解　説

マルチプロセッサ用高性能システム・バス†

堀　川　　隆††

1. はじめに

計算機システムの性能向上はめざましく，ワークステーションの性能は年率2倍で向上していくとの予測もある．これは，RISC の採用やマルチプロセッサ化などのアーキテクチャ的な工夫と，集積化技術の発展によるプロセッサ動作周波数の向上や内蔵キャッシュ容量増大によるものである．

このように性能が向上したプロセッサを活用して，システムとしての性能を向上させるためには，プロセッサの性能に見合った速度で命令やデータを供給しなければならない．したがって，この役割を果たすバスの性能が重要となってきている．特に，マルチプロセッサ・システムでは，複数のプロセッサに命令・データを供給する必要があるため，1プロセッサのシステムよりも，バス性能に対する要求は大きい．また，マルチプロセッサ・アーキテクチャを効率的にサポートするための機能もバスに必要とされるようになってきている．

本稿では，高性能マルチプロセッサ・システムに用いられるバスの技術動向について，主に，Futurebus＋ を例に述べる．

2. バスの概要

2.1 目　的

計算機システムのデータ転送路であるバスの目的は，次の2点（図-1）．

目的 1)　バスに接続されている任意のモジュール間でのデータ転送を，可能なかぎり高速に行うこと，

目的 2)　モジュールをバスに接続するためのインタフェース，および，データ転送の方法を規格化して，システム構築に必要なモジュールの組合せを容易にすること，

であり，これらを両立させることが理想である．しかし，次の三つの理由から，上の二つの目的は相反することが多い．

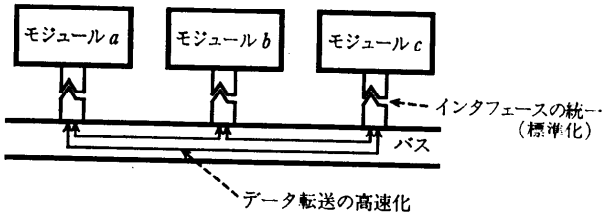


図-1　バスの目的（データ転送の高速化と標準化）

1)　規格にはないが，ユーザの応用分野において有用な機能を追加することで，性能の優れたバスを実現できる[1]．

2)　規格に含まれる機能のうち，利用しないものを削除することにより，バスのコストを押さえることができる．

3)　定められた規格が一般に広まるのには時間がかかるため，この間の技術革新を取り入れた独自仕様のバスを設定することにより，規格化されたバスよりも高性能なバスを実現できる．

どちらの目的を重視するかは，次節で述べるように，バスの種類に依存するようである．

バスの規格では，1)各ハードウェア信号線の役割やトランザクションのプロトコルを定めた論理的な仕様と，2)コネクタの形状，個々のハードウェア信号の物理的なレベル，電流特性を規定した物理的な仕様，の両者を定める必要がある．たとえば，Futurebus＋ では，P 896.1[2] で論理的な仕様，P 896.2[3] で物理的な仕様を規定している．

2.2 バスの種類

バスは，それに接続されるモジュールの規模に応じて，下記のとおりに分類できる．

1)　ALU，レジスタなどプロセッサの構成要素

† High Performance Multiprocessor System Bus by Takashi HORIKAWA (NEC Corporation, C&C Systems Research Laboratories).

†† 日本電気(株) C&C システム研究所

間でデータ転送を行うための内部バス

2) プロセッサとキャッシュ・メモリ間で命令やオペランドを転送するためのプロセッサ・バス

3) CPUボード，主記憶，入出力制御部の間でデータ転送を行うためのシステム・バス

4) 各種入出力装置と計算機システム間でデータを転送するための入出力バス

下になるほど，バスに接続されるモジュールの規模は大きくなる．一般に，バスで接続されるモジュールの規模が小さいと高速性（目的1）が重視され，大きくなるほど標準化（目的2）が重視されるようである．

IEEE などの機関において標準化の対象となるのは，主に，3) のシステム・バスと 4) の入出力バスである．前者の代表例として，VME バス[4]，MultibusII (PSB バス)[5]，EISA (Extended Industry Standard Architecture)[6]，NESA (New Extended Standard Architecture)[7]，マイクロ・チャネル (Micro Channel)[8]，NuBus[9]，Futurebus[10),18]，Futurebus＋[2),3),11]，があり，後者の代表例として，SCSI[12]，GP-IB[13]，IPI[14]，がある．

2.3 バスのハードウェア構成

システム・バスは，ハードウェアとしてみると，複数の信号線から構成されるデータ転送路である．これらの信号線は，その目的により，下記の3種類に大別できる．

1) アドレス・バス：データ転送の相手を指定するアドレスを転送するための信号線．

2) データ・バス ：モジュール間で受け渡したいデータを転送するための信号線．

3) 制御信号 ：アドレスやデータを転送するタイミングや転送方向を指定するための信号線．

アドレス・バスとデータ・バスが分離されておらず，同じハードウェア信号線を時分割で用いてアドレスとデータを転送する方式のシステム・バスもある．

2.4 データ転送方式

一般に，システム・バスを介したデータ転送は，下記の4フェーズから構成されている．

フェーズ1) データ転送を要求するモジュール (マスタと呼ぶ) がバスの使用権を獲得する (Arbitration phase)

フェーズ2) データ転送の相手（スレーブと呼ぶ）を指定するアドレスをバスを通して送る (Connection phase)

フェーズ3) マスタとスレーブ間でデータの転送を行う (Data transfer phase)

フェーズ4) バスの使用を終了する (Disconnection phase)

各フェーズの実現方法（ハードウェア信号の動作）は，バスにより異なる．これは，できるかぎり高速・効率的なデータ転送を実現するための技術的な工夫・改良が各バスごとに行われてきたためである．なお，( )内には，Futurebus＋ での呼び方を示した．

2.4.1 同期，非同期のデータ転送

モジュール間でのアドレスやデータの受け渡し方法には，下記の2種類がある．

1) 同期転送方式

各モジュールが，バスから供給されるクロックに同期してデータの送受を行う方式．

2) 非同期転送方式

データの送出や受取を示すハンドシェーク信号により，マスタとスレーブ間で互いに状態を確認しあってデータを転送する方式．

同期転送方式のバスに，クロックを必要とするモジュール (CPU を搭載したモジュールなど) を接続する場合，バスを駆動するクロックとモジュールを駆動するクロックは通常異なる．このため，各種の信号をクロックに同期させるための操作がオーバヘッドとなる．これに対し，非同期転送方式では，一つのデータを送るたびに行うハンドシェークの操作がオーバヘッドとなる．

2.4.2 転送モード

バス・トランザクションの本来の目的は，フェーズ 3) で行われるモジュール間でのデータ転送であるため，これ以外のフェーズはオーバヘッドとみなすこともできる．このオーバヘッドは，1回のトランザクションごとに必要となるため，1トランザクションで一つのデータを転送していたのでは，データ転送の効率が悪くなる．そこで，最近のシステム・バスでは，1回のトランザクションで複数のデータを転送する『バースト転送モード』により，このようなオーバヘッドを削減し，データ転送の効率を向上させている[16]．

これは，フェーズ 3) において複数のデータを転送することにより，指定されたメモリ・アドレ

スから始まる連続したデータを転送する方式である．この方式は，特に，キャッシュ・メモリを搭載したシステムで有効である[17]．16〜64バイト程度のブロック転送を効率的に行うことにより，キャッシュのブロック・サイズを大きくしてミス率を低減することができるためである．

2.4.3 ブロードキャスト

マルチプロセッサ・システムでは，プロセッサ間の同期などのため，全プロセッサに対して同じ情報を送るブロードキャスト操作が必要となる場合がある．このため，最近のシステム・バスでは，マルチプロセッサ・システムの構築を容易にするため，ブロードキャスト操作をアーキテクチャで規定している．

一般に，1対1のハンドシェークをデータ転送の基本とする非同期転送方式では，1対多のデータ転送に向いていないとされていたが，最近では，非同期転送方式のバスでも，ブロードキャスト機能をサポートするようになってきている．

2.4.4 バス占有方式

マスタがバスの使用権を獲得している間は，他のモジュールはバスを使用することができない．すなわち，バスはロックされるのである．通常のデータ転送では，マスタは，アドレスを送り出してから，データ転送を完了するまでの間，バスの使用権を解放しないため，バスで受け付け可能なデータ転送要求は一つに限られる．

しかし，この方式では，スレーブがアドレスを受け取ってから，データ転送完了をマスタに通知するまでの間，バスがロックされてしまい，マスタ・スレーブ間でのデータ転送以外は，システムにとって有用な動作ができなくなってしまう．この点は，特に，応答時間の長いスレーブをバスに接続した場合や，バスがシステム性能のボトルネックとなるバス共用型のマルチプロセッサ・システムで重要な問題となる．

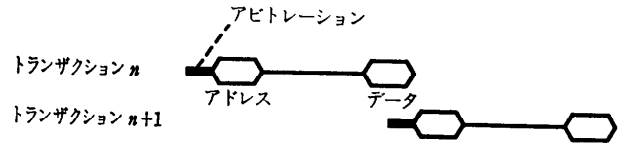
最近では，このような問題点を解決するため，図-2に示すように，アドレスの転送とデータの転送を分離したパイプライン転送方式やスプリット・トランザクション方式を採用した高性能バスが出現している．

1) パイプライン転送

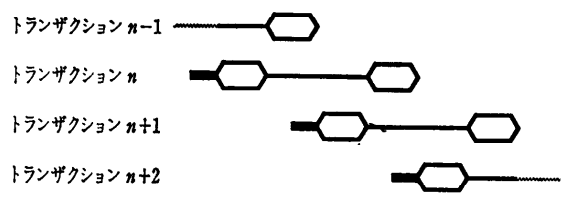
アドレス転送とデータ転送をパイプライン的に行う方式．複数のトランザクションを，時間的にオーバラップさせて行うことにより，バスのスループット向上を目的としている．パイプライン転送を行うバスの例には，Symmetry に搭載されている SSB (Sequent System Bus)[20] がある．

2) スプリット・トランザクション

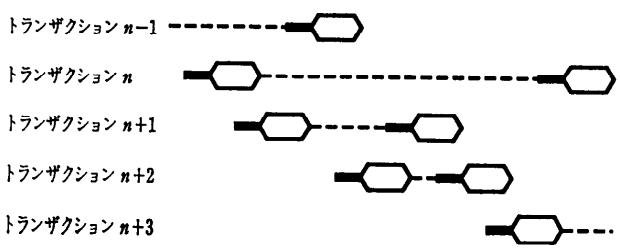
データ転送要求と転送完了通知を分離し，おのおのを別個のトランザクションで行う方式．二つのトランザクションの間，バスはアイドル状態に戻るため，この間に別のトランザクションを行うことが可能である．このデータ転送方式は，特に，応答時間の長いスレーブに対してアクセスを行う際，長時間バスを専有することによる性能低下を防ぐのに有効[21]である．



(a) 通常の転送モード
トランザクションを行っている間は，バスをロックするため，他のトランザクションは実行不可



(b) パイプライン転送
トランザクションを行っている間も，他のトランザクションを実行可能 データ転送用のスロットは，アビトレーション時に予約される



(c) スプリット転送
データを転送するのにも，アビトレーションを行ってバスの使用権を獲得する

図-2 パイプライン転送，スプリット転送の概念

2.5 アビトレーション方式

バスに接続されているすべてのモジュールはバス使用権獲得の要求を行ってもよいことになっているため，複数のモジュールが同時にバス使用権獲得要求を行う可能性がある．このような場合に，複数のモジュールからの要求を調停（アビトレーション）し，どのモジュールに対してバスの使用権を与えるのかを決定する役割を果たすのがバス・アビタである．通常，アビトレーションのための信号線は，データ転送用の信号線とは別に設けられており，データ転送操作と同時に，次回のバス使用権割り当てのためのアビトレーション操作を行うようになっている．このような制御により，バス上でのデータ転送操作を連続して行うことができ，システム・バスのデータ転送能力を最大限に活用できる．

アビトレーション方式の分類には，ハードウェアの物理的な構成からみた分類と，アビトレーションの論理的な動作（アルゴリズム）からみた分類がある[22]．前者の分類によると，アビトレーション方式は，次の2通りに分類できる（両者を併用する方式もある）．

1) 集中アビトレーション方式

バス・アビタが独立したハードウェア・モジュールとして，1カ所に配置されている方式．バス使用権を要求するモジュールとバス・アビタは，その間に配置された『バス使用権要求およびバス使用許可を通知するための信号線』を介してハンドシェークを行う．

2) 分散アビトレーション方式

バス・アビタ機能を，バス上に分散させた方式．バス使用権を要求するモジュールは，バスに用意されたアビトレーション用の信号線に対して要求を出すとともに，この信号線をモニタすることにより，バスの使用権が得られたかどうかを調べる．

また，アルゴリズムからみると，アビトレーション方式は，次の3種類に分類できる．

1) 優先順位の高いモジュールにバス使用権を割り当てる方式．

2) バス使用権獲得が，モジュール間で公平になるように割り当てる方式．

3) 1と2を組み合わせた方式．

アルゴリズムにおいて重要な点は，各モジュールに対して公平にバスの使用権を割り当てることと，リアルタイム性を要求されるバス・トランザクションを優先させることである．

2.6 現　状

現在，ワークステーションの分野ではVMEバスが事実上の標準となっているが，このバスをシステム・バスとしてではなく，I/O機器を接続するためのバスとして搭載し，メモリ・アクセスは独自のバスで行うようになっているシステムも多い．また，パーソナル・コンピュータの分野では，PC/ATとの互換を考慮したEISA，PC-9800シリーズ用のバスを維持拡張したNESA，IBMのPS/2（日本語版 PS/55）で搭載しているマイクロ・チャネル，MacintoshIIに採用されたNuBusなど，さまざまなバスが用いられている．主なバスの特徴を，表-1［一部[17]］に示す．

これらの従来バスは，単一のシステム・バスをもつ1 CPUシステムでの利用を基本に設計されているため，複数のシステム・バスや階層的なバスをもつマルチプロセッサ・システムに必須のキャッシュ・コンシステンシ制御や排他制御に関する規定はバス・アーキテクチャに含まれていな

表-1 主なシステム・バスの概要（MultibusII, VMEbus, Nubusは，文献17）より引用）

	転送方式	データ／アドレス	最大転送能力	アビトレーション
VMEbus	非同期式	分離 (32+32)	20 MB/sec	集中型
MultibusII	同期式	共用 (32)	40 MB/sec	分散型
EISA	—	分離 (32+32)	約 33 MB/sec	集中型
NESA	—	分離 (32+32)	約 33 MB/sec	集中型
マイクロ・チャネル	非同期式	分離 (32+32) 注1)	20 MB/sec 注3)	分散型
NuBus	同期式	共用 (32)	40 MB/sec	分散型
Futurebus+	非同期式	共用 (32 or 64) 注2)	200 MB/sec 以上	分散型

注 1) アドレス・バスの32本は，データ・バスの一部として使用可能
注 2) データ・バスは，256 bit まで拡張可能
注 3) PS/2 での値

い．このため，従来バスを用いてマルチプロセッサ・システムを構成する場合，これらのアーキテクチャはモジュール側で規定する必要があった．このような状況下では，同じ仕様のバスに接続されるモジュールでも，互いに異なるアーキテクチャを採っている場合は，それらのモジュールを用いたシステム構築ができないという，インタ・オペラビリティの問題が発生する．

従来バスにおいては，この問題は，「スレーブのサポートしていないモードによるデータ転送を要求された」というような単純な場合にも発生することがある．このため，バスの仕様を統一したとしても，任意のモジュールを組み合わせてシステムを構築することは困難であった．

また，従来バスでは，アビトレーション方式やデータ転送方式，バス幅が厳密に規定されているため，用途（コストやパフォーマンス）に応じてバス幅や転送能力を選択するというような柔軟な応用や，デバイス技術向上による性能向上を取り入れることが困難であった．計算機で扱うデータ量は，今後ともますます増えていくと予想されるが，これに応えるためのバスとしては，64ビット・アドレッシングへの対応やデータ転送能力の強化も望まれるところである．

3. 高性能システム・バスの実例

ここでは，IEEE P 896.1 Futurebus＋ の仕様[2]を，性能面やマルチプロセッサ・サポートの面で強化された機能を中心に説明し，高性能システム・バスの動向を示す．

Futurebus＋ は，1983年から1987年にかけて開発された scalable performance multiprocessor system bus である Futurebus に，リアルタイム性，フォルト・トレラント性，メンテナンス性を追加したバスであり[23),24)]，現在，次世代バスとして注目されている．Futurebus＋ では，1)バス幅の拡張，2) 3線ハンドシェークを用いた2エッジ・ハンドシェークによる高速ブロック転送[18)]，3)パケット・モードのトランザクション導入，によるデータ転送性能向上の結果，**図-3** に示すとおり，数100 MBytes/sec 以上のバンド幅が実現されている[2)]．

3.1 Futurebus＋ のハードウェアにおける特徴

高速なデータ転送のためには，バック・プレーンにおいて，信号線のドライブ開始から論理値の確定までの時間（セトリング・タイム）を短縮することが必須であり，また，クロストーク，EMI（電磁放射）を少なくすることも重要となる．このため，Futurebus＋ では，ハードウェア信号の物理的なインタフェースとして，従来の TTL (Transistor Transistor Logic) ではなく，BTL (Backplane Transceiver Logic) が採用されている．

BTL は，1)高いドライブ能力をもつ低キャパシタンス（1～2 pF）のバス・ドライバを採用した

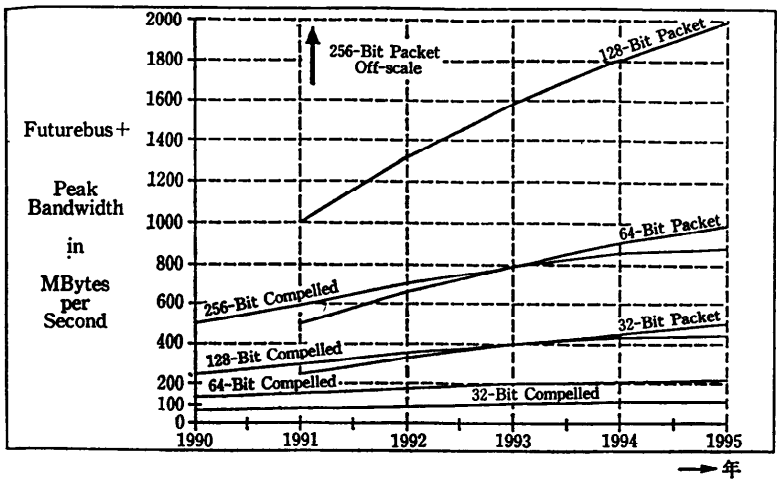


図-3[3)] Futurebus＋ の転送速度（ピーク）向上の予測
半導体デバイスの性能向上予測をもとにしている．

点，2)信号の振幅を1V (TTL では2V) と小さくした点，3)立ち上がり，立ち下がりを長くした台形波形を理想 (図-4) とした点，が大きな特徴である[11]．1)によりバスの負荷を低くすることができ，2)によりバス・ドライブに必要な電流を削減できるため，高いドライブ能力と相まって，セトリング・タイムの短縮が可能となっている．さらに，3)によりバック・プレーン上の波形がなだらかとなるため，高調波の発生が少なくなり，クロストークや EMI を押さえることができる．

このような BTL デバイスの採用は，論理素子の3V (または，3.3V) 動作化[25]とともに，計算機ハードウェアの新しい流れといえる．

3.2 Futurebus+ のアビトレーション

Futurebus+ でのアビトレーションは，Arbitration Bus と呼ばれる8ビットのバス，および，3本のハンドシェーク線，2本のステータス線，および，各モジュールに与えられている固有の Arbitration Number を用いて行われる．

アビトレーション方式の特徴は，次の5点である．1)リアルタイム性を要求される応用のために各モジュールのバス使用要求に優先順位を付ける．2)次回のバス使用権を決めた後であっても，そのトランザクションが開始される前であれば，より優先順位の高いモジュールにバスの使用権を横取り (プリエンプション) させることを可能とする．3)同じ優先順位のバス使用要求の間では，公平にバス使用権を割り当てることを可能とする．4)モジュール間の割り込み通知を実行できるように，アビトレーション・フェーズだけで1バイトのメッセージを送れるようにする．5)バスがアドレス状態の場合は，アドレス／データ・バスを利用して，アビトレーション時間を短縮する．

3.2.1 優先順位の決定

モジュールに Arbitration Number をもたせる方法は，バス要求に関してモジュール間の優先順位が固定された方式のようにみえるが，Futurebus+ では，この番号を以下に示す三つのフィールドに分ける (図-5) ことにより，バス使用要求に256レベルの優先順位をもたせるとともに，同じ優先順位をもつモジュール間でのラウンドロビン方式を実現している．

1) Priority Field (8〜1 ビット) モジュールの優先順位を決める部分

2) Round Robin Field (1ビット) ラウンドロビンのために動的に変化させる部分

3) Unique Field (5ビット) 同一優先順位のモジュール内でユニークな番号を割り当てる部分

各モジュールは，バスの使用権を獲得したモジュールの Arbitration Number を，常にモニタすることになっているが，この Priority Field が自分と同じであり，かつ，Unique Field の値が自分よりも大きい場合には，Round Robin Field を1にする．このような制御により，Unique Field の値が大きい順にバスの使用権を獲得していくことになる．Unique Field の値が最も小さいモジュールがバスの使用権を獲得した後は，全モジュールの Round Robin Field が0となるため，次にバス使用権を獲得するのは Unique Field の値が最も大きいモジュールとなる．

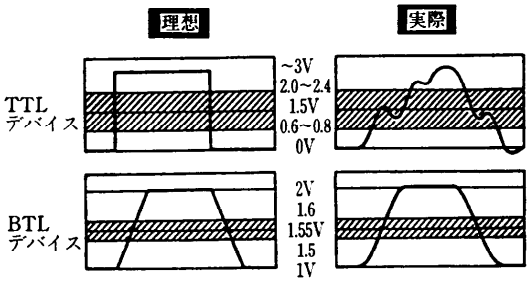


図-4[11] TTL/BTL の波形比較

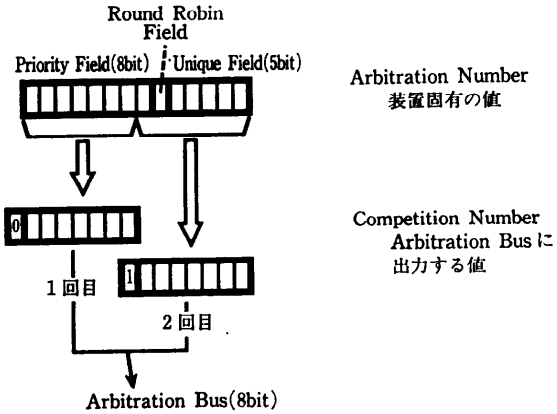


図-5 Futurebus+ における Arbitration Number の扱い
—Unrestricted Mode で Arbitration Number が14ビットの場合—
14ビットの Arbitration Number は，二つの Competition Number にマップされ，おのおのについてアビトレーション操作が行われる．Arbitration Number が7ビットの場合は，アビトレーション操作は1回．

アビトレーションの方式には，サポートを義務づけられている Unrestricted Mode と，オプショナルな Restricted Mode がある．前者のモードでは，Arbitration Number は 14 ビット，または，7ビットである．また，Unique Field がすべて1の Arbitration Number は，次に述べるアビトレーション・メッセージのためにリザーブされている．

3.2.2 アビトレーション・メッセージ

メッセージは，割り込みなどのイベント通知を，バスの使用権を獲得することなく行うためのものである．Futurebus＋ では，アビトレーションにおいて最も優先順位の高い Arbitration Number をメッセージに割り当てている．モジュールに搭載されたプロセッサへの割り込みは，このアビトレーション・メッセージを用いて通知することになっているため，Futurebus＋ には割り込み信号線はない．

メッセージを送りたいモジュールは，最初の Competition Number (図-5) をすべて1としてバス要求を行い，2回目の Competition Number として8ビットの値を送る．この操作により，データ転送のフェーズを経ることなく，Arbitration phase だけで，8ビットの値（メッセージ）を，他のモジュールに送ることができるようになっている．

3.2.3 プリエンプション

バスの使用権を獲得したモジュールがトランザクションを開始するまでの間に，より優先順位の高いモジュールでバス使用権の獲得要求が発生したときのために，Futurebus＋ では，プリエンプション（バス使用権の横取り）を許している．この場合，当該モジュールは，ステータス線をドライブすることにより，他のモジュールに対してアビトレーション操作の再起動を要求する．

3.2.4 アイドル・バス・アビトレーション

アビトレーション操作はデータ転送と並行して行われるため，バスの使用率が高い場合には，アビトレーション時間は他のデータ転送時間に隠されてしまい，システム性能には影響しない．しかし，バスがアイドル状態の場合は，バスを介したアクセスの時間にアビトレーション時間が陽に入ることになるため，システム性能上，これを削減することが重要になる．

Futurebus＋ では，バスがアイドル状態の場合には，アドレスやデータを転送するためのバス（必須の32本）を用いて，アビトレーション操作を高速に行うようになっている．これを，アイドル・バス・アビトレーションという．この操作により，バスを要求するモジュールが1個の場合には，通常のアビトレーション操作よりも短い時間でバスの使用権を割り当てることができる．なお，複数のモジュールがバスを要求していることが判明した場合は，並行して行われている通常のアビトレーション操作により，バス使用権を割り当てる．

3.3 Futurebus＋ におけるデータ転送

3.3.1 概 要

Futurebus＋ では，データ転送のために，次のハードウェア信号が用意されている．

1) アドレスとデータを送るためのアドレス／データ・バス（最低32ビット）

2) データ転送の種類を示す8ビットのコマンド・バス

3) 3本のケーパビリティ線（マスタから要求されたデータ転送モードを，スレーブ側でサポートしているかどうかをマスタに通知するための信号）

4) データ転送の結果を通知するための8本のステータス線

5) 6本のハンドシェーク線（アドレス用とデータ用に各3本）

6) アドレス／データ・バスに付随する情報を送るための8ビットのタグ

P 896.1 では，アドレス／データ・バスは64ビットまでの拡張を規定している．データ・バスについては，さらに，データのみを送るバスを加えることで，128ビットまたは256ビット・バスへの拡張を規定している．

データ転送は，1) Connection phase, 2) Data transfer phase, 3) Disconnection phase, の三つのフェーズにより行われる．なお，Futurebus＋ では，Data transfer phase の存在しないデータ転送モードもあるが，Connection phase と Disconnection phase は必ず存在することになっている．

3.3.2 データ転送方式の特徴

同期転送方式によりデータ転送を行うバスでは，そのデータ転送性能はバス・クロックによっ

て決まる．したがって，このような同期バスの性能を向上させるには，バス・クロック周波数を上げる必要がある．デバイス技術の進歩を取り入れるなどにより，クロック周波数を向上させたバスとモジュールを新規に作成することは可能であるが，この方法によると，それまでの周波数では動作していたモジュールが動作しなくなる可能性がある．このような理由から，同期バスにおいては，従来のハードウェア資産を活用するため，仕様を保ったままでバス性能を向上させていくことは困難である．Futurebus＋ では，このような制限を設けることなく，徐々にバス性能を向上させていけるように，非同期転送方式を採用している．また，従来の非同期転送方式で問題とされていたブロード・キャストと効率的なブロック転送を実現するために，3線ハンドシェークによる2エッジ・ハンドシェーク方式のデータ転送方式を採用している[18]．

3線ハンドシェークとは，従来までの「ストローブ信号」と「アクノレッジ信号」に，「反転アクノレッジ信号」を加えた3本のハンドシェーク線を用いる方式である．複数のモジュールから出力されるこれらの信号は，バス上で論理和（負論理の場合）されるため，**図-6**[18] に示すように，バス上のアクノレッジ信号は「データ転送を開始しても良い（転送に係わる全モジュールがデータを受け取れる状態にある）」ことを示し，バス上の反転アクノレッジ信号は「データ転送が終了した（転送に係わる全モジュールがデータを受け取った）」ことを示す．このため，バス・マスタは，複数のモジュールとの間でデータ転送を行う場合にも，データ転送の開始と終了を認識できるため，ブロード・キャストを効率良く行うことができる．

2エッジ・ハンドシェークとは，アクノレッジ信号の1エッジで，「データ転送の終了」と「次のデータ転送の開始」を兼用することにより，ブロック転送を効率的に行えるようにしたデータ転送方式である．この方式により，1データの転送終了と同時に次のデータ転送を開始できるため，**図-7**[18]に示すように，従来の4エッジ・ハンドシェークよりもブロック転送を効率的に行うことができる．

3.3.3 バス・トランザクションの種類

Futurebus＋ では，21 種類のトランザクションを規定している．バス・マスタは，Connection phase において，データ転送の相手となるスレーブのアドレスと要求するバス・トランザクションの種類を，おのおの，アドレス／データ・バスとコマンド・バスに送り出すことにより，マスタとスレーブの接続を行う．さらに，このフェーズでは，データ転送に用いるモード（スプリット・トランザクションやパケット・モード，次節参照）をマスタとスレーブの間で動的に決めるようになっている．この方式により，スレーブがサポートしないモードでのデータ転送を要求された場合でも，基本的なデータ転送モードに切り替えてトランザクションを実行することが可能である．これにより，モジュールでサポートする機能の違いに起因するインタオペラビリティの問題を解決している．

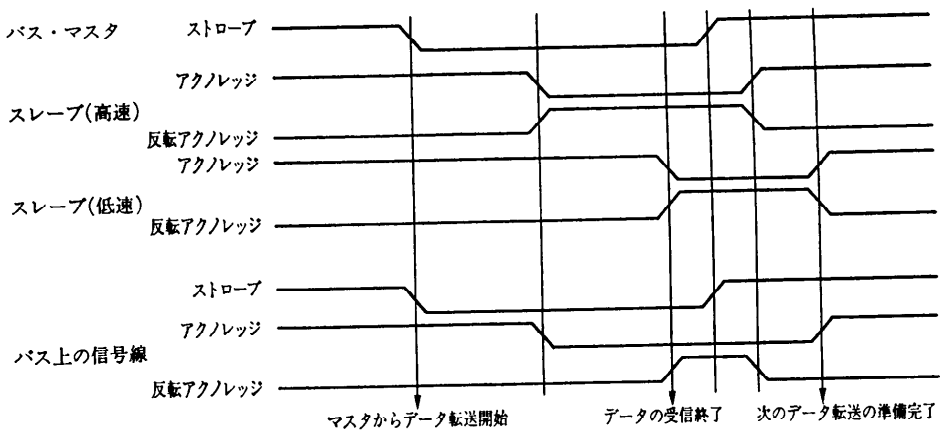


図-6[18]　3線ハンドシェークによるブロード・キャスト

3.3.4 データ転送モード

Futurebus＋ では，データ転送性能を向上させるため，1) スプリット・トランザクション，2) パケット・モードのデータ転送，が規定されている．これらは，従来のシステム・バスと大きく異なる点である．

a) スプリット・トランザクション

スプリット・トランザクションとは，**2.4.4** で示したとおり，データ転送要求とデータ転送完了通知を別個のトランザクションにより行うデータ転送方式であり，Futurebus＋におけるデータ転送の大部分は，このモードで行ってもよいことになっている．

ただし，Futurebus＋ ではスプリット・トランザクションはオプショナルとなっている（モジュールでのサポートを義務づけられているわけではない）ため，同一バス上に，スプリット・トランザクションをサポートしているモジュールと，サポートしていないモジュールが混在することがある．このような場合でも，データ転送相手におけるスプリット・トランザクション・サポートの有無を，バス・トランザクションの最初から意識する必要のないように，データ転送に用いるモードは，Connection phase 時におけるケーパビリティ線を用いたマスタ・スレーブ間の調節により，動的に決定するようになっている．

b) パケット・モード転送

従来の非同期バスでバースト転送を行う場合，データを受け取るモジュールは，全データ線のデータが確定したときにバス上のデータを取り込むとともに，モジュール間のハンドシェークにより次のデータを要求するようになっている．このようなデータ転送プロトコルを compelled protocol と呼ぶ．このモードにおけるデータ転送速度は，ハンドシェーク信号がモジュール間を往復する時間に制限される．その値は 40～50 nS (20～25 M 転送/sec) といわれている[15]．

これに対し，データを送るモジュールのクロックに同期させてデータを転送する方法が，パケット・モード転送である．このモードでデータ転送を行う場合は，おのおののデータ線は独立した伝送線として扱われるため，各データ線間でのデータ到達時間の違い(スキュー)を意識せずにデータ転送ができる．このモードにおいて，データ転送速度を制限するのは，バックプレーンの伝送線路としての物理的な限界である．その値は，各データ線あたり 100 Mbit/sec (100 M 転送/sec) といわれている[15]．

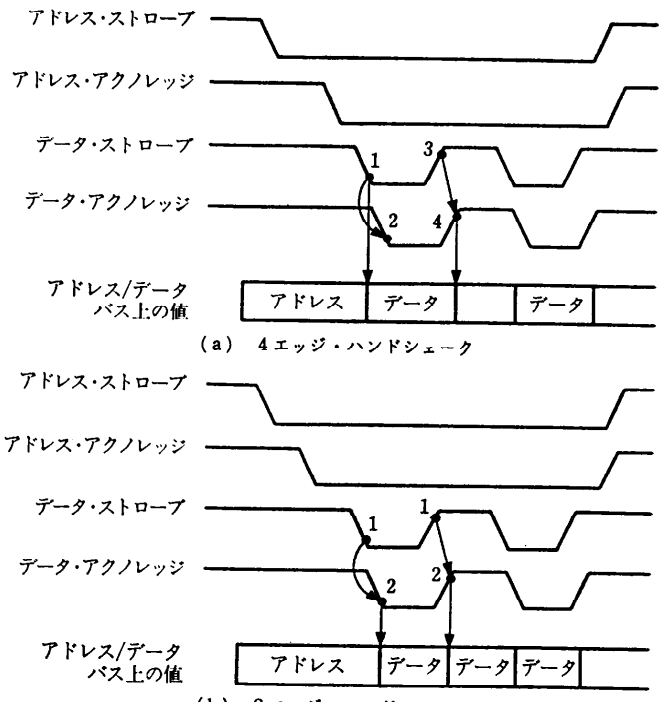


図-7[16]　非同期転送におけるハンドシェーク方式

Futurebus＋ でのパケット・モード転送では，データは，同期ビットとパリティを付け，NRZ (Non Return to Zero) コーディングされて転送される．なお，Futurebus＋ の規定では，このモードによるデータ転送はオプショナルである．

3.4 マルチプロセッサ・サポート機能

Futurebus＋ では，プロセッサ間の同期をサポートする機能や，キャッシュのコンシステンシを保つためのスヌーピング制御機能をバスの仕様に組み込んでいる．これにより，スヌープ・プロトコルの違いに起因するインタオペラビリティの問題が生じないようになっている．また，階層的なバスや複数のバスを用いてシステムを拡張することを容易にするため，複数バス間でのキャッシ

ュ・コンシステンシを考慮したスヌープ・プロトコルを採用している．

同期機構をサポートするためのメカニズムとしては，セマフォを実現するのに必要な不可分アクセスのための『ロック付きトランザクション』と，Futurebus＋ で導入された『ロック・コマンド付きトランザクション』がある．

3.4.1 ロック付きトランザクション

従来のシステム・バスでのロックは，バス自身をロックすることにより実現されていた．この方法は，バスが1本のシステムでは有効に機能するが，2本のバスからアクセスされるデュアル・ポート・メモリなどのモジュールについては，ロックを保証できないという点が問題となる．そこで，Futurebus＋ では，モジュール側にロック機能をもたせるようにしている．すなわち，モジュールがロック付きのトランザクションを受け付けた場合，そのすべてのバス・インタフェースをロックするように規定されている．

また，バス・トランザクションの終了時にロックを解放する従来方式のロックでは，複数のモジュールを同時にロックすることができないため，Futurebus＋ では，ロックなしのトランザクションが行われるまで，モジュールはロックを保持することになっている．したがって，マスタが連続してロック付きのトランザクションを行うことにより，複数のモジュールを同時にロックすることができる．

3.4.2 ロック・コマンド

前節に示したロック方式では，マスタがロックをかけている間は，他のモジュールがトランザクションを行うことができないという問題がある．特に，バスの使用率が高い場合には，これがシステムの性能ボトルネックになる可能性がある．

このため，Futurebus＋ では，バスを専有することなく，同期のための基本プリミティブを実行できるようになっている．これは，ロック・コマンドを付けたライト・トランザクションにより行われる．このロック・コマンドには，1) Swap Locked, 2) Fetch and Add, 3) Compare and Swap, の3種類がある．

さらに，これらのトラザンクションは，スプリット・トランザクションにより行うことも可能である．スレーブでは，このような不可分の操作が完了した後，マスタに転送完了通知を返すのである．これにより，長時間にわたってバスを専有することなく不可分アクセスを行うことができる．

3.4.3 キャッシュ・コンシステンシ・サポート

すべてのプロセッサが同一のアドレス空間を共有する密結合型のマルチプロセッサ・システムでキャッシュを使用する場合には，複数キャッシュ間でのデータの一致（コンシステンシ）を保つための制御を行う必要がある．すなわち，複数のキャッシュが，同じ主記憶アドレスのコピーをもっている場合，このブロックへの書き込みを他キャッシュに通知することにより，複数キャッシュの間でデータに矛盾が生じないようにするのである．このための制御方式を，キャッシュ・プロトコルと呼ぶ．

キャッシュのコンシステンシをハードウェア制御によって保つための方法は，ディレクトリ方式とスヌーピング方式に大別される．バス結合型のマルチプロセッサ・システムには後者が適合しており，ミニ・コンピュータや高性能ワークステーションに採用されつつある[26]．一般に，スヌーピング方式によるキャッシュ・プロトコルでは，キャッシュ・ラインを表-2 に示す5状態（状態名の頭文字をとって MOESI と呼ばれる）で表現し，プロセッサ・アクセスや他キャッシュの行うバス・トランザクションによる状態遷移を定義している．なお，5つの状態は，すべて使用されるわけではなく，キャッシュ・プロトコルによっては，許されない状態もある．

Futurebus＋ では，キャッシュのライン・サイズを 64 バイトとすることや，MESI の4状態を用いるキャッシュ・プロトコルを用いることが規定されている．このキャッシュ・プロトコルは，

表-2 スヌープ方式で管理するキャッシュ・ラインの状態

状態名	キャッシュラインが有効／無効	同一ラインが他キャッシュに存在する／存在しない	キャッシュと主記憶の内容が一致する／一致しない
Invalid	無 効	—	—
Shared	有 効	存在する	一致する
Exclusive	有 効	存在しない	一致する
Owned	有 効	存在する	一致しない
Modified	有 効	存在しない	一致しない

Futurebus で採用されていた MOESI の5状態をすべて用いるプロトコル[19),27)]から変更されたものである．Futurebus のプロトコルは，これまでに提案されたプロトコルのスーパセットであり，種々のプロトコルをもつモジュールの混在を可能としていたが，Futurebus＋ において，これが変更されたのは次の理由による[2),15)]．

1) MOESI 方式は，複雑さの割には性能を期待できないことが判明してきた．

2) MESI方式では，バス・ブリッジ（後述）を用いることにより，階層的なバスでもキャッシュのコンシステンシを保つことが可能．

Futurebus＋ のキャッシュ・プロトコルは，Illinois プロトコル[28)]をベースにしたものであり，これに，Dragon プロトコル[29)]や Firefly プロトコル[30)]における『キャッシュ・ラインの共有を示す信号を用いた制御』を追加している．キャッシュ・ラインの状態遷移に関係するバス・トランザクションは，下記の6種類である．

1) Read shared リード・ミスによるブロック・ロードを行う

2) Read modified ライト・ミスによるブロック・ロードを行う

3) Copyback ブロックを主記憶に書き戻す

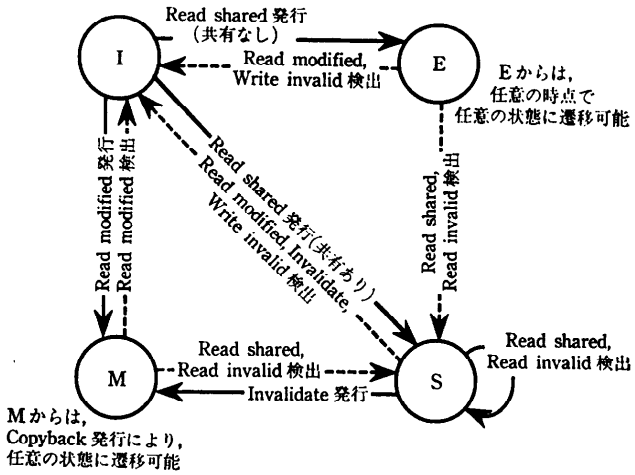
4) Invalidate 他キャッシュを無効化する

5) Read invalid 入出力装置との間のブロック転送を行う

6) Write invalid 入出力装置との間のブロック転送を行う

上記のバス・トランザクションは，すべてのキャッシュでモニタすることになっており，図-8 に示す状態遷移を行う．なお，この図は，キャッシュがスヌープにより Read shared や Read invalid を検出したときは，共有を示す信号をアクティブにする場合を示している．Futurebus＋ の規定では，共有を示す信号をアクティブにせずに，当該ブロックを無効化する動作も許している．また，必要であれば，他キャッシュによるブロック転送（Read shared, Read invalid, Copyback）をモニタし，当該ブロックをもたないキャッシュがデータを取り込むこと（snarf）も許されている．この場合も，共有を示す信号をアクティブにすることで，当該ブロックの共有を他キャッシュに知らせる必要がある．

さらに，Futurebus＋ では，図-9 に示すような，バス・ブリッジを介して接続される複数バスについてのキャッシュ・コンシステンシを保つ機能に



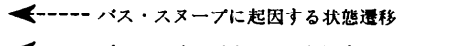


図-8 Futurebus＋ のキャッシュ・プロトコル（状態遷移図）

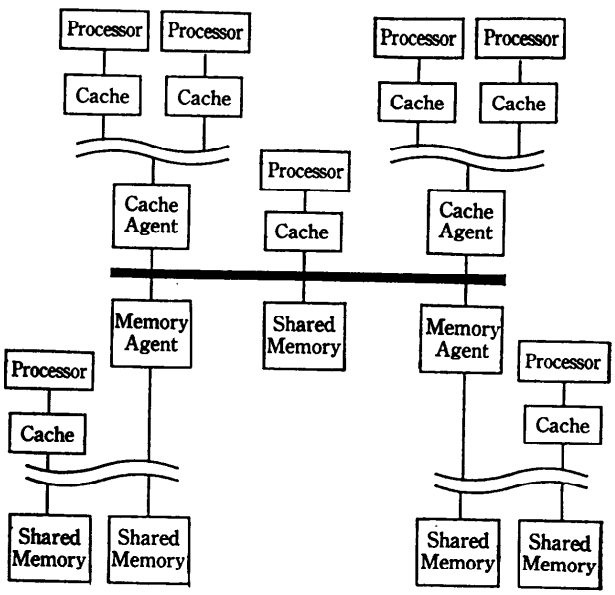


図-9[31)] 複数のバスをもつシステムの例
スプリット・トランザクションを用いてキャッシュのコンシステンシを保つ．

ついても規定がなされている[15]. このバス・ブリッジには, 1)バスを介して接続されているすべてのキャッシュの状態を管理するキャッシュ・エージェント, 2)バスを介して接続されているすべてのメモリ・ブロックの状態を管理するメモリ・エージェント, の2種類が規定されている. 両者とも, スプリット・トランザクションを利用してキャッシュ・コンシステンシを保つことになっている. バス・ブリッジを介したアクセスは時間がかかるため, このようなアクセスによるバス閉塞の回避が, 性能上・重要となるからである.

4. まとめ

本解説では, Futurebus+ を例にとり, データ転送能力やマルチプロセッサ・サポート機能の面での強化が進められていくマルチプロセッサ用高性能バスについて概説した. Futurebus+ は, このような機能強化を進めたバスの一つであり, 現時点においては, 将来の標準バスとして最も有力な候補となっているバスである.

今後のシステム・バスは, 機能面や性能面で強化されて発展していく結果, アビトレーションやデータ転送のプロトコルは, 従来のバスよりも複雑になっていくと考えられる. どのようなバスが将来の標準になるのかは予測できないが, 標準システム・バスとして普及するための重要な要素は, バス・コントローラなどのサポート LSI (Futurebus+ では, たとえば, National Semiconductor[31]) の充実を早く進めることであろう.

謝辞 本解説をまとめるにあたって多くの貴重なコメントをいただいた日本電気 C&C システム研究所コンピュータ・システム研究部の中田登志之主任に感謝します.

参考文献

1) Bybec, R.: Customized Buses Build on Industry Standards to Enhance Performance, Computer Design, Vol. 28, No. 3, pp. 109-112 (1989).
2) Futurebus+, IEEE Std. P896.1 Logical Layer Specifications Draft 8.2, IEEE Computer Society Press (1990).
3) Futurebus+, IEEE Std. P896.2 Physical Layer and Profiles Draft 4.0, IEEE Computer Society Press (1990).
4) 桜井: VME バスはなぜ重要か?, エレクトロニクス, Vol. 35, No. 11, pp. 53-58 (1990).
5) 宮園: Multibus-II はどこが違うか?, エレクトロニクス, Vol. 35, No. 11, pp. 47-52 (1990).
6) 万木: EISA のめざすもの, エレクトロニクス, Vol. 35, No. 11, pp. 38-41 (1990).
7) 大塚: NESA のねらいと戦略, エレクトロニクス, Vol. 35, No. 11, pp. 42-46 (1990).
8) 中山: マイクロチャネルはどこまで進歩するか?, エレクトロニクス, Vol. 35, No. 12, pp. 52-57 (1990).
9) 中村: NuBus (ニューバス) の完成度はなぜ高い?, エレクトロニクス, Vol. 35, No. 12, pp. 66-71 (1990).
10) ANSI/IEEE Std. 896.1-1987, IEEE Standard Backplane Bus Specification for Multiprocessor Architectures: Futurebus, IEEE Computer Society Press (1988).
11) 加藤: Futurebus+ はどこが Future なのか?, エレクトロニクス, Vol. 35, No. 12, pp. 61-65 (1990).
12) 森: なぜ SCSI-2 が必要なのか?, エレクトロニクス, Vol. 35, No. 11, pp. 33-37 (1990).
13) 辻村: GP-IB の世界, エレクトロニクス, Vol. 35, No. 11, pp. 59-63 (1990).
14) 森: 今, なぜ IPI なのか?, エレクトロニクス, Vol. 35, No. 12, pp. 58-60 (1990).
15) Hawley, D.: Supperfast Bus Supports Sophisticated Transactions, High Performance Systems, Vol. 10, No. 9, pp. 90-94 (1989).
16) Wilson, R.: Designers Seek New Approaches to Open I/O Bottlenecks, Computer Design, Vol. 27, No. 2, pp. 57-73 (1988).
17) 天野: 標準バスをマルチプロセッサで使う (上), 日経エレクトロニクス 1989.6.12, No. 475, pp. 328-336 (1989).
18) 天野: 標準バスをマルチプロセッサで使う (中), 日経エレクトロニクス 1989.7.24, No. 478, pp. 264-269 (1989).
19) 天野: 標準バスをマルチプロセッサで使う (下), 日経エレクトロニクス 1989.8.21, No. 480, pp. 320-323 (1989).
20) Symmetry System Summary, SEQUENT COMPUTER SYSTEMS INC. (1987).
21) 岡田, 岡本, 小町谷, 脇村: スプリット転送形システムバスの特性評価, 信学技報 CPSY 90-4, pp. 25-32 (1990).
22) van de Goor, A. J.: Computer Architecture and Design, p. 353, Addison-Wesley (1989).
23) Curran, L.: Futurebus Work Groups Close in on Final Spec, Electronic Design, Vol. 37, No. 19, pp. 37-44 (1989).
24) 1990年代の標準バス Futurebus+ が登場, 日経エレクトロニクス 1990.3.4, No. 521, pp. 141-150 (1990).
25) さらば, 5V 単一電源, 日経エレクトロニクス 1991.5.13, No. 527, pp. 143-199 (1991).
26) 浦城: キャッシュメモリの一致性について, 情報処理, Vol. 32, No. 1, pp. 64-73 (1991).
27) Sweazey, P. and Smith, A. J.: A Class of Compatible Cache Consistency Protocols and their Support by the IEEE Futurebus, Proc. of 13th Int. Symp. Computer Architecture, pp. 414-423

(1986).
28) Papamarcos, M. et al.: A Low-Overhead Coherence Solution for Multiprocessors with Private Cache Memories, Proc. 11 th ISCA, pp. 348-354 (1984).
29) McCreight, E. M.: The Dragon Computer System: An Early Overview, Technical Report, Xerox RARC (1984).
30) Thacker, C. P. et al.: Firefly: A Multiprocessor Workstation, IEEE Trans. Comput, C-37, No. 8, pp. 909-920 (1988).
31) 標準化進む Futurebus+，チップ・セットのサンプル出荷始まる，日経エレクトロニクス 1990. 2. 4, No. 519, pp. 84-85 (1990).

(平成3年7月24日受付)

堀川 龍 (正会員)

1959年生. 1981年同志社大学工学部電子工学科卒業. 1983年京都大学大学院工学研究科修了. 同年日本電気(株)入社. マイクロ・プロセッサや計算機システムのアーキテクチャおよび計算機能評価ツールの研究開発に従事. 現在は同C&Cシステム研究所コンピュータ・システム研究部主任.