マルチメディア指向並列計算機 KUMP/D とそのメッセージ処理機構

富安洋史[†] 川野哲生[†,☆]
谷口倫一郎[†] 雨宮真人[†]

マルチメディアアプリケーションに必要な高速な動画像処理のために，様々なハードウェアアクセラレータが開発されている．しかし，将来のマルチメディアでは双方向の Visual Interface 等の複雑な処理が要求されるため，ハードウェアアクセラレータは不向きであると考えられる．我々は，次世代のマルチメディアアプリケーションのために，並列計算機 KUMP/D を設計製作中である．KUMP/D はメッセージをハードウェアで処理する FMP（Fine-grain Message Processor）と呼ぶ機構を設けており，細粒度並列処理を効率良く行うことができる．本稿では，KUMP/D の構成を述べ，細粒度並列処理において重要なメッセージ処理能力の評価を行う．

Fine-grain Message Handling Mechanism in Multi-media Oriented Parallel Processor KUMP/D

HIROSHI TOMIYASU,[†] TETSUO KAWANO,[†,☆] RIN-ICHIRO TANIGUCHI[†] and MAKOTO AMAMIYA[†]

High speed image and video processing is a key technology in multi-media applications. Currently, many hardware accelerators are developed and used in order to speed up such processing. However, the hardware accelerators are insufficient for the advanced multi-media applications such as bidirectional visual interface. Considering the above situation, we have been developing a multi-media processor KUMP/D (the Kyushu University Multi-media Processor on Datarol-II). The KUMP/D can efficiently execute a fine-grain multi-thread program by FMP (Fine-grain Message Processor). This paper describes the structure of the KUMP/D and the performance analysis of message handling, which is important for fine-grain multi-thread processing.

1. はじめに

動画像処理等を行うマルチメディアアプリケーションでは非常に大きな処理能力が要求される．従来，この要求を満たすためにハードウェアアクセラレータが用いられてきた．ハードウェアアクセラレータは動画像の圧縮伸張など，定型の処理を高速化するのには適している．しかし，近い将来必要になると思われる高度なマルチメディア処理では，動画像理解や，双方向のビジュアルインタフェースなどが要求されるであろう．これらは知識処理や多体問題などの処理を含み，パターンマッチングや探索問題などを解く必要があり，また，与えられるデータによって処理が大きく変化するため，ハードウェアアクセラレータで高速化することが難しい．

そこで，我々はこれらの知識処理を含む高度なマルチメディア処理を高速化するため，MIMD 型のマルチメディア指向並列計算機 KUMP/D（the Kyushu University Multi-media Processor on Datarol-II）[1] を設計試作している．

一般に，並列処理において問題となるリモートメモリアクセス等によって生じる待ちを隠蔽するためには細粒度並列処理が有効であると考えられている[2)~4)]．KUMP/D のプロセッサエレメントは Datarol-II アーキテクチャ[5),6)]に基づいて設計されており，スレッドと呼ぶ細粒度の処理単位を効率良く実行する．

また，KUMP/D は独立した高速な I/O ネットワークを持ち，動画像の入出力に必要なバンド幅を確保す

† 九州大学システム情報科学研究科
Department of Intelligent Systems, Graduate School of Information Science and Electrical Engineering, Kyushu University
☆ 現在，NTT ソフトウェア研究所
Presently with NTT Software Laboratories

るとともに，各 PE で動作するプロセスと画像との同期を行う機構を設けている．

このように KUMP/D は動画像理解や，高品質のバーチャルリアリティなどを用いたマルチメディアアプリケーションを高速化することを目的としている．本稿では，KUMP/D における細粒度メッセージ処理機構の構成と，その性能評価について述べる．

2. KUMP/D

2.1 概　　要

図 1 に KUMP/D の全体図を示す．各々のプロセッサエレメント（PE）は Datarol-II アーキテクチャ[6]に基づいて設計されており，2 次元トーラス状の PE 間ネットワークで結合されている．PE 間ネットワークは細粒度のメッセージのために最適化されており，パケットバッファを多重化することによってデッドロックフリーとなっている．

また，KUMP/D はリング状の I/O 専用 ネットワーク（I/O network）を持ち，動画像データの転送に必要なバンド幅の確保および，画像入出力と各 PE 上の処理との同期を保証している．

外部ビデオコントローラ（EVC）は，各 PE への画像のマッピングを変換する機能を持つフレームバッファを複数持ち，画像の入出力を行う[1]．

2.2 PE（Processor Element）

図 2 に PE の構成を示す．各 PE は，スレッドと呼ぶ細粒度の処理要素を効率良く実行する Dataroll-II アーキテクチャに基づいている．

KUMP/D では Datarol-II アーキテクチャを汎用マイクロプロセッサと FMP（Fine-grain Message Processor）と呼ぶ細粒度並列処理のためのハードウェアで構成している．

現在の商用高性能マイクロプロセッサは安価であり，専用に設計したプロセッサよりも高速にスレッド本体を実行することができる．このような汎用のマイクロプロセッサを要素プロセッサとして用い，効率良く動作させることができれば，優れた性能の計算機を短期に設計し，安価に製作することができる．

しかし，一般的な市販マイクロプロセッサはプロセッサ間通信のための機構を持たないため，KUMP/D では細粒度のメッセージ処理を行うハードウェアを付加している．以下 2.3 節で，付加ハードウェアを単純化し，高速化と低廉化をはかるために，通信を単純化した細粒度メッセージ駆動方式（FMD: Fine-grain Messege Driven Mechanism）を提案し，2.4 節で FMD に基づく付加ハードウェア FMP について述べる．

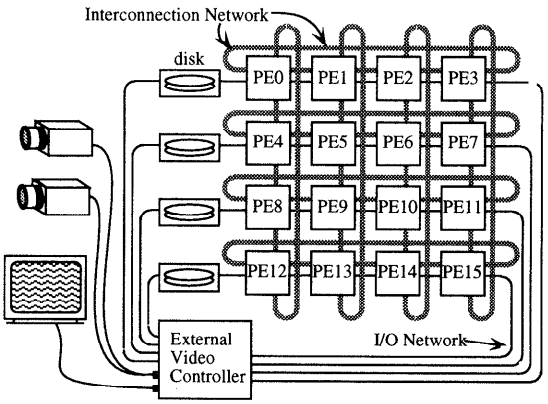


図 1 KUMP/D の構成
Fig. 1 Overview of the KUMP/D.

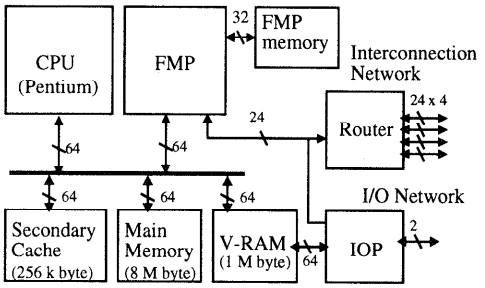


図 2 プロセッサエレメント
Fig. 2 Processor Element (PE).

また，KUMP/D では，マルチメディアアプリケーションで重要となる I/O 要求を処理する IOP（I/O processor）を設けている．IOP は画像の入出力を自動的に行うとともに，画像と PE 上の処理との同期を補助する機構を持つ．この IOP の画像入出力と同期機構の詳細については 2.6 節で述べる．

2.3 細粒度メッセージ駆動方式（FMD）

細粒度メッセージ駆動方式（FMD: Fine-grain Message Driven Mechanism）は，Datarol-II のパケットをより単純化したメッセージにより実行を進める方式である．

これらのメッセージを用いた，関数適用の手順について説明する（**図 3** 参照）．FMD では，各関数実行のインスタンスはフレームと呼ぶ記憶領域を持ち，関数は複数のスレッドと呼ぶプログラム片から構成される．ここで，スレッドとは途中で中断されることなく実行される命令列であり，コンテクストスイッチの単位である．

FMD では同一インスタンスから送出される同じ宛先へのメッセージ間の順序性が保証されていることを

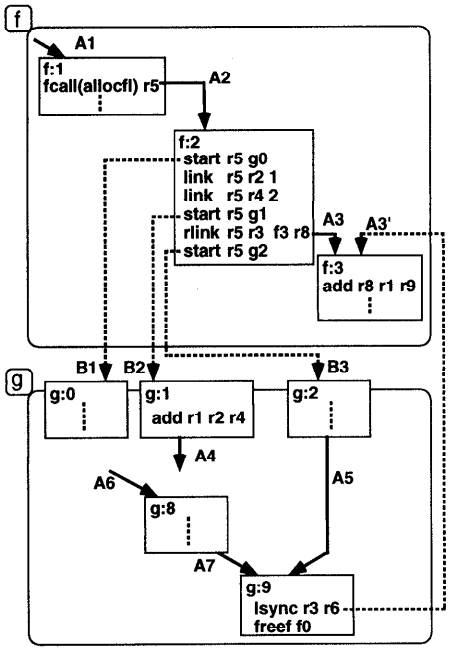


図3 FMD における関数適用
Fig. 3 Function call in FMD.

利用して引数渡しを簡略化している☆.

まず，関数適用を行うインスタンス（caller）は，fcall 命令によりフレーム獲得メッセージを発行する．次に caller は獲得されたフレーム（callee）に対して start 命令によりスレッド起動メッセージを送り初期化スレッド g0 を起動する．次に caller はリモートメモリ書き込みにより引数および関数の返り値の戻り先を callee に送出した後，start 命令によりスレッド g1 を起動する．

スレッドの同期もメッセージにより実現される．各スレッドはスレッド内でメッセージを発行することにより後続のスレッドを起動する．複数の入力を持つスレッドはすべてのメッセージがそろった時点で起動される．このように，スレッド間のメッセージの依存関係に従ってスレッドが順次活性化され，プログラムの実行が行われる．

2.4 細粒度メッセージプロセッサ（FMP）

FMP（Fine-grain Message Processor）は FMD を実現するハードウェアであり，細粒度のメッセージの送受信，スレッドの同期，リモートメモリアクセスおよびインスタンスフレーム管理を行う．

FMP はメモリマップされたデバイスとして扱われ，

☆ KUMP/D の PE 間ネットワークはパケットの順序性を保存するように設計している[1]．

frame (24 bit)	IP (32 bit)	count (8 bit)

図4 同期セル
Fig. 4 Synchronization cell.

CPU は FMP 命令をメモリ書き込みもしくはメモリ読み出しによって FMP に対して発行する．

FMP から CPU への通信は 2 次キャッシュ上への書き込みと後述するスレッドの起動によって行われる．まず，FMP は CPU が発行した FMP 命令や PE 間ネットワークから取り込んだ他 PE からのメッセージをデコードしてメモリに対する操作を行う．続いて FMP は該当するスレッドを起動し，CPU はスレッド中に明示的に埋め込まれた CPU のロード命令によってこれを読み込む．

この 2 次キャッシュでは FMP によってのみ使用される同期のためのカウンタや，インスタンスフレーム管理のための作業領域を，CPU の使用する領域とはできる限り分離し，CPU と FMP のキャッシュの一貫性維持を簡略化している．以下に FMP の各機能とその実装について述べる．

2.4.1 同期制御

KUMP/D でのスレッドの同期制御には同期セルと呼ばれるメモリセルが用いられる．図4 に同期セルを示す．同期セルは FMP によって FMP メモリ上にとられ，フレームアドレス（frame），スレッド開始アドレス（IP），同期カウンタ（count）からなる．

各同期セルは関数呼び出しのときに初期化スレッド中で獲得され，同期カウンタ等がセットされる．獲得された同期セルの番号は，SYNC メッセージによる同期制御に使用されるため，関数の呼び出し側（caller）に送られる．

スレッド間の引数渡しと同期は次のように行われる．CPU は引数を送信し，続いて後述する SYNC 命令を発行する．

PE 内で同期が行われる場合は，FMP は CPU からの SYNC 命令を受け取り，同期セル中のカウンタを減算しスレッドの同期制御を行う．同期成功時，すなわちスレッド実行に必要な引数がそろった時点で FMP は同期セルを解放し，同期セル内のフレームアドレス（framc）とスレッド開始アドレス（IP）により指定されるスレッドを Thread Queue と呼ぶ実行待ち行列に入れる．CPU は各スレッドの終了時に待ち行列を参照し，新たなスレッドを実行する．

他 PE で同期が行われる場合は CPU が発行した SYNC 命令は FMP によって PE 間ネットワークへ

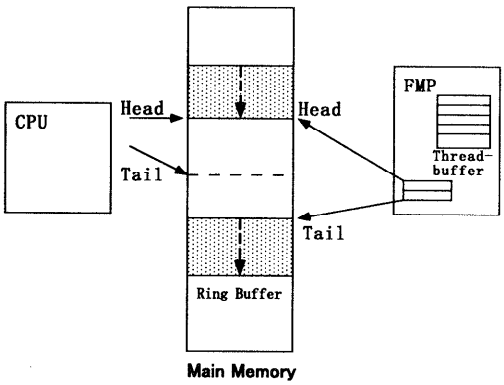


図5 実行可能スレッド待ち行列
Fig. 5 Thread Queue.

メッセージとして送信される．受取り側のFMPでは同様に同期セルのチェックが行われ，同期が成功した場合には受取り側PEのスレッドが起動される．

2.4.2 スレッド待ち行列（Thread Queue）

図5に実行可能スレッド待ち行列（Thread Queue）を示す．Thread QueueはCPUとFMPの双方によって管理され，CPUはThread Queueの先頭部分を管理し，FMPはThread Queueの後尾に新たな実行可能なスレッドをラインサイズ単位で追加する．CPUはThread Queueが空であると判断した場合にのみFMPへ現在のThread Queueの状態を問い合わせることによって，オーバヘッドの大きい割込みを用いずに，実行可能なスレッドの番号を取得する．

このように実行可能なスレッドが十分ある場合には，CPU側のキャッシュのラインの無効化と待ち行列の読み込みが，ラインサイズごとに行われるため，毎回FMPからCPU側のポインタを更新するのに比べてバスのトラフィックを減らすことができる．

2.4.3 インスタンスフレーム管理

KUMP/Dでは，メモリ管理のオーバヘッドを削減するために，比較的小さなインスタンスフレームについてはFMPで固定長のインスタンスフレームを管理する**図6**に示すように，FMPは数種☆のフレームの先頭アドレス（フレームポインタ）を示すスタックを管理し，フレームの割当て/解放はFMPのスタック操作で行う．

これらのフレームスタックはシステム起動時にシステム管理ルーチンによってメインメモリ上に作成され，スタックへのポインタがFMPに渡される．さらに，フレーム割当て/解放を高速化し，メインメモリへのアクセスを削減するためにFMP内に高速なバッファが設けられており，FMPは自動的にメインメモリ上のフレームスタックから，フレームポインタを転送する．

これらの固定サイズのフレームより大きなフレームの割当てが必要な場合にはCPUへ割込みを発生し，システム管理ルーチンによってフレームが獲得される．

☆ 試作機では2のべき乗サイズのフレームを4種類管理することができる．

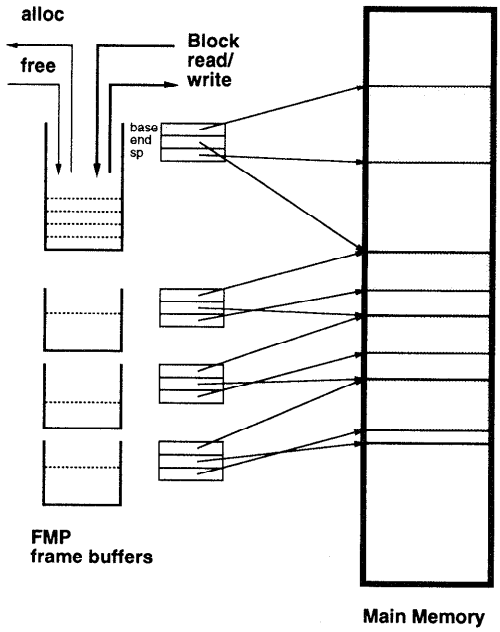


図6 インスタンスフレーム管理
Fig. 6 Instance frame stuck.

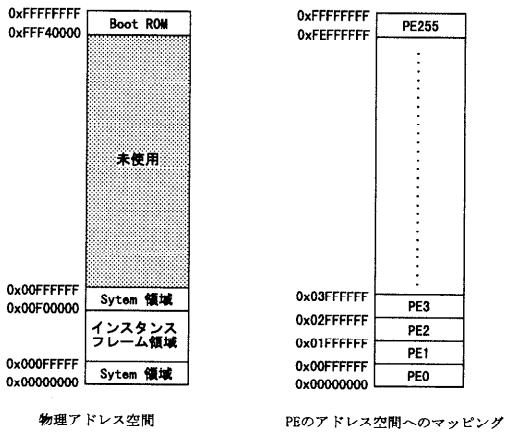


図7 メモリマップ
Fig. 7 Memory map.

2.4.4 メモリマップ

KUMP/Dでは，**図7**のようにすべてのプロセッサのメインメモリ領域に対し，グローバルなメモリアドレス空間を提供する．

FMPではメッセージ中のデータの書き込み先のメ

モリセルのアドレスを明示的に指定する．このアドレス指定には以下の2種類がある（図8参照）．

- 直接指定

  KUMP/Dは32 bitのメモリアドレス空間を持ち，32 bitのアドレスの上位8ビットがプロセッサ番号（PE）に相当し，下位24 bitがプロセッサ内のメモリアドレスオフセット（cell-addr）となる．

- 同期セル間接指定

  同期セル間接指定は，メモリセルの指定に同期セルのアドレスとフレーム内オフセット値により指定する方式である．同期セルのアドレスはプロセッサ番号（PE，8 bit）とプロセッサ内の同期セル番号（scell-addr，16 bit）からなる．これにより，いったん，同期セルをアクセスし，その中で指定されているフレーム領域の開始アドレスを参照する．このフレーム領域の開始アドレスにフレーム内オフセット値（offset, 8 bit）を加えたアドレスが目的のアドレスとなる．間接指定は主に同期をともなうスレッド起動の際に用いられる．

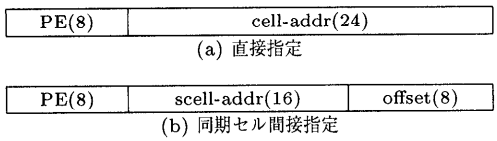


図8 アドレス指定
Fig. 8 Addressing mode.

KUMP/Dでのメッセージには上記いずれかのアドレスが宛先として指定される．双方ともにプロセッサ番号を含み，それによりネットワークのルーティングが行われる．

上位8 bitをPEの指定に使用しているため，PE数は256台までに制限されるが，将来のアドレス空間の大きいプロセッサではこの問題は緩和される．

2.4.5 FMP命令

表1 にFMP命令の一覧を示す．FMP命令は大きく分けて，FMPレジスタ操作，同期セル操作，メッセージ発行，ローカルフレーム割付け，の4種からなる．表中 *Rs* はCPUのレジスタ名を示し，FRaと

表1 FMP命令
Table 1 Instruction set of FMP.

命令	operand	アドレス指定形式	機能，メッセージ形式 < OP,ADDR,DATA >
FMPレジスタ操作			
SETFRA	*Rs*	−	FRaへの書き込み
READFRA	*Rs*	−	FRaの読み出し
SETFRS	*Rs*	−	FRsへの書き込み
READFRS	*Rs*	−	FRsの読み出し
同期セル操作			
ALLOCSCELL	*Rs*, *count*	−	同期セルの獲得 ([*Rs*]=Frame) (FRs:=獲得された同期セルアドレス)
SETSCELLIP	*Rs*	−	同期セルのIP設定 ([*Rs*]=IP)
メッセージ発行			
LINK	*Rs*, *offset*	直接	LINKメッセージ発行 ([*Rs*] = *data*) < LINK, [FRa]+*offset*,[*Rs*] >
RLINK	*Rs*, *offset*	直接	Return先情報のLINKメッセージ発行 < LINK,[FRa]+*offset*, ([FRs],[*Rs*]) >
START	*Rs*	直接	START メッセージ発行 ([*Rs*]=IP) < START, [FRa], [*Rs*] >
SYNC	*Rs*	−	SYNC メッセージ発行 (*Rs*=synccell) < SYNC, [*Rs*],− >
LSYNC	*Rs*	同期セル間接	LSYNC メッセージ発行 < LSYNC, [FRa], [*Rs*] >
その他			
ALLOCFL	*Rs*	直接	ローカルフレーム獲得 (*Rs*=size) ([FRa]:=獲得されたフレームアドレス)
FREEFL	*Rs*	直接	ローカルフレーム解放 (*Rs*=size) ([FRa]=解放するフレームアドレス)

Rs はCPUのレジスタ名を示し，FRaとFRsはFMPレジスタを表す．また[register名]は各々のレジスタの内容を表す．

表2 FMP レジスタ
Table 2 FMP register.

記号	レジスタ名	用途
FRa	アドレスレジスタ	アドレスデータを格納する．メッセージのアドレスフィールド指定用．
FRs	同期セルレジスタ	同期セル番号を格納する．メッセージの同期セルフィールド指定用．

FRs は特定の物理アドレスにマッピングされた FMP レジスタを表す（**表2** 参照）．*offset*, *count* は FMP 命令のオペランドデータ部で与える定数を表し，アセンブラ中では即値もしくはラベルが使用される．また，インスタンスフレームを指定する命令については，アドレス指定形式を示す．

KUMP/D では，FMP の設計を単純化するため，各種メッセージ発行に必要なパラメータの一部をあらかじめ FMP 内部のレジスタにセットしておく設計とした．FMP 命令は以下のように CPU によって発行される．

(1) FMP レジスタの値を使用する命令を発行する場合には，CPU はまず SETFRA 命令もしくは SETFRS 命令によって FMP レジスタに値をセットする．使用する CPU のレジスタ *Rs* にはあらかじめ値をセットしておく．

(2) LINK, ALLOCFL など動作を指定する命令を発行する．使用するレジスタ *Rs* にはあらかじめ同様に値をセットしておく．

(3) インスタンスフレーム獲得など，FMP のレジスタを読む必要がある場合は READFRA や READFRS で FMP レジスタの値を読む．

なお，前の命令と同一の値が使用できる場合には FMP レジスタの値の設定を省略することができる．

以下では主要な命令についてその用途と動作を述べる．

同期セルの獲得，同期セルへのスレッド開始アドレス（IP）の設定命令は以下のものがある．

- ALLOCSCELL *Rs*, *count*
  新たな同期セルを獲得し，そのセルのアドレスを FRs レジスタに入れる．同時に獲得した同期セルの frame 部に *Rs* の値を，*count* 部にオペランドで与えた *count* の値をセットする．
- SETSCELLIP *Rs*
  FRs レジスタの指す同期セルの IP 部に *Rs* の値を書き込む．

メッセージ発行命令には以下のものがある．

- LINK *Rs*, *offset*
  LINK 命令は関数引数渡しの LINK メッセージを発行する．
- RLINK *Rs*, *offset*
  RLINK 命令は関数結果を受け取るアドレスを送るための LINK メッセージ発行を行う．送り先のアドレス指定には直接指定形式が用いられ，メッセージのデータ部には FRs レジスタの値（同期セルアドレス）と [*Rs*] で表されるオフセットで示される同期セル間接メモリアドレスがセットされる．
- START *Rs*
  START 命令はスレッド起動を行う START メッセージを発行する．FRa レジスタの値によってフレームアドレスを指定し，*Rs* の値によって IP アドレスを指定する．
- SYNC *Rs*
  SYNC 命令は SYNC メッセージ発行を行う．SYNC メッセージのアドレス部には *Rs* により与えられる同期セルアドレスがセットされる．
- LSYNC *Rs*
  LSYNC 命令は LSYNC メッセージの発行を行う．通常 LSYNC 命令は関数結果値を呼び出しインスタンスへ送るために用いられ，メッセージのアドレス部は RLINK 命令によって送られた同期セル間接指定のアドレスが使用される．このアドレスはあらかじめ FRa にセットして用いられ，データ部には *Rs* の値がセットされる．

ローカル（同一プロセッサ内）フレーム割付け，解放命令は以下のものがある．

- ALLOCFL *Rs*
  ローカルフレームを獲得し，獲得したフレームを FRa にセットする．*Rs* にはサイズを指定する．
- FREEFL *Rs*
  ローカルフレーム解放を行う．

2.5 PE 間ネットワーク

スケーラビリティを重視して我々は2次元トーラス結合の PE 間ネットワークを採用した[7]．また，2次元トーラス結合は画像の PE へのマッピングにも適している．

前述したアドレス指定の制限から試作機の最大構成は256台で，X–Y ルーティングを用いている．すべてのパケットは72 bit 固定長であり，PE 間ネットワークはこの短いパケットのために最適化されている．

図2に示すように，各 PE は FMP を介して PE 間ネットワークに結合されており，CPU は FMP 命令を発行することによって PE 間通信を行う．FMP は外

部からのパケットをデコードして同期制御，メモリ割当て，リモートメモリアクセス等を行う．ネットワークの混雑によりシステム全体がストールするのを避けるため，これらのFMPの処理はCPUよりも優先して行われる．

X-Yルーティングでは，宛先PEによってルーティングが一意に決定されるため，パケットの順序性は保証される．また，ルーティングされるパスごとにパケットバッファを持つことにより，デッドロックフリーを実現している．各々のパケットバッファの大きさは最小に設定されており，すべてのパスのためのバッファを設けても全バッファ容量は小さく，1 PEあたり9 Kbyteである[1]．

2.6 I/O ネットワーク

マルチメディアアプリケーションでは，I/Oの設計は重要である．筆者らは，マルチメディアアプリケーションのI/O処理の特徴[8]を考慮してI/Oシステムを設計した．KUMP/DのPE間ネットワークは短いパケットのために最適化されているため，I/Oパケットを流すと様々な問題を生じる．すなわち，

- 新たにI/Oパケット用のバンド幅が必要になる．特に動画像用のバンド幅が問題である．
- PE内に十分なパケットバッファを用意できないため，長いパケットは2つ以上のPEを渡る長いパスを占有することになり，デッドロックの原因となる．
- 動画像の入出力のために，一定のバンド幅を保証する必要があり，プライオリティ制御が必要になり，プロトコルやハードウェアが複雑になる．

これらの問題を解決するためKUMP/DではI/O専用のネットワークを設けた．このI/Oネットワークでは動画像入出力のためのバンド幅を確保するとともに，画像入出力とPE上の処理との同期を行う．

I/Oネットワークは2次元トーラス状のPE間ネットワークと平行に張られたリング状のネットワークからなる．これらのリングは高速なシリアルラインを用いて構成されている．

各々のPEではI/O処理はIOPとFMPによって制御される．IOPは図2中に示されたV-RAM上にとられたI/O用のバッファにI/Oパケットを読み書きし，FMPにSTARTメッセージを送ることによって後続する処理を起動する．

2.6.1 固定長パケットによる同期

MIMD型の並列計算機では，各々のPEの処理は非同期に行われるため，リアルタイム性が必要なマルチメディアアプリケーションでは各PEとビデオフレームとの同期をとる必要がある[1]．KUMP/Dでは動画像の入出力を行う外部ビデオコントローラ（EVC）を設け，EVCから常時固定長パケットを流すことによって同期を行っている．リアルタイム性を必要とする通信はすべてこの固定長パケットを用いたI/Oネットワークを用いて行われる．

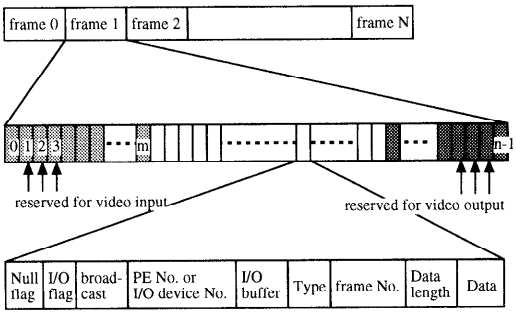


図9 Video Frame の分割
Fig. 9 Time slicing in video frame.

図9に示すように，各ビデオフレームはn個の固定長のパケットに分割され，ビデオデータの転送のバンド幅を確保するため，はじめのm個のパケットが画像入力に，最後のm個が画像出力に予約される．その他のパケットは，disk I/O等に使用可能である．この固定長のパケットは，常時I/Oネットワークを流れており，パケットの転送タイミングをビデオフレームと処理との同期に使用することができる．この同期はI/OパケットのIOPからFMPへSTARTメッセージを送信し，スレッドを起動することで行う．

さらに，各パケットヘッダ部にはビデオフレーム番号を付加し，これをIOP中に保持して，各ビデオフレーム間に行う処理の開始時に取得したビデオフレーム番号と比較することによって処理の遅れをチェックすることができる．処理の遅れが検出された場合は次のビデオフレームの処理を省略するなどして遅れを修正する（詳細は文献1)を参照）．

3. 試作機の仕様

現在我々は16 PEからなる試作機を製作中である．この試作機は小規模ではあるが提案方式の有効性を検証し，かつソフトウェア開発を行うためのテストベッドとして用いることを目的としている．

この試作機の仕様を**表3**に示す．各々のPEは66 MHzのPentiumを用いて構成しており，FMPとネットワークルータおよびIOPはFPGAを用いている．

PE間ネットワークの各リンクは24 bit幅であり，パ

表3 試作機の仕様
Table 3 Specification of prototype.

PE数	16（4×4構成）
PE間ネットワークの構造	2次元トーラス
CPU	Pentium（66 MHz）
PEあたりの記憶容量	8 [Mbyte]
PE間ネットワークの転送速度	99 [Mbyte/sec]
I/Oネットワークの構造	リング状
I/Oネットワークの転送速度	33 [Mbyte/sec]
対応するビデオシステム	NTSC

ケット長は72 bit固定で，PE間ネットワークのクロックは33 MHzである．この試作機では，現在NTSC（640×480 pixel/frame, 30 frame/sec）信号のみを扱う仕様になっており，1フレームの画像は900 kbyteである．各I/Oリンクの速度は33 Mbyte/secであるから動画像データによるI/Oリンクの占有率VOR（Video Occupation Rate）は0.41程度となる．ビデオパケットを除いたI/Oネットワークの通信速度は19.5 Mbyte/secである．

4. 性能評価

ここでは，まずKUMP/DとDatarol-IIのメッセージの処理能力を比較する．細粒度並列計算機ではメッセージが多く発生するためメッセージハンドリング能力について考察することは重要である．Datarol-IIプロセッサはメッセージ処理を高速化することに重点を置いて設計されており，これをメッセージ処理における指標として考えることができる[3),9)]．

4.1 予備評価

KUMP/Dプロセッサではメモリアクセスとメッセージ発行のためのFMP命令発行の双方で使用されるメモリバスが最もボトルネックとなりやすい．そこで，KUMP/Dプロセッサ，Datarol-IIプロセッサそれぞれについて，以下にあげる3つの典型的なメッセージ処理とスレッドの起動について，メモリバスの使用頻度に着目し，双方のメッセージ処理能力を比較する．

（1）**関数起動**

Datarol-IIにおけるcall命令はKUMP/Dプロセッサではフレーム獲得および初期化スレッドの起動に相当し，以下の5命令となる．

allocscell, setscellip } 同期セルの設定
allocfl } フレーム獲得メッセージ
setfra, start } 初期化スレッド起動メッセージ

（2）**関数引数の授受**

Datarol-IIでは関数引数の授受はlink，receiveの2命令を用いる．一方，KUMP/Dでは以下の3命令となる．

setfra, link } 引数データの書き込みメッセージ
start } エントリスレッド起動メッセージ

（3）**関数結果値の授受**

Datarol-IIでは関数結果値の授受はrlink，receive，returnの3命令を用いる．KUMP/Dでは以下の6命令に相当する．

allocscell, setscellip } 同期セルの設定
setfra, rlink } rlink（link）メッセージ
setfra, lsync } リターン（lsync）メッセージ

Datarol-IIプロセッサではFU（Function Unit）でcontinuation情報設定のためのメモリ書き込み（W_D）が，またCU（Communication Unit）でのパケット受理時にcontinuationの読み出し（R_D）とパケットの書き込み（W_D）が発生する．

KUMP/Dプロセッサでは，CPUがFMP命令発行（I_K）のためメモリバスを使用する．また，FMPはメッセージ受理時に内容をメモリへ書き込む（W_K）．

ここでは，KUMP/Dプロセッサのクロック（66 MHz）を基準にそれぞれのメモリバス使用時間を算定する．FMPの命令受付け部は2クロックを要し，2次キャッシュにアクセスタイム15 ns程度のS-RAMを使用すると，KUMP/Dのメモリアクセス時間I_K, W_Kは2クロックとなる．Datarol-IIプロセッサは専用設計であり高いクロック速度が望めないため，クロック速度をPentiumの半分，すなわち基本サイクルを2クロックと設定する．Datarol-IIプロセッサでは各基本サイクルに1回のRegister Buffer☆アクセスができるので，$R_D = W_D = 2$クロックである．

上記条件でのそれぞれのプロセッサのバスの使用回数を発行される命令に従って求めると，**表4**と**表5**のようになる．またここから必要クロック数を求めたものを**表6**に示す．2引数1結果値の関数適用の場合，Datarol-IIプロセッサでは30クロック，KUMP/Dでは44クロックとなる．

さらに，KUMP/Dではスレッドの起動にバスを使用するため，これを考慮する．スレッド待ち行列からのロードに1次キャッシュのミスヒットを考えると

☆ Datarol-IIプロセッサは2次キャッシュに相当するRegister Bufferと呼ぶ高速メモリを持っている．

表4 Datarol-II プロセッサのメモリバス使用回数
Table 4 Number of bus cycles (Datarol-II).

バスサイクル	FU	CU	計
関数起動	$1W_D$	$1R_D+1W_D$	$1R_D+2W_D$
関数引数の授受	$1W_D$	$1R_D+1W_D$	$1R_D+2W_D$
関数結果値の授受	$2W_D$	$2R_D+2W_D$	$2R_D+4W_D$

ただし，R_D は RB からの読み出しを，W_D は RB への書き込みアクセスを表す．

表5 KUMP/D プロセッサのメモリバス使用回数
Table 5 Number of bus cycles (KUMP/D).

バスサイクル	CPU	FMP	計
関数起動	$5I_K$	$1W_K$	$5I_K+1W_K$
関数引数の授受	$3I_K$	$1W_K$	$3I_K+1W_K$
関数結果値の授受	$6I_K$	$2W_K$	$6I_K+2W_K$

ただし，I_K は CPU が FMP 命令発行を行うためのバスサイクル，W_K は FMP からメモリへの書き込みアクセスを表す．

表6 メモリバス占有時間
Table 6 Memory bus transaction time.

	Datarol-II	KUMP/D
関数起動	6	12
関数引数の授受	6	8
関数結果値の授受	12	16

$(I_k+3)/4$☆，FMP へスレッド待ち行列の先頭が消費されたことを示すため I_k のサイクルが必要となる．この関数が2個のスレッドを含むとすると，バスの使用時間は 50 から 51 クロックとなる．

したがって，KUMP/D プロセッサではメモリバスを Datarol-II プロセッサの約 1.7 倍使用する．このとき，メモリバスが実行時間を支配しているとすると，KUMP/D のメッセージ処理能力は Datarol-II の約 0.6 倍となる．

KUMP/D プロセッサはメッセージ処理能力に関して，半分のクロック速度で動作する Datarol-II プロセッサの 0.6 倍程度の性能を持つと予想される．しかし，汎用プロセッサのクロック速度向上は著しく，専用設計とのクロック速度差は今以上に開く傾向があり，メッセージ処理能力はほぼ同等であるともいえる．また，KUMP/D プロセッサでは，計算能力自体は高いクロック速度とスーパスカラの効果により 4 倍程度あり，ピーク性能では Datarol-II プロセッサを上回る性能を持つ．

4.2 シミュレーションによるメッセージ処理能力の評価

ここでは前述のメッセージハンドリングの性能予測を検証するため，フィボナッチ数を再帰的に計算するプログラムと，n-Queen の解を再帰的に求めるプログラムの速度をシミュレートし，Datarol-II と比較する．

検証はハンドコンパイルしたコードを用いて PE 単体で評価を行った．本来，単体プロセッサのみの実行では LINK 命令等は使用せず直接 CPU によってメモリへ書き込むべきであるが，ここではメッセージ処理能力の評価のため，並列実行時と同様に関数展開を行い，引数渡し等もすべて FMP 命令を使用した．

KUMP/D のクロック速度は Datarol-II の 2 倍とし，**図 10** および**図 11** に KUMP/D のクロックを基準とした実行時間の比較を表す．

フィボナッチ数を求めるプログラムは各スレッドの粒度が非常に小さく，メッセージ処理のオーバヘッドが大きいため，メッセージ処理能力の比較に適している．処理時間全体に占めるメッセージ処理の割合から，実行時間の比をほぼそのままメッセージ処理能力の比であると考えると前述した予想どおりの性能となる．

問題サイズが大きいときに性能が逆転しているのは Datarol-II の Register Buffer のラインの入換えのためのオーバヘッドが Pentium より大きいためである．

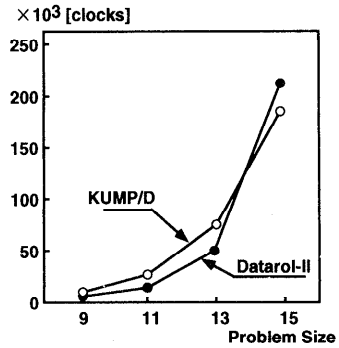


図 10 fibonacci (n) の実行時間
Fig. 10 Execution time of fibonatti (n).

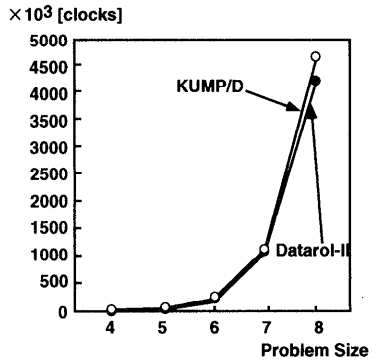


図 11 Queen (n) の実行時間
Fig. 11 Execution time of Queen (n).

☆ スレッド待ち行列は 1 要素 64 bit，Pentium のキャッシュのラインサイズは 32 byte である．待ち行列が十分に長い状態を想定した．

Datarol-II は通常のプロセッサとは異なり，各スレッドが使用するフレームを先行して Register Buffer に読み込む方式をとっているが，この Register Buffer のラインサイズが fibonacci (n) 関数で使用する変数の数に比べて大きいため，オーバヘッドが大きくなる．また，Pentium の外部バスは 64 bit，Datarol-II は 32 bit である．

n-Queen を求めるプログラムではスレッド長がフィボナッチ数を求めるプログラムより長くなるため，図 11 に示すように KUMP/D と Datarol-II の性能差は縮小している．また，関数中で使用する変数も増えるため，Datarol-II においての register Buffer の入換えのオーバヘッドによる性能の逆転は起こっていない．

実際に使用されるアプリケーションではスレッド長がさらに長くなると考えられるため，スレッド切替えのオーバヘッドの割合が小さくなり，KUMP/D は Datarol-II を上回る性能を持つと予想される．

4.3 試作機の性能

試作機では CPU に Pentium（66 MHz）を使用している，また，16 台構成時に NTSC レベルの画像を扱った場合の I/O ネットワークの占有率 VOR（Video Occupation Rate）を I/O リンクの転送速度から算出すると VOR = 0.41 となる．

この条件で，画像処理アプリケーションにおける性能を検討するため，ハンドコンパイルによるコードを用いて実行時間を予測したものを**表 7** に示す．画像は各プロセッサに均一にマッピングされており，1 プロセッサあたりの画素数は 19.2 [kpixel] であるとした．いずれも VOR を考慮し，画像の転送に必要な時間を実行時間に加算している．

KUMP/D のプロセッサエレメントはレイトレーシングなどの非常に重いアプリケーションを除いてリアルタイム処理が可能なレベルにあるといえる．

さらに，PE 間ネットワークと PE の計算能力とのバランスを考察するため，16 プロセッサで FFT を行った場合の性能を算出した．KUMP/D は EVC により画像を自由に PE にマッピングできるため，あらかじめ PE 間の転送が少なくなるように画像を配置することができる．この場合 16 台構成においてはほぼ隣接 PE 間の通信のみで FFT を行うことが可能である．

表 7 プロセッサエレメントの性能
Table 7 Performance of PE.

例題	性能予測値
Ray-tracing	1 M [polygon/sec] 10～20 [objects/frame]
Polygon rendering	60～80 k [polygon/sec]
Canny フィルタ	3.7 k [pixel/frame]
領域分割	21.6 k [pixel/frame]

この条件でハンドコンパイルされたコードを用いて実行時間の評価を行った結果，画素数 512×512 の場合 16 台構成の KUMP/D は 1 画面の FFT に 5 フレーム程度を要した．また，PE の演算能力と PE 間ネットワークの性能はほぼバランスしており，1 要素の演算に必要な時間と 1 要素の PE 間転送に必要な時間がほぼ同等であることを確認した．

以上の性能予測により，KUMP/D は現在製作中の試作機においても NTSC レベルの画像のリアルタイム処理に必要な演算能力に近い性能を持つと思われる．今後 PE 数の拡張と単体のプロセッサの性能向上により高度なマルチメディア処理に必要な性能が得られるものと考えている．

5. 関連研究

KUMP/D には細粒度並列計算機としての側面と画像処理を行う計算機としての側面がある．KUMP/D 同様細粒度のマルチスレッド処理を行う計算機として KUMP/D の基となった Datarol-II[3),9)]，電総研の EM-4[4)] および MIT の *T[2),10),11)] があげられる．また，一般的なプロセッサにメッセージ処理を行うプロセッサを付加するものとして Stanford FLASH[12)]，MIT Alewife[13)] がある．

Datarol-II と EM-4 は専用プロセッサを使用する並列計算機である．Datarol-II プロセッサはメッセージ処理を高速化することに重点を置いて設計されており，ほとんどのメッセージの処理を専用のハードウェアで行い，細粒度処理に特化したパイプラインを持つため，メッセージ処理に関して優れた特徴を持つアーキテクチャの 1 つであると考えられる．

EM-4 は同様に細粒度の並列処理に特化した専用のプロセッサで 2 入力の直接待合せ方式による同期を行っており，多入力の同期を許す Datarol-II よりも自由度が少ないが，高いメッセージ処理能力を持つ．

これらの専用プロセッサを用いる計算機に共通する問題点は専用設計による高い開発コストである．

これに対し，*T では同一種の一般的なマイクロプロセッサを用いてデータ処理用プロセッサとメッセージ処理用のプロセッサを構成している．特殊なハードウェアを使用することなく細粒度並列計算機を構成することができるが，ソフトウェアでメッセージを処理するため，メッセージ処理に時間がかかる．また同期用のカウンタの領域をメインプロセッサと共有しているため，キャッシュの一貫性維持のためのバストラヒッ

クが増加する傾向がある．

これらの中間に位置するものとしては，メッセージ処理のためのプロセッサを付加するFLASH，Alewifeが参考となる．これらは，キャッシュコヒーレント制御とメッセージ処理を行うプロセッサを付加している．しかし，いずれもメッセージ処理にはtrap handlerを用いているため細粒度のメッセージに対してはオーバヘッドが大きい．また，多入力の待合せや，メモリ管理の補助などは実装されていない．他の問題点としては共有メモリモデルを提供するなどの機能が多いため付加プロセッサのハードウェアが大きくなりやすいことがあげられる．

これに対しKUMP/Dは，細粒度メッセージ処理に特化し，汎用プロセッサに最小限のハードウェアを付加することによってDatarol-IIに近いメッセージハンドリング性能を持った計算機を短期にかつ安価に実現している．

マルチメディア指向計算機としてのKUMP/Dの特徴はMIMD型の並列計算機であり，かつ実時間処理のための動画像との同期機構を持つ点である．画像処理においてはフィルタ処理などのlow-level処理にSIMD型が向くため，特に実時間処理を行うものではSIMD型の並列計算機が主流であったが，知識処理などを含むhigh-level処理にはMIMD型が適している．このため，SIMD型とMIMD型を結合した計算機が提案されている[14),15)]．

これらの計算機ではSIMDとMIMDを結合し，MIMD部でhigh-level処理を行うことを目的としている．しかし，SIMD部が専用設計であるうえに，SIMD部に画像を入出力するためのI/O，SIMD内の通信用ネットワーク，MIMD用のネットワークを別個に持つ複雑な構成となっているため，ハードウェアコストが増大している．

これに対しKUMP/Dでは，汎用マイクロプロセッサの性能が著しく向上しているため，従来SIMD部で行っていたlow-level処理に関してもMIMD型で十分な性能を得られると考え，MIMD型のみで構成した場合に問題となる動画像との同期をI/O専用のネットワークとFMPによるスレッドの実行制御によって実現した．これにより，高度な画像処理を行う並列計算機を汎用マイクロプロセッサを用いたPEと単純な構成のネットワークで構成し，ハードウェアコストの増大を避けることができる．

6. おわりに

高度なマルチメディア処理に必要となる演算能力を実現するため，自由度の高いMIMD型並列計算機KUMP/Dを提案し，その性能予測を行った．

KUMP/Dは並列処理において問題となるリモートメモリアクセス等によって生じるレイテンシを隠蔽するため，細粒度並列処理を実行の基本とする．各プロセッサエレメントは汎用マイクロプロセッサに細粒度のメッセージを処理するハードウェアを付加することにより，高いメッセージ処理能力を低コストで実現することを可能にしている．

また，I/O専用のネットワークを持ち固定長のパケットを用いることによって，動画像の入出力に必要なバンド幅を確保するとともに，MIMD型並列計算機で問題となる動画像と処理との同期を実現している．

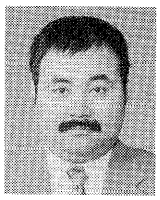
現在，16プロセッサからなる試作機を製作中であり，今後この試作機上で実際のマルチメディアアプリケーションについて評価を行っていく予定である．

参考文献

1) Tomiyasu, H., Kawano, T., Taniguchi, R. and Amamiya, M.: KUMP/D: The Kyushu University Multi-medea Processor, *Proc. Computer Architectures for Machine Perception '95*, pp.367–374 (1995).
2) Nikhil, R.S., Papadopoulos, G.M. and Arvind: *T: A Multithread Massively Parallel Architecture, *Proc. 19th ISCA*, pp.156–167 (1992).
3) Kawano, T., Kusakabe, S., Taniguchi, R. and Amamiya, M.: Fine-grain Multi-thread Processor Architecture for Massively Parallel Processing, *Proc. HPCA '95*, pp.308–317 (1995).
4) 児玉，坂井．山口：データ駆動型シングルチッププロセッサEMC-Rの動作原理と実装，情報処理学会論文誌，Vol.32, No.7, pp.849–858 (1991).
5) Amamiya, M. and Taniguchi, R.: Datarol: A Massively Parallel Architecture for Functional Language, *Proc. SPDP*, pp.726–735 (1990).
6) 川野，日下部，谷口，雨宮：細粒度マルチスレッド処理向けプロセッサDatarol-IIプロセッサの構成とその評価，情報処理学会論文誌，Vol.36, No.7, pp.1700–1708 (1995).
7) Dally, W.J.: Performance Analysis of k-ary n-cube Interconnection Networks, *IEEE Trans. Comput.*, Vol.39, No.6, pp.775–785 (1989).
8) Reddy, A.L.N. and Wyllie, J.C.: I/O Issues in a Multimedia System, *IEEE Computer*, pp.69–74 (1994).
9) 川野，日下部，谷口，雨宮：細粒度スレッド処理のためのコンテクストスイッチ機構—並列計算機 Datarol-II の階層メモリシステム，*Proc. JSPP94*, pp.81–88 (1994).
10) Chiou, D., Ang, B.S., Greiner, R., Arvind,

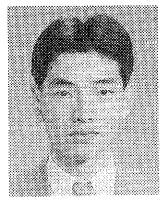
Hoe, J.C., Beckerle, M.J., Hicks, J.E. and Boughton, A.: StarT-NG: Delivering Seamless Parallel Computing, *Proc. Euro-Par'95*, pp.101–116 (1995).
11) Ang, B.S., Chiou, D., Rudolph, L. and Arvind: Message Passing Support in StarT-Voyager, MIT Laboratory for Computer Science, CSG Memo 387 (1996).
12) Kuskin, J., Ofelt, D., Heinrich, M., Heinliein, J., Simoni, R., Gharachorloo, K., Chapin, J., Nakahira, D., Baxter, J., Horowitz, M., Gupta, A., Rosenblum, M., and Hennessy, J.: The Stanford FLASH Multiprocessor, *Proc. 21st International Symposium on Computer Architecture*, pp.302–313 (1994).
13) Agarwal, A., Kubiatowicz, J., Kranz, D., Lim, B.-H., Yeung, D., D'Souza, G., and Parkin, M.: Sparcle: An Evolutionary Processor Design for Large-Scale Multiprocessors, *IEEE Micro*, pp.48–61 (1993).
14) Jonker, P.P.: An SIMD-MIMD architecture for image processing and pattern recognition, *Proc. Computer Architectures for Machine Perception '93*, pp.222–230 (1993).
15) Weems Jr., C.C.: The Second Generation Image Understanding Architecture, *Proc. Computer Architectures for Machine Perception '93*, pp. 276–285 (1993).

(平成8年9月24日受付)
(平成9年2月 5 日採録)



富安 洋史（正会員）

昭和41年生．1989年九州大学電気工学科卒業．1991年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．同年日本電気C & Cシステム研究所入所．1993年九州大学大学院総合理工学研究科情報システム学専攻博士課程入学．1996年より九州大学大学院システム情報科学研究科助手．並列計算機アーキテクチャの研究に従事．

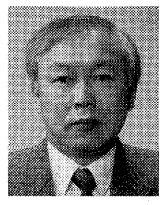


川野 哲生（正会員）

1968年生．1991年熊本大学電気情報工学科卒業．1993年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．1996年同博士後期課程修了．同年日本電信電話株式会社ソフトウェア研究所入所．工学博士．並列計算機アーキテクチャ，高速プロトコル処理の研究に従事．電子情報通信学会会員．

谷口倫一郎（正会員）

1980年九州大学大学院工学研究科修士課程修了．同年同大学院総合理工学研究科助手．平成元年同助教授．平成8年同大学院システム情報科学研究科教授．工学博士．画像理解，並列処理，画像処理システムに関する研究に従事．電子情報通信学会篠原記念学術奨励賞，本会論文賞，坂井記念特別賞授賞．人工知能学会，電子情報通信学会，映像情報メディア学会会員．



雨宮 真人（正会員）

1942年生．1967年九州大学工学部電子工学科卒業．1969年同大学大学院工学研究科修士課程修了．同年日本電信電話公社武蔵野電気通信研究所入所．以来，プログラミング言語・処理系，自然言語理解，データフロー・アーキテクチャ，並列処理，関数型/論理型言語，知能処理アーキテクチャ，等の研究に従事．現在九州大学大学院システム情報科学研究科知能処理システム学専攻教授．工学博士．電子情報通信学会，ソフトウェア科学会，人工知能学会，IEEE，ACM，AAAI会員．