蓄積プログラム制御用処理装置のメモリアクセス高速化手法

High Speed Memory Access Method
for Stored Program Control Processing Equipment

山田 喬彦　　宮山 哲
Takahiko YAMADA　　Satoshi MIYAYAMA
日本電信電話公社 武蔵野電気通信研究所
Musashino Electrical Communication Laboratory, N.T.T.

1．まえがき

高集積部品技術の進展とともに、大容量のメモリやマイクロプロセッサなどが極めて低価格で利用できるようになった。これに伴ない、従来、布線論理で実現されていた制御機能が、蓄積プログラム制御に置き換わる傾向が顕著である。蓄積プログラム制御方式では機能の変更、追加が容易であるが、反面これらに付随して、処理ステップが増加するため、より高速な処理装置が求められるのは必然である。

処理装置の演算部や制御部は部品技術の進展から自ずと高速化されていくが、メモリアクセスについては、大容量化などによる実装遅延を伴なうため方式的な手法による高速化が必要となる。（図１）

本報告では、並列記述機能をもつ高水準言語を利用したプロセスの性質を利用して、命令の先取り制御を効率良く行なうメモリアクセス高速化手法を提案する。また、蓄積プログラム制御の代表例である電子交換制御方式にこの手法を適用した場合の構成例及び概略評価について述べる。

2．メモリアクセス高速化の既存手法

2．1 蓄積プログラム制御方式の特性

蓄積プログラム制御方式のプログラム構成は図２のように、

① 被制御回路の状態を走査し、変化を検出して、状態の変化に応じたタスクを登録する実時間監視部

と、

② 登録されたタスクを処理し、被制御回路に制御情報を送出するタスク処理部

からなるのが一般的と考えられる。

一般に蓄積プログラム制御は、ハードウェア制御の論理をプログラム制御に置き換えたものが多いため、被制御回路の監視点が多く、呼の多重度が高いわりに各タスクの処理は小規模な処理の組み合わせによるものが多いという特徴がある。

蓄積プログラム制御方式の代表例である電子交換機においては運転されるプログラムは常時、メモリに固定的に格納されている。プログラムの単位は小規模であり、主記憶上に分散して配置されている。

2．2 異速度メモリ制御方式

一般に、蓄積プログラム制御方式のメモリアクセス範囲とメモリアクセス率の累計は、図３のようになることが予想される。このため、主記憶装置に高速なメモリと低速なメモリを置き、アクセス頻度の高いプログラムは高速メモリに静的に配置し他を低速メモリに置いて経済性と高速性のバランスをとった方式が採用された。これを異速度メモリ方式という。[１]

しかし、この方式では、

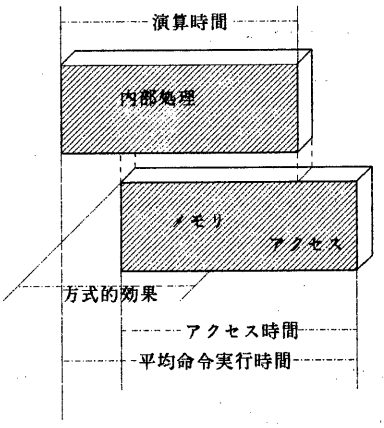

図1　命令実行時間の内訳

（1）機能の追加とともにプログラム量が増加し、高速メモリから高頻度アクセスのプログラムが脱落して処理能力低下を招く

（2）高頻度、低頻度のプログラムをプログラマが識別する必要が生じ、ソフトウェアの生産性が悪い

などの問題がある。

2．3　キャッシュメモリ方式

汎用計算機では最も一般的なメモリアクセス高速化手法であるが、蓄積プログラム制御の処理では図2に示したように単純な処理(タスク)の組み合わせが多いことから、汎用計算機の処理のように小ブロック内のプログラムを連続して何度も使うといったアクセスのローカリティは少ない。このため、蓄積プログラム制御にキャッシュメモリを採用してもヒット率は小さいと予想され、その効果はあまり期待できない。

3．　有効なメモリアクセス高速化手法

メモリ素子の技術動向では、低速大容量チップと高速小容量チップの2つの開発方向がある。この傾向を考慮し、かつメモリの大容量化の要求を満足するため、命令実行部(演算部)と主記憶の間に高速バッファ記憶を設ける方式がメモリアクセス高速化の前提となる原理である。この効果を十分引き出すには、

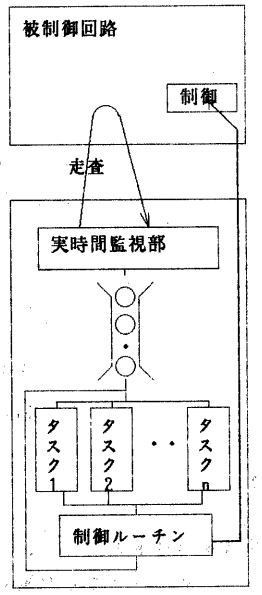

図2 蓄積プログラム制御のプログラム構成

高速バッファの利用率をできるだけ上げることが重要である。

ここでは、独立した多数呼の多重処理(後述)に適したプロセスの並列実行方式を利用したメモリアクセス高速化手法を提案する。

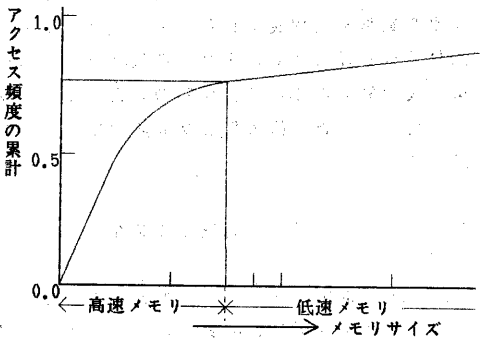

図3 異速度メモリ制御方式

4．蓄積プログラム制御における多重処理方式を考慮したメモリアクセス高速化

4．1 プロセス概念 [1]

多重処理において、独立な制御のもとで並列的に処理される対象(プロセス)の実行は、プロセス実体(図4)を使用して進行する。プロセス実体は、そのプロセスの実行及び状態を管理するプロセスコントロールブロック(PCB)と、実行中に使用するローカルデータをスタック方式で格納するワークエリアからなる。PCBのもつ主な制御情報は以下のとうりである。

(1) PCB自身の使用／未使用の表示
(2) プロセスの状態表示
(3) プロセス中断時の退避情報
(4) 実行待ちキューのリンク情報
(5) ワークエリアに関する情報

複数のプロセスは、それぞれがもつPCBの情報により多重的に実行される。

4．2 プロセスの並列実行[2,3,4]

プロセスにはI／Oアクセスにともなう待ち状態をもつため、実行、停止、実行待ちの3つの状態が存在し、図5に示した状態遷移にしたがう。

停止と実行待ちの2つの状態ではキュー制御によりプロセスを管理する。生成されたプロセスはまず実行待ちキューに登録される。並列機能の実行に伴い、停止状態にあってイベントキューにつながれたプロセスは中断が

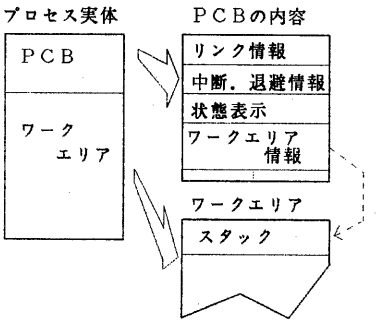

図4 プロセス実体

解除されたときに実行待ちとなる。

実行待ちキューは図6に示すようにチェーン構造をなしている。実行待ちのプロセスに対応したPCBは、他の実行待ちプロセスとのリンク情報として次につながれたプロセスのPCBを指すポインタをもつ。チェーンの先頭と最後尾は別にテーブルで管理する。

実行待ちプロセスは、PCBのリンクをたどって予め知ることが可能である。

4．3 メモリアクセス高速化手法

次にアクセスする対象を予め知ることが可能な場合には、実際のアクセスに先行してその対象を高速バッファへ転送する方法が考えられる。

ここでは多重実時間処理におけるプロセス実行環境及びキュー制御を利用した命令アクセス高速化手法を具体的に説明する。

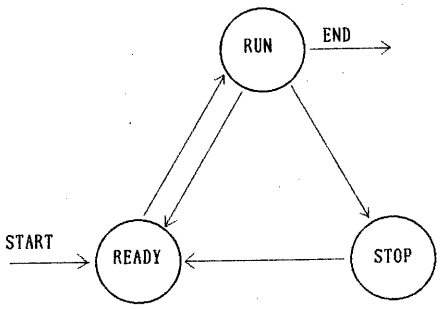

図5 プロセスの状態遷移図

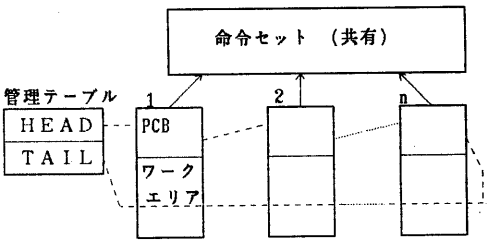

図6 実行待ちキューの構成

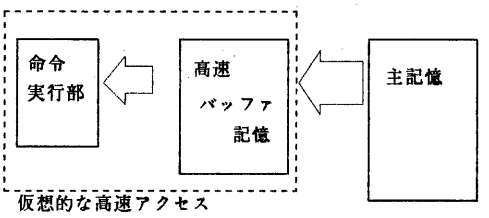

図７　　高速化手法の基本構成

（ｉ）　全体構成

本方式の全体構成を図７に示す。動作概要を順を追って示す。

①　まず実行待ち用キューに登録されているプロセスの識別と実行順を、キューにリンクされたＰＣＢ情報から検知する。

②　これをもとに次に実行されるプロセスの命令セットを実行に先行して主記憶から高速バッファに転送する。

③　命令実行部は所望の命令セットを常に高速バッファから読みだし可能となる。

④　高速バッファにない命令にアクセスしたときには現在実行中のプロセスを中断して、実行待ちキューに再登録する。

⑤　同時に、高速バッファへ命令セットを転送済みの他のプロセスに実行を移す。

このようにプロセスの実行を切り替えることで高速バッファへのアクセスを連続的に進行できる。

主記憶と高速バッファ間の転送速度が大きければ、キューから取り出されたプロセスが実行される時点ではそのプロセスの実行内容である命令セットの一部は既に高速バッファへ転送済みである。

分岐命令の実行などによりプログラムの走行が連続しないことを考えれば、命令セットの転送単位はあまり大きくても無効な転送が増えるだけでありバッファサイズの選択は重要である。

高速バッファは多面あって、複数の並列実行されるプロセスに割り当てる。バッファの各面の空き／塞りの管理及び主記憶からの命令セット転送

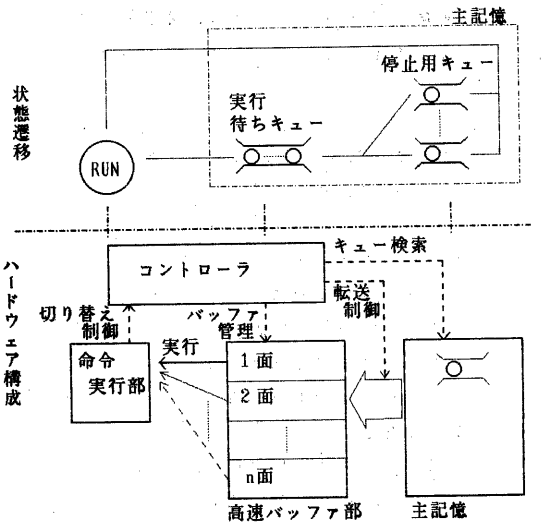

図８　　全体構成

制御は図8に示したハードウェアコントローラで行なう。　また、プロセスの実行切り替え制御では、命令実行部が未転送命令のアクセスを検出し、バッファ部に表示された各面の空き／塞り情報及びプロセスの実行待ちキューの情報を照合して切り替え先のプロセスを決定する。中断されたプロセスは前述のように実行待ちキューに再登録される。

このときキューでは新たに生成されたプロセスと本方式で中断されたプロセスとが混在することになる。本来、プロセスは各々独立な制御のもとに実行されるので、本方式で中断されたプロセスを実行待ちキューにつなぎこんでも支障ない。

（ii）　動作の説明

処理フローを図9に示す。全体の動作は主記憶から高速バッファへの転送制御と、命令実行・プロセス切り替え制御の２つに分けられる。

（１）　転送制御

①　ソフトウェア管理の実行待ちキューを検索し、実行待ち状態のプロセスがあれば、実行順にしたがってＰＣＢより命令開始アドレスを得る。

②　高速バッファ部の各面の空き／塞りを調べ、空きならば①のプロセスに

対応した命令セットの転送を起動する。全バッファ面が塞りのときは空くまで転送を待つ。

③ 命令セットの転送は、該当するプロセスの使用するプログラムの先頭アドレスあるいは中断アドレスからある一定の単位で行なう。

④ 転送終了時にバッファ面の塞りを表示しその面とプロセスとの対応を管理する。

（２） 命令実行・プロセス切り替え制御

① 転送済みの高速バッファがなければ転送終了を待つ。既に転送済みのバッファ面があればそのプロセスの実行を開始する。

② 高速バッファより命令セットを逐一読み出して実行する。

③ 転送単位分の実行終了あるいはプログラム中の分岐命令により高速バッファにない命令をアクセスした場合には、命令実行部でこれを検出する。

④ 現在実行中のプロセスを中断し、ＰＳＷ、レジスタ内容などの実行中断情報を再開のため退避する。

⑤ 退避終了後、中断プロセスを実行待ちキューに再登録する。

⑥ バッファ部の各面の空き／塞り、及びプロセスの実行順をみて、切り替え先のプロセスを決定する。

⑦ 中断したプロセスが使用していたバッファ部に空きを表示する。

⑧ 次実行プロセスの実行情報をロードし実行を開始する。

５．電子交換制御方式への適用

５．１ 電子交換プログラムの特性

蓄積プログラム制御用処理装置の代表例として、交換．通信処理システムにここで述べた方式を適用することを考える。交換処理は多数の呼を同時に処理することから、以下のような特徴がある。

（１） 端末数、同時接続数が大きい

（２） 多数の呼で共通リソースを共用する

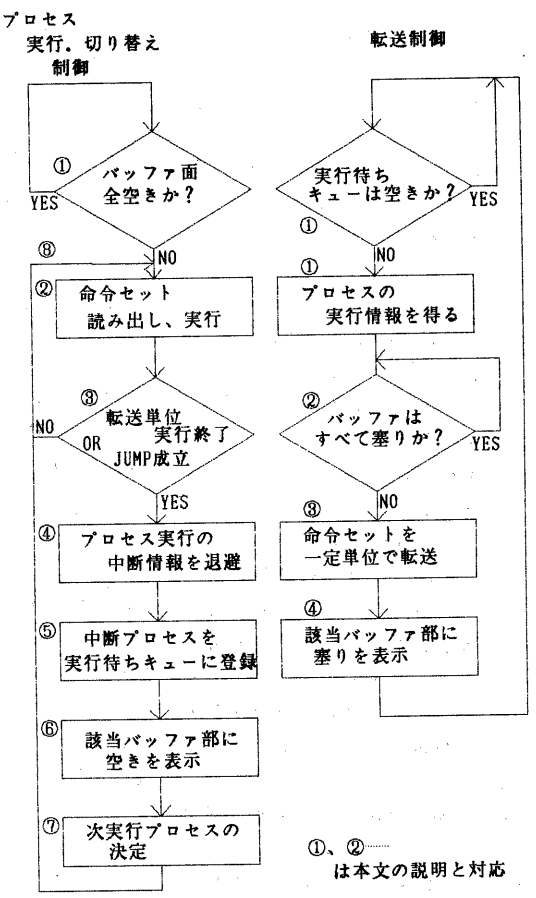

図９ 動作フローチャート

（３） 固定的な制御プログラムを常時運転するため、同一プログラムの寿命が長くかつ多数局で使用される

（１），（２）の特徴は多重実時間処理に共通しており、（３）は蓄積プログラム制御の一般的な特徴といえる。（１），（２）に対応して、呼に関する処理が相互に独立なことから呼ごとにプロセスを与えて独立に並列実行する多重処理方式が有効であり、交換．通信処理でもコンカレントＣＨＩＬＬを使用し、プロセス概念を導入した多重実時間処理を簡明に記述する方向にある。[5,6]

交換処理ではプログラム構成として優先順にＨ、Ｌ、Ｂの各レベルを設け、これらを一

定周期で多重処理している。特にBレベルのプログラムは主に多数のタスクとよばれる処理単位によって呼に関する論理的な処理を分担しており、他のレベルより処理量が大きい。従って、交換処理では処理能力向上のためBレベルプログラムを高速に実行することが必要である。

Bレベルプログラムは、実時間性が比較的厳しくなく、タスク管理にキュー制御を行なっていることから、呼対応にプロセスを与えることにより本方式を適用できる。高速バッファを使用したシステム構成例として、表1のように命令セット、データを格納した図10の構成が考えられる。またH、Lレベルのプログラムは比較的小容量であることから高速バッファに常駐可能である。

処理の実行に必要なデータについてはキー情報、ワークエリア、及び大容量のテーブル形式をとったサービス用データがある。

ワークエリアはスタック方式をとることとし、スタックに積み上げたデータのうちすぐに使用するものを高速バッファに置く。

サービス用データはアクセス頻度が低いことから主記憶への直接アクセスとする。

表1　アクセス対象格納法

	アクセス対象	特性	格納法
プログラム	Bレベルプログラム	大容量 低頻度	高速バッファへの転送
	H,Lレベルプログラム	小容量 高頻度	高速バッファ常駐
データ	ワークエリア	FIFO スタック	一部 高速バッファ常駐
	キュー	小容量	高速バッファ常駐
	テーブル形データ	低頻度	MM常駐

5．1　提案した方式の効果測定

ここで提案した方式について命令アクセス高速化への効果をみるため、電子交換制御方式へ適用した場合について、シミュレーションを行なった。その方法の説明図を図11に、評価のパラメータを表2に示す。シミュレーション手順を以下に示す。

① 交換プログラムを実際に走行させたときの命令アドレスの軌跡データを計算機にかかるよう変換する。

② 計算機上で、データファイルからBレベルのプログラム軌跡を抽出する。

③ 作成したデータファイルをシミュレーションプログラムの入力ファイルとする。

④ 本体プログラムは命令実行．プロセス切り替え部とデータ転送部からなり、本方式を採用した処理装置の平均命令実行時間で規格化した時間単位で両者が同時に実行されるように模擬する。

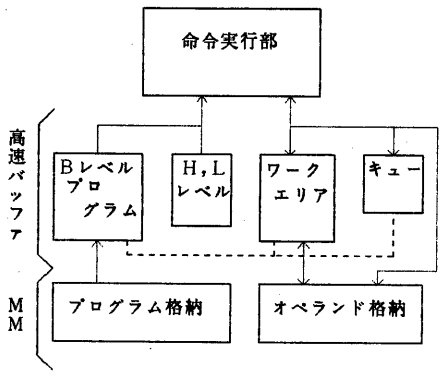

図10　システム構成例

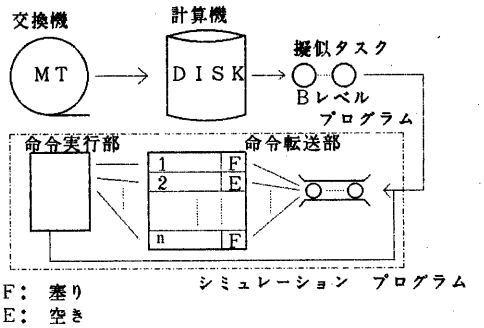

図11　シミュレーションの説明

表2　評価パラメータ

関連部	パラメータ
高速バッファ	ブロック数(面数) n
	転送単位　　Sz [W/面]
主記憶ーバッファ命令転送	1w当たりの転送時間 Ttr [τ]
プロセッサ内部処理	切り替えに要する時間 Tts[τ]

(注)　Ttr, Ttsは、平均命令実行時間τで規格化した

⑤　全アクセスのうち直ちに高速バッファから読み出した割合(高速アクセス率)を出力する。

本シミュレーションの結果を図12、13に示す。図12はバッファが一面しかない場合、図13は2面ある場合の高速アクセス率を、転送単位をパラメータに示したものである。

この結果をみると、転送速度とプロセスの切り替え速度の影響が大きい。特に、転送時間については、1W当たり平均命令実行時間(τ)程度あるいはそれ以下の転送時間に抑える必要がある。1W当たり(1/2)τの転送時間では、転送単位32Wとした場合切り替え時間によらず、80－90%以上の高速アクセス率が得られる。　ただし、転送単位が小さい時には切り替え動作の回数が増えるため、切り替えに要する時間の影響が大きく現われる。

本方式の効果を規定する要因は、高速バッファ記憶と主記憶の転送能力(転送速度)及び実行プロセス切り替え速度にある。

転送速度に関してはかなりの速度が要求されるといえる。このため、ニブルメモリの使用、インタリーブの採用、転送バス幅の拡大といった高速転送手段を施す必要がある。

プロセス切り替え速度の向上のため、VLSIを活用したレジスタファイルによるレジスタの多面化などの手段をもって退避情報、再開情報の転送量を極力減らすことが必須であり、これらは今後の課題といえる。

今回のシミュレーションでは機構を簡単化したためにバッファ面数3以上については評価出来ていない。　2面の場合のみ示したが、本方式の本来の効果は多面のバッファ間で実

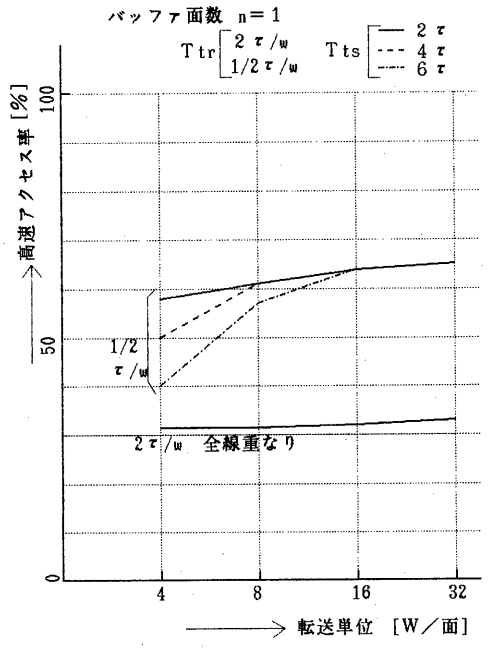

図12 シミュレーション結果 その1

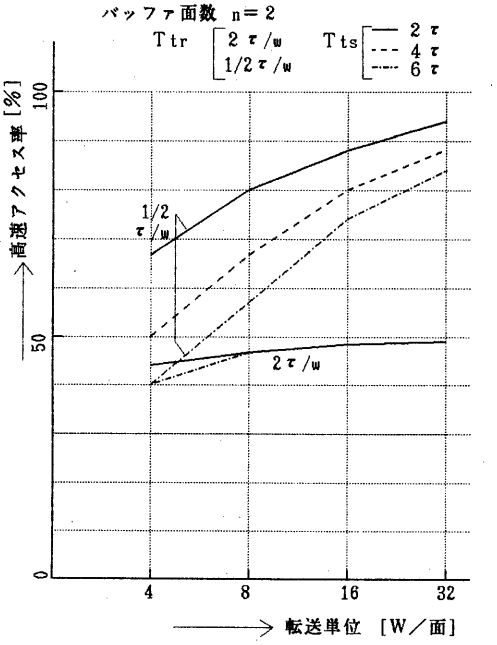

図13　シミュレーション結果　その2

行切り替え制御を行なって高速バッファからの命令アクセスを連続させることにある。この場合今回の結果よりさらに高い高速アクセス率が期待できる。

おわりに

蓄積プログラム制御用処理装置として、交換.通信処理などの多重実時間処理に適した処理装置の高速命令アクセスの一手法を提案した。またこの方式について簡単なシミュレーションを行ない、その効果をおおまかに評価し有効性を示した。

今後、ここで提案した方式について高速バッファと主記憶間の高速転送手段の実現法、多面バッファを有効利用できる切り替え機構、ワークエリアへの高速アクセス手法などの検討を進めていく。

最後に、本報告にあたって有益なご意見、ご助言を頂いた武蔵野通研、処理方式研究室の浜田泰昭室長、脇村慶明室長補佐、久保田稔研究主任に深謝致します。

[参考文献]

[1] 丹羽: D10用高速中央処理系装置中央制御装置の構成、研究実用化報告、Vol. 26 No.12

[2] 久保田: CHILL並列処理機能を用いた交換プログラムの実行制御手法、昭58 信学会総全大 458

[3] 久保田: CHILL並列処理機能の交換プログラムへの適用法、昭57 信学会総全大 164

[4] 田村 他: コンカレントCHILLを適用した交換プログラム構成、昭59 信学会総全大

[5] 丸山 他: 交換プログラム用並列処理言語、信学技報 Vol.79 No.125

[6] 丸山 他: 交換プログラム用仕様記述法と高水準言語、情報処理 昭55 Vol. 21 No.3