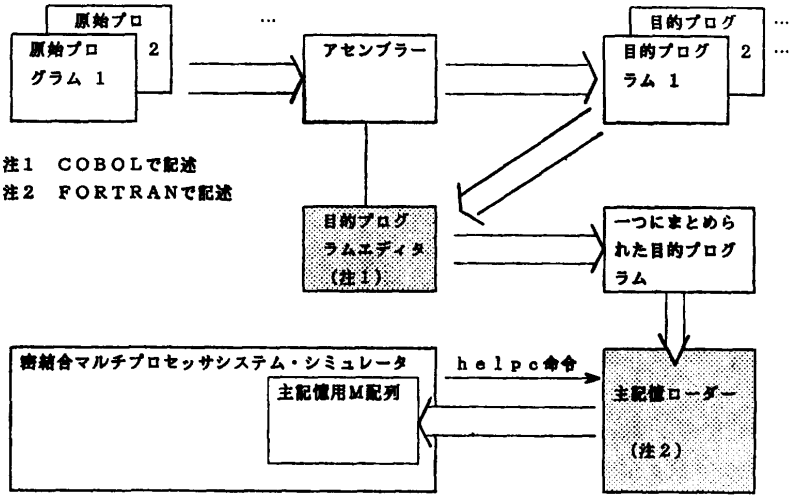

図 3 大規模プログラムのローデング方式
Fig. 3 A method to load a large program in the disk file into matrix savevalue.

そこで**図3**に示すように，GPSS の helpc 命令を用い，磁気ディスクファイルから主記憶装置用M配列に自動的にプログラムを読めるようにした．アセンブリ言語で記述した原始プログラムをアセンブルし，目的プログラムファイルを作り，次に『目的プログラムエディタ』で，（もしあれば）複数の目的プログラムファイルを一つのロードモジュールファイルに編集する．シミュレーション実行時，helpc 命令により『主記憶ローダ』を動的に呼び出し，上記ファイルの内容を一語単位に主記憶用M配列に書き込む．

この方式の採用により大規模ロードモジュールのシミュレータへの動的なローデングが容易になり，シミュレータの利用範囲が拡大した．図3にこの関係を示すが，目的プログラムエディタは COBOL，主記憶ローダは FORTRAN で記述した．

4.7 性能計測機能とデバッグ機能

（1） メモリ参照履歴の収集機能

シミュレーションの実行時，オプションで指定すると，それぞれ下記の装置でアドレスなどの参照情報や参照時刻，ヒットの有無などの情報を時系列的に採取する．また簡単なコーデングの修正で採取時刻の範囲を限定したり，特定の処理装置や主記憶についてのみ採取することなどもできる．

（a）参照アドレス，読み書きの別，要求発生時刻と終了時刻，固有キャッシュのヒットの有無，書き込みデータなどの情報を処理装置単位に収集する．

（b）要求した処理装置の番号，参照アドレス，読み書きの別，要求発生時刻と終了時刻，共有キャッシュのヒットの有無，書き込みデータなどの情報を主記憶接続路単位に収集する．

（c）要求した処理装置の番号，参照アドレス，読み書きの別，要求発生時刻と終了時刻，後置キャッシュのヒットの有無などの情報を主記憶接続制御部単位に収集する．

（2） ヒット率の計測

固有キャッシュ，共有キャッシュ，後置キャッシュは，参照される都度，命令読み出し，演算数読み出し，書き込みなど動作種別ごとにヒットの件数やミスヒットの件数を計測する．この情報をもとに，各キャッシュのヒット率を動作別に算出することができる．

（3） デバッグ命令の設置

プログラムが並列に動作する環境では，そのシステム上で走行するプログラムの誤りやシステムの論理設計不良のため，誤動作が発生しても，並行して動作中の他のプログラムは動作し続ける．このため誤動作発生時点の正確な状況把握が困難で原因解明が難しい．

この事情はシミュレータの場合も同じである．そこで，その都度再コーデングは必要だが，異常状態に応じて，その原因解明に有益な情報を収集できるデバッグ命令を設けた．原始プログラム中の適当な箇所にこ

の命令を挿入することにより，誤動作の解明に必要な情報を得ることができる．この命令はロジックスコープやインサキットエミュレータに相当するもので，例えばデバッグ命令発行時点の各種レジスタ，各所のバスあるいは主記憶の内容，現在時刻など GPSS で参照可能な情報はすべて採取できる．

（4） タスクの実行時間計測機能

シミュレータではマルチタスク制御命令群の開始時刻や起動割込みの発生時刻を，処理装置単位に時系列的に記憶する．これらによりタスクの開始から終了（または待機）までの時間を知ることができる．

4.8 シミュレーションの高速化

シミュレーションの高速化のため次の配慮をした．

（1） 命令間の割込み検査の廃止

一般に割込み要求の有無は命令の終了直後で次の命令の読み出し前に調べられる．この方式をそのまま用いると，タスクを実行中の処理装置はもちろん，タスクを割り当てられていない処理装置も，いつ発生するかわからない割込みの検出のため，割込みを検出可能な命令からなる遊休ルーチンを繰り返す必要がある．この非生産的な遊休ルーチンは，それを構成する各命令の実行に XAC が待たされる度合いが少ないほど，繰り返し回数が多くなり CPU 処理時間を消費する．そこで命令終了の都度割込みを検査する機能は削除し，割込みを停止中の処理装置への起動割込みに限定した．

（2） マルチタスク制御命令のファームウェア化

マルチタスク制御命令は通常の機械命令語の範囲を超えた複雑な機能をもっている．したがって実現するには SVC 割込みを用いたマクロルーチンによるか，ファームウェアを用いた機械命令による必要がある．前者はその機能をシステム 370 で定義された命令を用いて記述するため，シミュレーションに時間がかかるほか，SVC 割込みを許す必要もある．このいずれも認めることができないため，表 4 に示したマルチタスク制御命令は機械命令として設け，その機能をファームウェアで実現することを想定し GPSS で記述した．

4.9 シミュレータの記述規模

シミュレータの記述量は，手続きの記述に約 7.5 K 命令を使用しているほか，M配列や関数 (variable) などの定義に 1.5 K 行を費やし，注釈行を合わせると合計で 13.5 K 行の規模になった．なお表 2 に各部の手続きの記述に費やした命令の概数を示すが，マルチタスク制御に最も多くの命令を用いている．

5. シミュレータの利用例

以下では『重複ブロックマージソート法』と呼ぶ並列マージソートプログラムを例に，シミュレータの利用経験を述べる．

5.1 例題プログラムの原理と構成

複線マージ分割に基づくブロック奇偶ソート法[10]では，p 組の入力データブロックに対し，$p/2$ または $(p/2)-1$ の並列度の複線マージ分割操作を p 段繰り返すことにより，ブロック単位に整列した p 組のデータブロックの順序づけられた並びを得る．『重複ブロックマージソート法』は先の算法で p が大きいと処理時間が増大する欠点を改善するもので，複線マージ分割[10]の出力である p 組のデータブロックのキー値の最大値と最小値をもとに，ブロック間のキーの価域の重複を調べ，重複する二つのブロックについて複線マージ分割をする操作を，すべてのブロック間で重複がなくなるまで繰り返すことでデータを整列する．

5.1.1 例題プログラムの構成要素とその機能

（1） 整列主タスク：本タスクはジョブの開始で起動され以下の機能をもつ．

（a） 8 組の整列タスクの生成をマルチタスク制御部に順次依頼した後，指定した二つの整列タスクの終了で自動的に起動される 4 組の複線マージ分割タスクの生成を依頼し，自身はその終了を待って休止する．

（b） 併合制御表整列サブルーチンを用い，後述の併合制御表に報告された各ブロックのキーの最大値により，併合制御表の各行を降順に整列する．

（c） 併合制御表に報告された各ブロックのキーの最大値と最小値により，データブロック間のキー値の重複を調べ，重複があればバッファ管理ルーチンに依頼し二つの出力バッファを確保した後，併合タスクを起動する．上記の操作をまず制御表の先頭行と 2 行目の間で行い，それ以降は先に併合タスクを起動してなければ 2 行目と 3 行目，併合タスクを起動していれば 3 行目と 4 行目について同様の検査を行う．この操作を制御表の最後の行に達するまで行う．

（d） 先の操作で 1 件以上の併合タスクを起動していれば，それらのすべての終了を待った後，(b)から再開する．なお終了を待つ間に併合操作の対象となったブロックが占めている領域を返却する．

（e） 1 件の併合タスクも起動しなかった時は整列の終了を意味する．この場合は作業バッファ中の整列

済のデータを出力域に移すため8組の転送タスクを生成し，それらの実行終了を待った後ジョブを終了する．

（2）併合制御表：この表にはデータブロックの主記憶装置上のアドレスとそのブロックに含まれるキーの最大値と最小値などの情報が記録される．併合タスクはその終了に先立ち，出力した2つのデータブロックのキーの最大値と最小値をこの表に報告する．

（3）整列タスク：実行時パラメータで指定された領域の指定数のデータをクイックソート法で整列する．

（4）併合タスク：実行時パラメータで指定された2つの領域のデータを指定された数だけ複線マージ[10]し，指定された2つのバッファ領域に出力する．

（5）転送タスク：指定された領域にあるデータを指定された出力領域に指定された数だけ転送する．

（6）併合制御表整列ルーチン：主タスクから呼び出され，併合制御表に報告された各ブロックの最大キー値に着目し，併合制御表の各行を降順に整列する．

（7）バッファ管理ルーチン：主タスクから呼ばれ，要求された数のバッファの貸与と返却の処理をする．

5.1.2 マルチタスク制御部との関係

本プログラムの主要なタスクとマルチタスク制御部との関係を図4に示す．

（1）整列主タスクが併合操作や整列操作の子タスクを生成する時は，データの起点アドレスやデータ数などの実行時情報（図ではEPと略称）を任意の領域に用意し，そのアドレスと起動アドレス，タスク識別番号などのタスク生成用の情報（図ではAPと略称）を任意の領域に書き，そのアドレスを特定の汎用レジスタに入れ，タスク生成命令を発する．

（2）タスクが事象の発生を待機する場合，待ち事象の種別と対象タスクの識別番号などのタスク待機用情報を任意の領域に書き，そのアドレスを特定の汎用レジスタに入れ，タスク待機命令を発する．この時当該タスクを実行していた処理装置の固有キャッシュは，演算数を記憶しているブロックに限り無効にされる．

（3）タスク終結命令でタスクの終了の処置がなされる．当該タスクの終結を待機中のタスクは，この命令の実行過程で再開の処置がとられる．なお待機命令と同様，固有キャッシュ中の演算数を収容したブロックは無効にされる．この操作は無用なバッファ合わせの発生を避けるためなされるが，該当処理装置のキャッシュから演算数を含むブロックが削除されたことを，共有キャッシュの管理情報に反映させる必要がある．このため共有キャッシュの全ブロックから演算数を収容し該当処理装置の複写表示子がセット状態にあるブロックを探し，リセットする指示を，接続された全主記憶接続路に指令する．

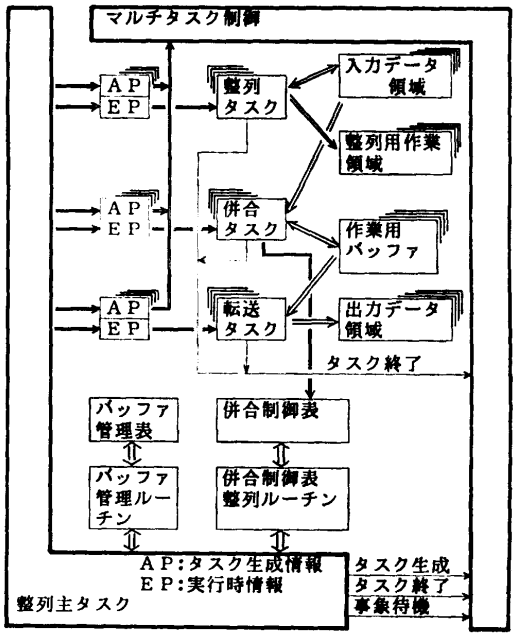

図4 例題ジョブの構成とマルチタスク制御部との関係
Fig. 4 Task structure of parallel merge-sort program.

5.2 シミュレーションの実行

シミュレーションジョブはGPSSで記述したシミュレータをアセンブル後，その実行を始める．

（1）まずシミュレータは下記の初期設定作業を行う．

（a）4.6節の方法で磁気ディスクに格納されているジョブ制御プログラムと並列マージソートプログラムを主記憶用M配列群にロードする．

（b）整列用テストデータに用いる乱数を発生し主記憶用M配列群に書きジョブ制御タスクを起動する．

（2）ジョブ制御タスクは重複マージソート用整列主タスクの起動をマルチタスク制御部に頼み，自らはその終了を待って休止する．

（3）その後は整列主タスクの指示で必要に応じタスク群が生成され，複数の処理装置上で並列に動作する．

（4）整列主タスクが並列ソートプログラムの終了点を検知すると，その出口でジョブ終了命令を発し，休止中のジョブ制御タスクを再開する．これによりシミュレーションジョブが終了する．なお主記憶用M配列にロードした手続コードは，同時に複数のタスクで

共用するため，再入可能な形式でコーデングする．

5.3 並列ソートジョブの動作解析

960 組のデータが記憶されている領域を，120 組のデータをもつ8つの領域に等分し，8台の処理装置にクイックソート法で整列させた後，前述のマージソート法により整列するジョブを実行させた．

（1） タスクの実行状況

シミュレータに設けたタスクの実行時間計測機能を用いると，並列に動作する各タスク（したがってタスクの集合としてのジョブも）の実行時間を知ることができる．**図5**はこの機能から得た情報により，例題ジョブの各タスクの実行状況を処理装置単位に図示したものである．各タスクの開始と終了時刻，待機開始時刻と再開時刻などがわかるため，ハードウェア各部の速度や処理方式の変更の効果，算法の良否などを所要時間の大小により判断できる．

（2） キャッシュのヒット率

シミュレータは各キャッシュのヒットの有無などの情報を，メモリ参照の都度計数している．例題ジョブについてこの計数情報を集計し，キャッシュの参照状況としてまとめた．なお本論文はシミュレータの構成に関する報告のため，ここで得られたキャッシュの利用率や，ヒット率そのものの考察は割愛する．

（a） 固有キャッシュの参照状況

表5は例題ジョブの実行中に発せられたメモリ参照要求を，処理装置単位に，命令読み出し，演算数読み出し，演算数書き込みに分け，それぞれヒットとミスヒットの件数を集計したものである．

表5 処理装置別メモリ参照状況
Table 5 Number of memory references of each processing unit.

処理装置番号	命令読み出し		演算数読み出し		演算数ストア	
	ヒット	ミス	ヒット	ミス	ヒット	ミス
1	13,401	103	4,083	231	1,281	225
2	9,806	66	4,269	369	937	346
3	18,803	44	6,457	367	1,243	287
4	18,110	43	6,307	365	1,262	281
5	13,858	34	4,489	223	886	161
6	9,292	13	2,640	80	512	35
7	8,664	13	2,450	80	518	36
8	8,887	13	2,533	78	534	34
9	8,934	13	2,528	79	542	36
合　計	109,755	342	35,756	1,872	7,715	1,441

この表の利用はその視点により種々考えられるが，例えば処理装置番号2のヒット率が特に低いことがわかるので，ミスヒット時の参照アドレスを収集し，その原因を追跡することなどもできる．

（b） 各キャッシュの参照状況

3種のキャッシュのヒット率を**表6**に示す．同表で固有キャッシュは動作した全処理装置の総和を，共有キャッシュと後置キャッシュは，16組の主記憶接続路に発出された参照要求の総和を示している．

この表からは，例えば，例題ジョブでは後置キャッシュへの参照頻度が少ないことや，演算数のストア動作の場合，後置キャッシュのヒット率が高く，先取り

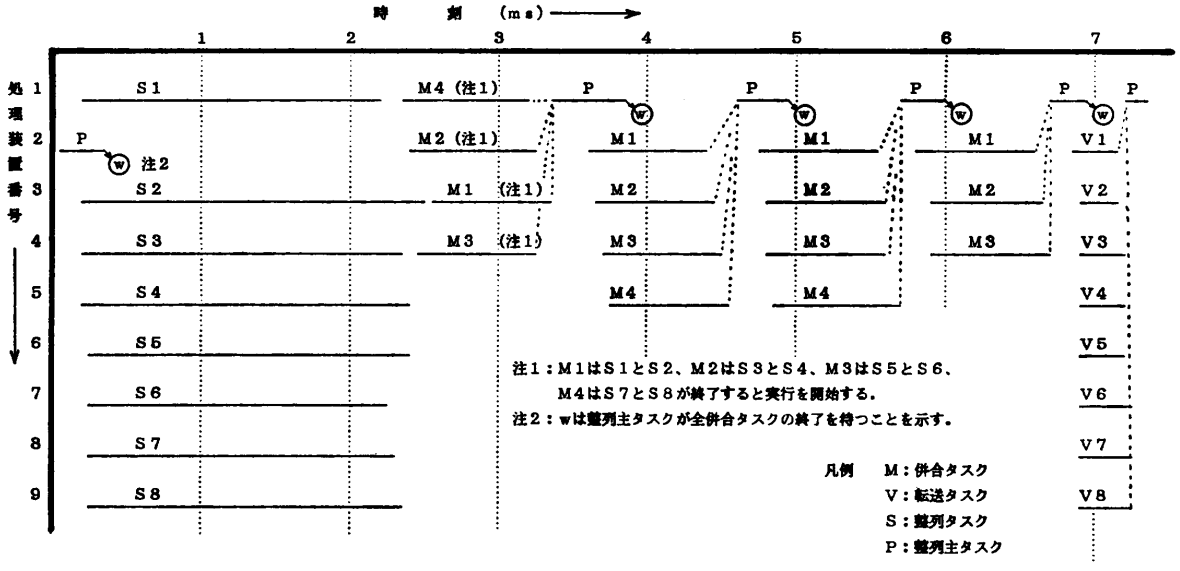

図5 重複ブロックマージソート法のタスク群の実行状況（データ数：960）
Fig. 5 Timing chart of overlapped block merge-sort method in case of 960 integer data.

表 6 各キャッシュの参照状況
Table 6 Access property of each cache.

	参照種別	ヒット件数	ミス件数	ヒット率
固有キャッシュ	命令読み出し	109,755	342	99.7
	演算数読み出し	35,756	1,872	95.0
	演算数ストア	7,715	1,441	84.3
	割込み連絡域	0	37	0.0
	合計	153,226	3,692	97.6
共有キャッシュ	命令読み出し	217	125	63.5
	演算数読み出し	1,557	352	81.6
	演算数ストア1	7,715	0	100.0
	演算数ストア2	142	1,299	9.9
	合計	9,631	1,776	84.4
後置キャッシュ	命令読み出し	112	13	89.6
	演算数読み出し	211	141	59.9
	演算数ストア2	1,188	111	91.5
	合計	1,623	265	86.0

注) 割込み連絡域の読み出し操作は常に共有メモリを参照する.
演算数ストア1は固有キャッシュに写しのある領域へのストア.
演算数ストア2は固有キャッシュに写しのない領域へのストア.

の効果が現れていることなどを知ることができる.

なお必要なら,これらのデータを主記憶接続路単位に集計することもできる.

(3) バッファ合わせ

固有キャッシュに複写された記憶域を他の処理装置が書き換えた場合バッファ合わせが必要になるが,例題ジョブでは下記の理由からバッファ合わせは1件も発生しなかった.

例題ジョブでは親タスクが書いた情報(実行時パラメータ)を子タスクが読み,子タスクが応答した情報を親タスクが読むという形で情報を共用するほか,同一作業バッファも繰り返し使用される.またタスク制御の関係では,起動するタスクに関する情報が割込み連絡域に書かれて渡される.しかし子タスクはその終了時,タスク終結命令で固有キャッシュの演算数を収容しているブロックを無効にしているため(5.1節参照),上記の場合でもバッファ合わせは発生しない.また表3に示したマルチタスク制御用の管理表群を主記憶中に設ければ,その変更に伴いバッファ合わせが発生するが,本シミュレータでは専用メモリに収容しているためバッファ合わせの発生はない.

再書き込み方式のバッファ合わせを用いる場合,タスク終了時に演算数ブロックを放置すると固有キャッシュの容量を大きくするほどバッファ合わせが多く発生する.したがってこの場合はタスクの休止時や終結時に固有キャッシュの演算数を含むブロックを無効にする機能は不可欠と言える.

(4) 複写表示子のリセット操作

固有キャッシュの演算数を収容したブロックが置換される時,共有キャッシュに設けた複写表示子を無効にする.シミュレータでは固有キャッシュの行(ライン)数の標準値を4としたため置換される頻度も高く,例題ジョブの場合,361件の複写表示子無効化処理を行った(行の数を8に増やすと皆無になった).

(5) 誤動作の原因追及

シミュレータの開発はバッファ合わせを含め図1に示した各装置のメモリ参照に関する制御系模擬部を動作させ,次いでデータ系の回路群(命令やデータを記憶する主記憶やキャッシュ等のメモリ部とその読み書き制御部,データバスなど)と命令実行機構,helpc命令によるプログラムの動的読み込み機構などを追加し,その後マルチタスク制御機構や相互排除機構などを実装した.この間機能の複雑なマルチタスク制御機構や,時間的に非同期に発生するバッファ合わせ処理などを中心に,多くの設計不良が検出された.またプログラムが並列に動作するようになっても,並列ソートプログラムのバグや,それまでに未検出のマルチプロセッサシステムの論理設計不良,マルチタスク制御機構のバグなどがあり,その都度原因を追及しシミュレータの完成度を高めてきた.対象システムでは三階層のキャッシュが動作すること,処理装置の並列動作によりスイッチ装置や主記憶装置が同時に動作するため,誤動作の原因追及が難しい.このような場合,まずメモリ参照履歴収集機能を用い誤動作発生の時点と場所を把握した後,デバッグ命令を用いて原因の探求を行った.数の上では少ないが,これらの機構なしには究明が不可能と思える設計不良が,模擬対象のバッファ合わせ機構やマルチタスク制御機構に存在した.

6. まとめ

6.1 性能評価用ツールとしての効果

本論文で例にしたプログラムを含め,4種の並列ソートプログラムを設計し,そのデバッグ後,各算法の性能評価の一環として整列対象のデータの数を160, 320, 480…960… と変えて実行させた.またキャッシュの1ブロックの容量や主記憶接続路の数など,ハードウェアの条件を変えて動作させ,性能に及ぼす

影響を評価した．それらの使用経験から，簡単なコーデングの変更で可能となる事項を含め，シミュレータは下記の実行条件を単独あるいは組み合わせてジョブを実行し，下記の評価項目を計測できることを確認した．

（1） 実行条件の設定

■固有キャッシュがある場合とない場合．
■後置キャッシュがある場合とない場合．
■主記憶接続路数の変化．
■キャッシュの行の数や1ブロックの容量の変化．
■キャッシュや主記憶のアクセス時間．
■キャッシュのストア方式．
■バッファ合わせ方式（無効方式と再書き込み方式）．

（2） 計 測 項 目

■ジョブ（タスク）の実行所要時間．
■各キャッシュの参照頻度とヒット率．
■メモリ参照動作の履歴．
■処理装置単位，主記憶接続路単位の参照状況．
■要求処理時間（要求発生時刻−終了時刻）．
■要求発生時の待ち行列の長さ（競合状況）．

6.2 シミュレーションジョブの規模

本シミュレータは所要時間の短縮のため，命令終了後の割込み検査の廃止を含め4.8節に述べた処置をした．その結果，例題ジョブを約23時間の正味 CPU 時間で実行できた．使用した計算設備は FACOM M 340 R 型処理装置（キャッシュなしで 0.7 MIPS 程度）で，約 7.5 MB の仮想記憶域を要した．なおシミュレータ上で動作する処理装置の数やキャッシュの行数を増やすと所要時間はさらに増えることを確認している（例えば固有キャッシュの行を4から8に増やすと28時間余の CPU 時間を要した）．

この程度の規模のジョブは，研究用計算施設においても大規模ジョブに属するであろうが，実行不可能な規模ではない．さらに性能の高いシステムを使用すれば所要時間が 1/2〜1/10 に減少することも期待できる．したがってメモリ参照動作や命令実行機能を回路レベルで模擬し，タスクを並列に実行するシミュレータは実現可能な性能評価ツールと考える．

6.3 実現上の隘路など

GPSS では整数型データは符号付 24 b 長である．主記憶用M配列に記憶された命令語はレジスタやバスを模擬する複数のM配列を経由して命令実行部に達するが，M配列への代入命令は 32 b の情報を扱えるので支障はない（しかし 64 b のデータバスをもつ大型システムを模擬する場合は，32 b ずつ2回に分けて記述する必要がある）．

一方，命令語から操作コード，R_1，X_2，B_2 などのフィールドを取り出すため，剰余演算や乗除算を用いる．しかし語の最上位ビットが1の時は負数とみなされるため，例えば RS 形式の命令コードや，8番以上の汎用レジスタを用いた B_2 フィールドが語の先頭に位置する場合は補正しないと正しい値が取り出せない．命令読み出し部の随所でこの補正が必要なため，命令解読部の記述量が増えた．この不便は 32 b の符号なし整数が使用できれば解消する．

6.4 並列算法検討ツールとしてのシミュレータ

下記の機能をもつことから，密結合マルチプロセッサシステム用並列算法の開発や評価に利用できる．（1）マルチタスク制御機構を装備している．（2）任意のプログラムを任意の時点に任意の場所に動的にロードできる．（3）デバッグ手段としてデバッグ命令やメモリ参照履歴収集機能などを備えている．（4）並列タスクの開始／終了時刻を把握できる．

7. おわりに

密結合マルチプロセッサシステムのアーキテクチャ検討用ツールとして，並列実行単位に分割されたタスクを複数の処理装置に割り付け，その起動・実行・終結を制御するマルチタスク制御機能と，IBM 社のシステム 370 系の機械命令を解釈・実行する命令処理部をもち，メモリの参照動作を論理回路のレベルで模擬するシミュレータを開発した．

このシミュレータは磁気ディスク上に記憶された目的プログラムを，主記憶を模擬する配列に動的にロードし実行することができる．

本論文では，論理回路を GPSS で記述する方法やシミュレータの構成方法，設計に当たり考慮した事項などを述べた後，例題として実行させた並列マージソートの走行結果から得たヒット率等の性能指標を紹介し，その実現可能性と有効性を示した．

今後は本来の目的であるキャッシュ系の評価に使用するほか，記憶域の制約を考慮しながら仮想記憶機構の実装をすすめると共に，並列ソートや並列計算の検討も計画している．

参 考 文 献

1) Enslow, P. H., Jr.: *ACM Comput. Surv.*, Vol.

9, No. 1, pp. 103-130 (1977).
2) Quatember, B.: Modular Crossbar Switch for Large-Scale Multiprocessor Systems-Structure and Implementation, *Proc. of AFIPS*, pp. 125-135 (1981).
3) 富田眞治：並列計算機構成論，p. 306，昭晃堂，東京 (1986).
4) 山本 登：マトリクス・スイッチ結合式マルチプロセッサシステムのバッファ合わせの一方式，情報処理学会論文誌，Vol. 26, No. 3, pp. 429-437 (1985).
5) 山本 登：二階層キャッシュをもつマルチプロセッサシステムにおけるアドレス変換機構の実装方式の考察，情報処理学会論文誌，Vol. 26, No. 3, pp. 438-445 (1985).
6) 三上 徹，亀田壽夫：コンピュータと情報システムにおけるシミュレーション技術，信学誌，Vol. 60, No. 7, pp. 761-768 (1977).
7) 山本 登：キャッシュ内蔵スイッチ装置を用いた密結合マルチプロセッサの後置キャッシュの検討，第 30 回情報処理学会全国大会論文集，5 B-1, pp. 111-112 (1985).
8) 石川裕之，村上勝彦，北川 一：アドレスパターンデータによるワーキングセットの解析，第 30 回情報処理学会全国大会論文集，2 B-3, pp. 59-60 (1985).
9) 松岡浩司，堀川 隆，大野直哉：トレースデータを使ったマルチプロセッサ性能評価システム，第 36 回情報処理学会全国大会論文集，5 C-8, pp. 247-248 (1988).
10) Hsiao, D. K., Menon, J. 著，野下浩平訳：並列ソート算法の分類，bit 臨時増刊号，pp. 91-117 (1986).

(昭和 63 年 7 月 8 日受付)
(平成 元 年 4 月 11 日採録)

山本 登（正会員）

昭和 14 年生．昭和 39 年 3 月東京都立大学工学部電気工学科卒業．同年 4 月(株)日立製作所に入社．同社神奈川工場に勤務し，HITAC-5020 E/F, 8500, 8800, DIPS-1 などの大型電子計算機や端末機の開発プロジェクトに参加の機会を得る．昭和 52 年 4 月に日本大学工学部電気工学科に移り，密結合マルチプロセッサシステムの構成法と並列処理の研究に着手し現在に至る．計算機アーキテクチャ全般およびソフトウェアエンジニアリング，知識工学などに興味をもつ．現在同大学助教授．電子情報通信学会会員．