専用プログラマブルプロセッサによるマルチメディア処理技術に関する研究

平成16年12月

小島 啓二

専用プログラマブルプロセッサによる
マルチメディア処理技術に関する研究

平成 16 年 12 月

小島 啓二

内容梗概

本論文は，筆者が 1983 年から現在まで（株）日立製作所中央研究所ならびにシステム開発研究所において，2003 年から現在まで大阪大学大学院情報科学研究科マルチメディア工学専攻在学中に行ってきた，専用プログラマブルプロセッサによるマルチメディア処理技術に関する研究成果をまとめたものである。

近年，音楽や映画といったマルチメディアコンテンツのデジタル化が急速に進み，デジタル放送やブロードバンド接続を介して，多様なマルチメディアサービスの提供が開始されている．サービスを実現するためのシステムは，プロバイダ側のマルチメディアデータベースサーバとコンシューマ側のマルチメディア端末を中心として構成される．マルチメディアサービスを普及させるには，顧客の嗜好に合った高音質，高画質のコンテンツを低コストで提供する必要がある．コンテンツの高品質化については，音声や映像をより低いビットレートに圧縮して伝送する新規格が次々と開発されている．マルチメディア端末においては，これらの新規格に柔軟に対応し，低コストでそのメリットを享受できるようにすることが課題となる．また顧客の嗜好に合った新サービスを迅速に提供するには，サービスに対する顧客の反応を常に蓄積，分析する必要がある．したがって，マルチメディアデータベースサーバにおいては，見たいコンテンツの一覧を表示する，といったコンシューマからの定型的な検索要求を高速に処理するだけでは不十分である．顧客のコンテンツ利用状況の分析を目的とする，サービス企画者からの大規模で非定型的な検索要求に効率良く対応できることが重要な課題となる．

このような背景のもと，本論文では，専用プログラマブルプロセッサによるマルチメディア処理により，上記課題を解決する方法を提案する．本論文は，5 章から構成されている．第 1 章では，研究の背景と方針を述べる．第 2 章では，マルチメディアの実時間処理を，機能固定のハードウェアでなく，C 言語で記述されたソフトウェアで実行するシステムオンチップを提案する．次に第 3 章では第 2 章で述べたシステムオンチップを用いて，端末ハードウェアの変更なしに新サービスに対応するための，マルチメディア端末のソフトウェア設計手法を提案する．さらに第 4 章では，顧客の嗜好分析を目的とした大規模で非定型的な検索の処理効率を向上させることにより，新サービスの迅速な提供を支援する，マルチメディアデータベースサーバの高速化方式を提案する．最後に，第 5 章で，本研究で得られた成果と今後の課題を要約する．

第 1 章では，端末とデータベースサーバから構成される，マルチメディアサービスシステムの課題について述べ，それらを解決する手段として，専用のプログラマブルプロセッサを用いたマルチメディア処理技術に焦点を当てる．従来のマルチメディア処理向けのプロセッサ技術の概要と残された課題について議論し，その解決に向けた研究方針を述べる．

第 2 章では，まずマルチメディア処理を行う従来の専用プログラマブルプロセッサが，プログラミングにアセンブラ言語の使用を必要とし，そのソフトウェア開発の生産性が低いという問題点について論じる．次に C コンパイラの最適化可能性を分析評価しながらアーキテクチャを設計することにより，この問題を解決するプロセッサアーキテクチャを提案する．本アーキテクチャは，コンパイラによる最適化を支援するための情報を命令フォーマットに埋め込むと同時に，データ転送専用プロセッサを導入してデータキャッシュのヒット率を向上させることにより，C 言語でのプログラミングでも十分な性能を発揮する．さらに提案アーキテクチャを LSI として実装し，C 言語で記述したデジタル TV のデコーダソフトウェアと最適化コンパイラを用いて評価を行う．

第 3 章では，まず新しいソフトウェアをマルチメディア端末に動的に追加して，新サービスに対応する際に生じる問題点について述べる．前章で提案したプロセッサを適用して主要な端末機能をソフトウェア部品化すれば，ネットワーク経由のソフトウェアダウンロードによる端末の機能拡張を実現できる．しかしながら，機能追加後のソフトウェア群が消費するハードウェア資源の総量が，実資源量を超えると，実時間応答性が損なわれるという問題が生じる．この問題を解決するために，各ソフトウェア部品の資源消費プロファイルと，サービスにおけるソフトウェア部品の並行実行度から，機能追加後の資源消費量を予測するためのフレームワークを与える．その上で，画質等の処理品質を変えると消費資源量も変化することに着目し，処理品質をサービスの並行性に応じて動的に制御することにより，実時間応答性を保証する方式を提案する．さらに本設計手法に基づくマルチメディア端末を試作し，初期設計に対して複数のサービス機能追加を試行し，その実現性と有効性を評価する．

第 4 章では，まずサービス提供側の中心となるマルチメディアデータベースサーバにおける性能要件について議論する．この中で，サービス利用者による定型的な検索と，サービス企画者による大規模で非定型的な検索の両方に対して十分な応答性能が要求されることに焦点を当てる．定型的な検索に対しては，従来から用いられている索引付けによる高速化が有効であるが，非定型的な検索を同様の手法で高速化しようとすると，データ量の

増大と更新性能の低下を招き，現実的でない．次にこの問題を解決するデータベースサーバの高速化方式を提案する．具体的には，索引を利用できない非定型的な検索に対して，大容量の主記憶に必要なデータをベクトル形式で動的に展開し，データベース演算専用のプログラマブルプロセッサでパイプライン処理することで処理効率を向上させる．さらに本プロセッサを LSI 実装し，リレーショナル型のデータベース管理システムを改造したマルチメディアデータベースサーバに適用して性能評価を行う．

第 5 章では，結論として本研究で得られた成果を要約し今後の課題を述べる。

研究業績

A. 学術論文誌論文

1. 金田泰，小島啓二，菅谷正弘："ベクトル計算機のための探索問題の計算法「並列バックトラック計算法」，" 情報処理学会論文誌，Vol. 29，No. 10，pp. 985-994 (1988).
2. Torii, S., Kojima, K., Yoshizumi, S., and Sakata, A.: "A Relational Database System Architecture Based on a Vector-Processing Method," *Information Sciences*, No. 48, pp. 135-155 (1989).
3. 小島啓二，鳥居俊一，吉住誠一："ベクトル型データベースプロセッサ IDP，" 情報処理学会論文誌，Vol. 31，No. 1，pp. 163-173 (1990).
4. 鳥居俊一，小島啓二，金田泰，坂田明治，高橋政美："マージ型ベクトル演算機構を用いた非数値処理の高速化方式，" 情報処理学会論文誌, Vol. 34, No. 1, pp. 109-119 (1993).
5. 小島啓二，西岡清和，川口敦生，細木浩二，薦田憲久："ソフトウェアによる実時間デジタルメディア処理向けシステムオンチップ，" 電子情報通信学会論文誌 D-I，Vol. J87-D-I，No. 12，pp. 1051-1059 (2004).

B. 国際会議会議録

1. Yuasa, T. and Kojima, K.: "The IOTA Programming System - Supports for Building Modules, Modulebase," in Nakajima, R. and Yuasa, T. eds. "*Lecture Notes in Computer Science*," Vol. 160, pp. 61-71, Springer-Verlag (1983).
2. Kojima, K.: "A Pattern Matching Algorithm in Binary Trees," in R. Nakajima ed. "*Lecture Notes in Computer Science*," Vol. 147, pp. 99-114, Springer-Verlag (1982).
3. Torii, S., Kojima, K., Yoshizumi, S., Sakata, A., Takamoto, Y., Kawabe, S., Takahashi, M., and Ishizuka, T.: "A Relational Database System Architecture Based on a Vector Processing Method," in *Proc. of the Third IEEE International Conference on Data Engineering*, pp. 182-189 (1987).
4. Kojima, K., Torii, S., and Yoshizumi, S.: "IDP - A Main Storage Based Vector Database Processor -," in Kitsuregawa, M. and Tanaka, H. eds. "*Database Machines*

and Knowledge Base Machines," pp. 47-60, Kluwer Academic Press (1988).

5. Kanada, Y., Kojima, K., and M. Sugaya: "Vectorization Techniques for Prolog," in *Proc. of 1988 ACM International Conference on Supercomputing*, pp. 539-549 (1988).
6. Torii, S., Kojima, K., Kanada, Y., Sakata, A., Yoshizumi, S., and M. Takahashi: "Accelerating Non-Numerical Processing by an Extended Vector Processor," in *Proc. of the Fourth IEEE International Conference on Data Engineering*, pp. 194-201 (1988).
7. Kojima, K., Matsuda, Y., and Futatsugi, S.: "LIVE - Integrating Visual and Textual Programming Paradigms -," in *Proc. of the 1989 IEEE Workshop on Visual Languages*, pp. 80-85 (1989).
8. Kojima, K. and Myers, B. A.: "Parsing Graphic Function Sequences," in *Proc. of the 1991 IEEE Workshop on Visual Languages*, pp. 112-117 (1991).

C. 解説

1. 湯浅太一, 小島啓二, 中島玲二: "モジュラー・プログラミングのための支援環境," 情報処理, Vol. 23, No. 5, pp. 433-441 (1982).
2. 小島啓二: "次世代エディタ," 情報処理, Vol. 25, No. 8, pp. 869-874 (1984).
3. 鳥居俊一, 小島啓二, 吉住誠一, 河辺峻, 高橋政美, 久代康雄: "リレーショナル・データベースの処理速度向上を図る CPU 内蔵型データベースプロセッサ," 日経エレクトロニクス, 1987 年 2 月 9 日号, No. 414, pp. 185-206 (1987).
4. 小島啓二: "ビジュアルインタフェースの研究動向と応用," Chapter 2.9, ビジュアルインタフェース - ポスト GUI を目指して, bit 別冊, 共立出版, pp. 168-175 (1996).

D. 国内講演予稿集

1. 湯浅太一, 小島啓二, 柴山悦哉, 中島玲二, 萩野達也, 本田道夫: "イオタ入門," 情報処理学会記号処理研究会, No. 016-001, pp. 1-6 (1981).
2. 吉住誠一, 鳥居俊一, 小島啓二, 坂田明治, 井門徳安: "ベクトル型高速データベース

プロセッサ -基本構想-,” 情報処理学会第３２回全国大会, pp. 915-916 (1986).

3. 鳥居俊一, 小島啓二, 坂田明治, 井門徳安, 吉住誠一：“ベクトル型高速データベースプロセッサ - ソフトウェア方式 - ,” 情報処理学会第３２回全国大会, pp. 917-918 (1986).
4. 小島啓二, 鳥居俊一, 坂田明治, 吉住誠一：“ベクトル型高速データベースプロセッサ - アーキテクチャ - ,” 情報処理学会第３２回全国大会, pp. 919-920 (1986).
5. 坂田明治, 鳥居俊一, 小島啓二, 高本良史, 吉住誠一：“ベクトル型高速データベースプロセッサ - 性能評価 - ,” 情報処理学会第３２回全国大会, pp. 921-922 (1986).
6. 小島啓二, 鳥居俊一, 坂田明治, 高本良史, 吉住誠一, 河辺峻：“ベクトル型高速データベースプロセッサ IDP - ソート性能評価 - ,” 情報処理学会第３３回全国大会, pp. 871-872 (1986).
7. 鳥居俊一, 小島啓二, 金田泰, 坂田明治, 吉住誠一, 高橋政美：“拡張ベクトル演算による非数値処理高速化,” 電子情報通信学会データ工学研究会, DE88-8, pp. 57-64 (1988).
8. 小島啓二, 松田芳樹：“視覚的プログラミングシステム LIVE,” 情報処理学会第 30 回プログラミングシンポジウム, pp. 9-17 (1989).
9. 安村通晃, 小島啓二：“スーパーコンピュータ上でのグラフ問題の高速化,” 情報処理学会第 31 回プログラミングシンポジウム, pp. 9-17 (1990).
10. 小島啓二：“ヒューマンインタフェースの動向 - ポスト GUI の座をめぐって - ,” 情報処理学会ソフトウェア工学研究会, No. 093-013, pp. 101-108 (1993).
11. 湯浅寛子, 小島啓二：“情報のブロードキャッチシステム,” 情報処理学会情報メディア研究会, No. 013-006, pp. 37-44 (1993).
12. 藤澤浩道, 加藤寛次, 小島啓二, 友広修造, 和歌山哲：“概念中心型文書管理と全文検索による情報共有,” 情報処理学会情報学基礎研究会, No. 035-004, pp. 19-26 (1994).
13. 小島啓二, 西岡清和, 野尻徹：“メディアプロセッサ (MAP) のビジョンとアーキテクチャ,” 第 59 回情報処理学会全国大会, １C-1, pp. 87-92 (1999).

目次

第1章　序論

1.1 研究の背景と目的

21 世紀に入り，音声や映像を用いた情報サービスを受けるためのネットワークインフラの整備が，急ピッチで行われている．まず家庭用の広帯域通信ネットワークとして，ADSL (Asynchronous Digital Subscriber Line) と FTTH (Fiber To The Home) が普及し，ケーブル TV (Television) の配信網の利用も含め，ピーク速度が 1Mbps (Mega-bit per second) から 100Mbps のインターネット接続の提供が開始されている [1] [2]． 放送インフラも衛星デジタル放送，地上デジタル放送が開始され，2011 年のアナログ放送停波に向けて順次デジタルに置き換わっていくと予想される [3]．また爆発的に普及した携帯電話も，第 3 世代に入って EV-DO (Evolution Data Only) [4]，FOMA [5] など，広帯域化が進んでいる．無線接続としては，802.11a/b/g といった無線 LAN 規格も一般化し，街中のアクセスポイントを利用した低コストでの IP (Internet Protocol) 通信が可能となりつつある [6]．さらに，前述の無線 LAN (Local Area Network) や電灯線を利用した高速通信方式 [7]，あるいは Bluetooth [8] といった通信技術を用いた，宅内のネットワーキング技術の開発も進んでいる [9]．

一方，音楽や映画といったマルチメディアコンテンツのデジタル化も進んでいる．CD (Compact Disc) や DVD (Digital Versatile Disk)，あるいは HDD (Hard Disk Drive) といった蓄積メディア上での利用に加え，上述のネットワークインフラを介し，データ放送，VOD (Video On Demand)，音楽や映画のダウンロード配信，TV 電話，といったマルチメディアサービスの提供が開始されている [10]． 図 1.1 に示すように，サービスを実現するためのシステムは，コンシューマ側のマルチメディア端末と，サービスプロバイダ側のマルチメディアデータベースサーバとを中心として構成される．マルチメディア端末としては，デジタルセットトップボックスに代表される，TV をディスプレイ装置とするものをはじめとして，ネットワークに直結可能なデジタル TV も既に開発されている [11]．またマルチメディアデータベースサーバとしては，関係データベース管理システム [12] 等の汎用データベース管理システムでコンテンツの書誌情報を管理し，コンテンツの実体は配信性能を最適化するように，専用のサーバに置く構成が代表的である [13]．

マルチメディアサービスを普及させるには，コンシューマの嗜好に合った高音質，高画質のコンテンツを低コストでタイムリーに提供する必要がある．コンテンツの高品質化を

実現するには，高品位な音声やHD(High-Definition)映像をより低いビットレートに圧縮して伝送する必要がある．例えばMPEG2[14]でHD映像を送るには，20Mbps程度が必要となる．現状の広帯域ネットワークはベストエフォート型で，実効的には 2-3Mbps の使用環境が多く，安定したHD映像伝送のためにはさらに一桁上の圧縮率が必要となる．この状況を改善するため，ネットワークのさらなる広帯域化とともに，より圧縮率の高い新規格も次々に開発されている．

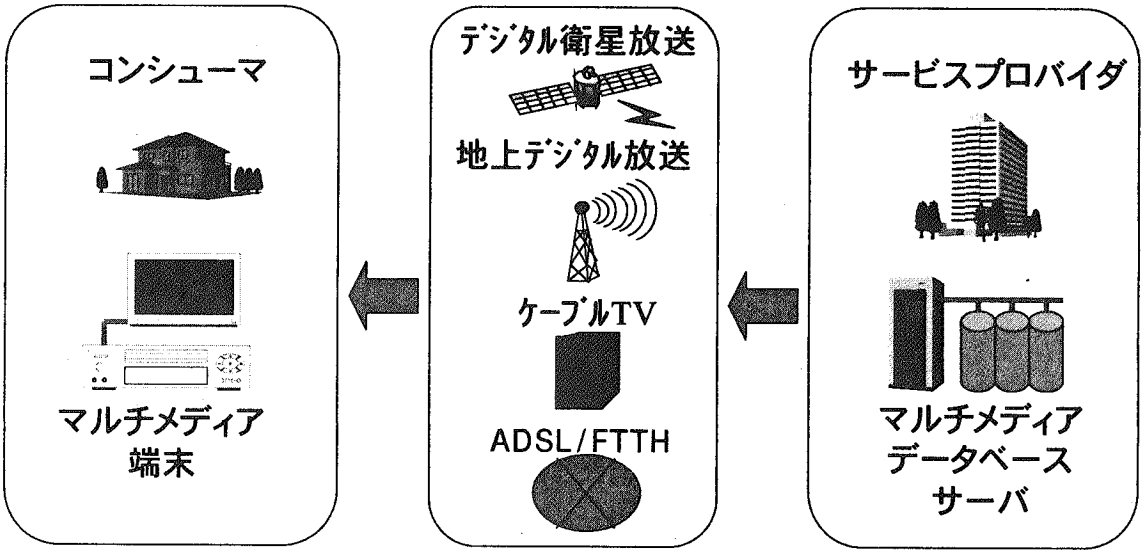

図 1.1　マルチメディアサービスシステム

具体的には，MPEG4，H.264 とより圧縮率の高い方式が提案，規格化され，放送やパッケージメディア用の標準規格として採用されている[15]．これ以外にも優れた方式が各ベンダによって個別に提案，実用化されて普及しつつあり，さらにフラクタルやウェーブレット変換を用いた圧縮アルゴリズムの研究開発も精力的に行われている[16] [17]．新方式は高い圧縮率が得られる半面，処理負荷が高く，マルチメディア端末においては，これらの新規格に対応できる高い処理能力が必要とされる[18]．また端末開発の生産性を高くして，低コストで迅速にその高品質を享受できるようにすることが重要である．

またコンシューマの嗜好に合った新サービスを提供するためには，従来のサービスに対

するコンシューマの反応を常に蓄積，分析することが有効となる．マルチメディアデータベースサーバにおいては，コンテンツ情報や課金情報とともに，CRM(Customer Relationship Management）を実現するための豊富な顧客情報を蓄積管理する必要が生じる．この新たなニーズは，データベースサーバに対する機能や性能の面についても再検討が必要なことを示している．すなわち，見たいコンテンツの一覧を表示するといった，コンシューマからの定型的な検索要求を高速に処理するだけでなく，コンシューマのコンテンツ利用状況の分析を目的とするデータマイニングのような，サービス企画者からの大規模で非定型的な検索要求に効率良く対応できることが必要になる[19]．

マルチメディア処理向けプロセッサは，上述の端末やデータベースサーバの課題を解決するために提案されている技術である．以下，本論文では，マルチメディア処理技術とは，音声や映像を含むマルチメディア情報を，加工，蓄積，あるいは伝送する技術であると定義し，マルチメディア処理向けプロセッサを，図 1.2 に示すように分類して議論する．

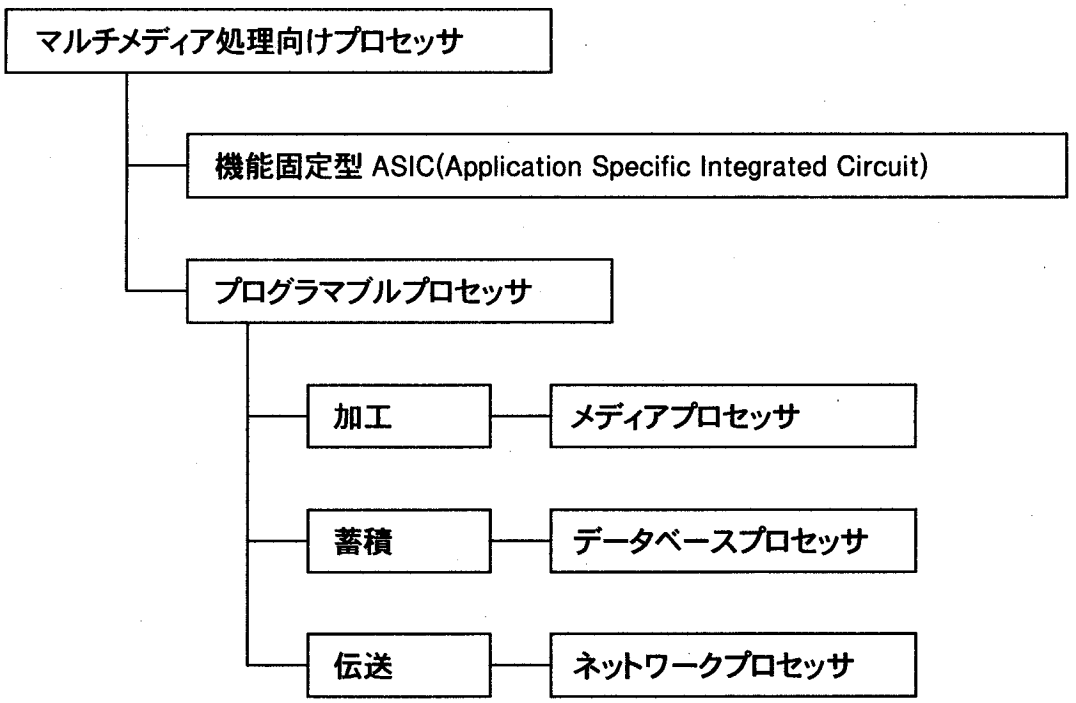

図 1.2　マルチメディア処理向けプロセッサの分類

機能固定型のASIC(Application Specific Integrated Circuit)[20]は，ターゲットとする仕様に特化してコストパフォーマンスを最適化できるため，デジタル放送の受信のように，規格が長期に渡って変化しないことが保証されている応用に適している．実際，多くのデジタル放送受信装置ではこの方式が採用されている[21]．

一方，プログラマブルプロセッサは最適化の度合いはASICに劣るものの，プログラムの入れ替えによって機能を変更できるため柔軟性に富んでいる．したがって前述のように，規格が短期間で変化していくと予想される，広帯域ネットワークを介したマルチメディアサービスに適している．プログラマブルプロセッサはさらにその用途に応じて，音声や映像データの各種加工を行うメディアプロセッサ，マルチメディアデータの蓄積管理を支援するデータベースプロセッサ，さらに伝送路であるネットワークでの，ルーティング処理を高速化するネットワークプロセッサに大別できる．

本研究の目的は，サービスプロバイダにとっての新サービス提供を容易化する，新たな専用プログラマブルプロセッサと，それを用いたマルチメディア処理方式を提案することにある．具体的には，先に述べた新サービス提供における，サービスプロバイダにとって優先度の高い次の2点を研究課題とする．

(1) メディアプロセッサの高性能とプログラマビリティを活かし，マルチメディア端末における新機能追加や新規格対応を低コストで迅速に行うことで，新サービス提供を加速化する．

(2) サービス企画者による，大量データのマイニングのような非定型的検索を，データベースプロセッサを用いて高速化することで，十分な顧客分析を可能とし，コンシューマの新サービスに対する満足度を向上させる．

なお，データベースサーバと端末を結ぶネットワークルータについても，例えばIPv4からIPv6でのネットワークプロトコル追加のように，種々のプロトコル拡張に迅速に対応する必要がある[22]．この課題を解決するために，プログラマブルなネットワークプロセッサが提案され適用が進んでいる[23]．これはネットワークキャリア側の課題と解決の動きであり，サービスプロバイダに焦点を当てる本研究においては対象外とする．

1.2 従来技術とその課題

1.2.1 メディアプロセッサとその課題

マルチメディアデータの圧縮伸長を含む加工処理の高速化を目的とするメディアプロセッサの研究は，汎用のマイクロプロセッサに対する演算ユニットの最適化から始まっている．最初の試みはUNIXサーバ用の高性能RISC(Reduced Instruction Set Computer)プロセッサにおいて，画像や音声の処理では8ビットや16ビットのデータを扱うのが多いことに着目し，32ビットの整数演算ユニットを8ビット演算が4個，あるいは16ビット演算が2個並列に実行できるように構成して性能向上を図るものである[24]．このような汎用マイクロプロセッサの演算ユニット強化によるマルチメディア処理の高速化手法は，PC(Personal Computer)用などの汎用マイクロプロセッサに対しても広く適用されている[25]．PC用の汎用マイクロプロセッサでこれらの機能強化を行っても，HD映像の圧縮伸長に適用するには少なくとも3GHz以上の高い動作周波数が必要となり[26]，消費電力も数十ワットを超えるためLSIとその冷却にかかるコストは高く，PC以外のデジタル家電機器に適用するのは困難である[27]．

一方，汎用プロセッサのマルチメディア演算機能強化に比べてより高い並列性を持たせ，大幅な性能向上を図るとともに，不要な機能を削除することで，低コスト化を狙ったメディアプロセッサが開発されている．具体的には，複数のDSP(Digital Signal Processor)コアをクロスバースイッチで結合するもの[28]，VLIW(Very Long Instruction Word)により1命令で複数のマルチメディア演算ユニットの制御を行って命令レベルの並列度を向上させるもの[29][30]，さらにこれらにハフマン復号回路や動き予測回路といった専用回路を付加してMPEG処理等の良く使われる特定応用をさらに高速化するもの[31][32]，などが提案されている．これらのメディアプロセッサは，コスト，消費電力ともにPC用汎用マイクロプロセッサより一桁低く，デジタル家電を含む幅広い応用をターゲットに開発されている．

メディアプロセッサの残された課題はソフトウェア生産性の向上にある．メディアプロセッサの並列性を十分に引き出すためには，アセンブラ言語を用いて複数の演算ユニットと各種レジスタやローカルメモリを，タイミングに留意しながら最適に制御する，高いスキルを要する複雑なプログラミング作業が必要となる[33]．結果としてソフトウェア生産性は低く，変更も困難で再利用が難しいという問題があり，新機能や新規格への柔軟で迅

速な対応，というメディアプロセッサ本来の目的達成に対する最大の障害となっている．

メディアプロセッサを応用して，新サービスに対応するマルチメディア端末の開発期間を短縮する手法としては，端末のハードウェア設計とソフトウェア設計を並行かつ協調的に行う，コンカレントデザインが適用できる[34]．さらにPCで行われているように，コンシューマがネットワーク経由でソフトウェアをダウンロードして端末機能を拡張することにより，新サービスに迅速に対応することも考えられる．しかし映像圧縮の新規格に対応するといった高負荷処理の追加においては，機能追加後に実時間応答性を保証するのが難しいという問題がある．PC では応答性の低下対策を含めて，端末ソフトウェアの管理はコンシューマの責任であるが，誰もが使えることを想定しているデジタル家電分野ではPCのソフトウェア更新モデルは適用が困難と考えられる．

1.2.2 データベースプロセッサとその課題

データの多角的分析を可能とする柔軟なデータベース管理システムとして登場したのが関係データベース管理システムである．関係データベース管理システムは，ポインタチェイン構造を持つ構造型データベース管理システム[35]が，チェインをたどる形での定型的な検索しかできなかったのに対し，正規化された 2 項関係である表に対して関係演算を施すことにより，非定型的な検索も簡単に指定できる[36]．構造型データベースシステムに比べ，検索性能が低下するという問題に対しては，高速にアクセスしたい表のカラムに予め B-tree 形式で索引付けをして高速化する方式が開発されている[37]．この方式により，定型的な検索に対しては構造型データベースシステムに比べて遜色無い性能を実現可能であるが，予め索引付けされていないデータに対する非定型的な検索については性能が改善されない．一方，すべてのデータに索引をつけるのは，データ量の増大と更新性能の悪化を招くため現実的でない．

索引を使わずに関係データベース管理システムを高速化する目的で提案されたのがデータベースプロセッサである．データベースプロセッサは，ディスク装置内に検索プロセッサを付加するディスクの高機能化方式[38] [39]，ディスク装置と CPU(Central Processing Unit)間の通信ラインに検索プロセッサを挿入してデータ転送に重畳して検索を行うストリーム処理方式[40]，MIMD(Multi-Instruction Multi-Data)型の並列検索を行うマルチプロセッサシステム方式[41] [42]に分類される．いずれの方式もデータベースの関係する領域を一括して並列に処理することにより，索引を使用せずに実用的な性能を得ることを狙

っている．反面，データベースプロセッサは少数のデータのみに定型的にアクセスするような並列度の低い検索に対しては，通信等のオーバヘッドが大きく効率が低い[43] [44]．1.1節で述べたように，マルチメディアデータベースにおいては，100万人規模のコンシューマからの定型的検索が同時に入ってくるためその高速化と，サービス企画者による大量データの一括アクセスを中心とする非定型的検索の高速化の両方が必要となる．従来のデータベースプロセッサの問題点はこの両立が難しいという点にある．

1.3 研究の方針

前節で述べた専用プログラマブルプロセッサによるマルチメディア処理の課題を踏まえ，サービスプロバイダにとっての新サービス提供の容易化を実現する．本研究の位置付けを図1.3に示す．

まずメディアプロセッサについて，C言語のような高水準言語によるプログラミングでは十分な性能を発揮できない理由を分析する．この分析はメディアプロセッサの構成要素を演算処理系とデータ転送系とに分解して行う．演算処理系については低コストで命令レベル並列性が得られるメリットを重視して，VLIWアーキテクチャを前提とする[33]．VLIWによるマルチメディア処理において，C コンパイラによる最適化のために必要な情報を命令形式の中に取り込む際の問題点について検討する．データ転送系については，映像処理におけるデータキャッシュのヒット率が低い点に着目して問題点を整理する．この分析結果にもとづいて演算処理系とデータ転送系の問題点を解決する方式を検討し，得られた方式をベースに，必要となる周辺ユニットを付加した実時間マルチメディア処理向けのシステムオンチップを提案する[45] [46] [47]．

次に，提案システムオンチップを応用する場合の課題である，ソフトウェア更新後の実時間応答性の保証を実現する方法について考える．ここで，実時間応答性が損なわれるのは，更新後のソフトウェアが，端末に用意されているハードウェアの最大資源量を超えて資源を消費しようとする場合である．したがって実時間応答性を保証するには，まず更新後のソフトウェアのハードウェア資源消費量を予測するための枠組みを与える．さらにその結果実時間応答性が損なわれると予測される場合に資源消費量を最大資源量以下に低減する方式が必要となる．ソフトウェアによるハードウェア資源消費量を低減する方法として，当該ソフトウェアを構成する音声や映像処理のためのソフトウェア部品の多くが，音

質や画質等の処理品質に関わるパラメータを変化させることで資源消費量を制御できることに着目する[48] [49] [50]．すなわち，端末の最大資源量に応じて目標とする品質を最適化させることで実時間応答性を保証する方式を検討する．結果として得られた資源消費量予測の枠組みと，品質最適化による実時間応答性保証方式とを組み合わせて，メディアプロセッサを応用したマルチメディア端末のソフトウェア設計手法を提案する[51]．

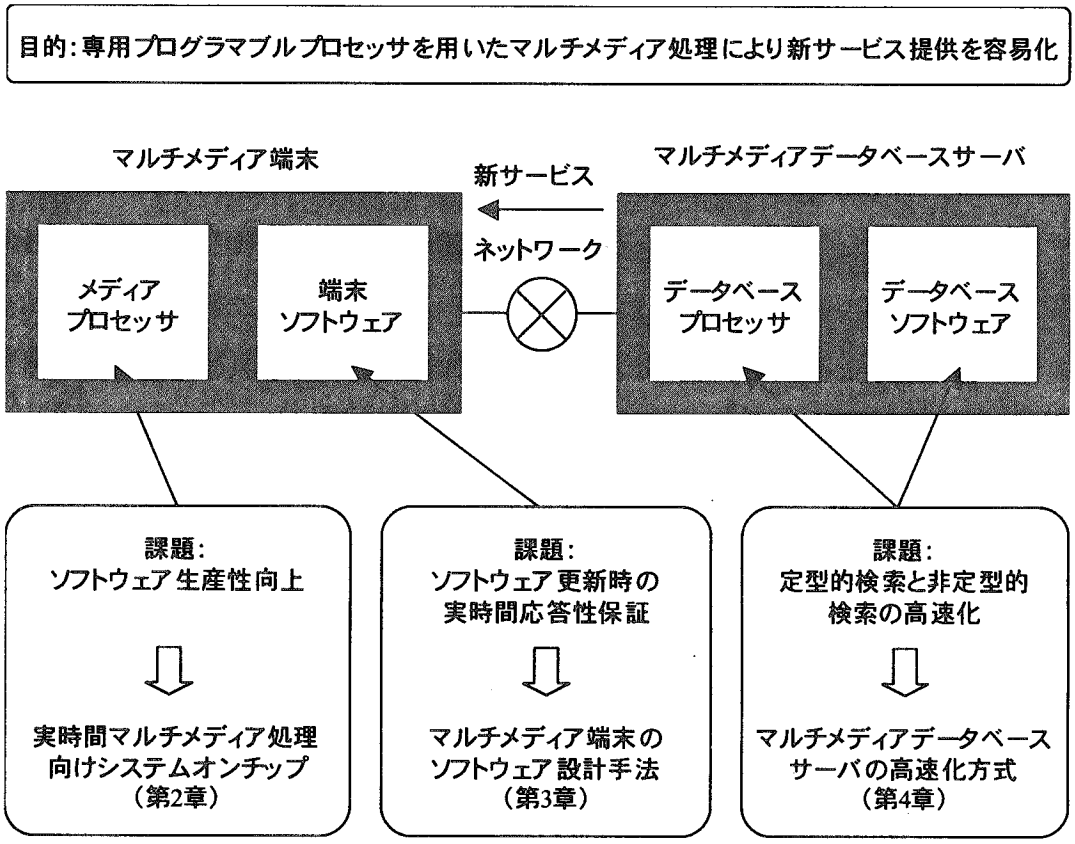

図 1.3　本研究の位置付け

マルチメディアデータベースサーバの課題である定型的検索と非定型的検索の高速化を実現するために，まず索引付けによる定型的検索の処理に最適化されている，従来の関係データベース管理システムのデータ格納形式と内部処理フローを分析する[52]．基本的な考え方として，処理フローの先頭で検索要求を解析し，定型的検索であれば索引を用いる従来のフローでそのまま処理し，非定型的検索であればデータベースプロセッサで並列処理する，ハイブリッドアプローチを採用する[53]．データの格納形式は従来と同様，定型

的な検索に最適な形式を用いるため，索引が使えない非定型的検索の処理においては，対象データ群を並列処理に適した形式に動的に変換してから，データベースプロセッサで処理する．非定型的検索の処理における本方式の効率は，この動的データ形式変換の性能と，変換後の関係演算処理の性能で決定される．そこでデータベースプロセッサの命令体系を，関係演算だけでなく，動的データ形式変換に対しても適用できることを目標に設計する[54] [55]．以上の検討に基づき，設計されたデータベースプロセッサと，それを用いた関係データベース管理システムから構成されるマルチメディアデータベースサーバの高速化方式を提案する[56] [57]．

1.4 論文の構成

本論文は，5章から構成されている．第2章ではマルチメディアの実時間処理を，機能固定のハードウェアでなく，C言語で記述されたソフトウェアで実行するシステムオンチップを提案する．次に第3章では第2章で述べたシステムオンチップを用いて，端末ハードウェアの変更なしに新サービスに対応するための，マルチメディア端末のソフトウェア設計手法を提案する．さらに第4章では，コンシューマの嗜好分析を目的とした大規模で非定型的な検索の処理効率を向上させることにより，新サービスの迅速な提供を支援する，マルチメディアデータベースサーバの高速化方式を提案する．最後に，第5章で，本研究で得られた成果と今後の課題を要約する．

第2章では，文献[47]に基づいて，まずマルチメディア処理を行う従来の専用プログラマブルプロセッサが，プログラミングにアセンブラ言語の使用を必要とし，そのソフトウェア開発の生産性が低いという問題点について論じる．次にCコンパイラの最適化可能性を分析評価しながらアーキテクチャを設計することにより，この問題を解決するプロセッサアーキテクチャを提案する．本アーキテクチャは，コンパイラによる最適化を支援するための情報を命令フォーマットに埋め込むと同時に，データ転送専用プロセッサを導入してデータキャッシュのヒット率を向上させることにより，C言語でのプログラミングでも十分な性能を発揮する．さらに提案アーキテクチャをLSIとして実装し，C言語で記述したデジタルTVのデコーダソフトウェアと最適化コンパイラを用いて評価を行う．

第3章では，文献[51]に基づいて，まず新しいソフトウェアをマルチメディア端末に動的に追加して，新サービスに対応する際に生じる問題点について述べる．前章で提案した

プロセッサを適用して主要な端末機能をソフトウェア部品化すれば，ネットワーク経由のソフトウェアダウンロードによる端末の機能拡張を実現できる．しかしながら，機能追加後のソフトウェア群が消費するハードウェア資源の総量が，実資源量を超えると，実時間応答性が損なわれるという問題が生じる．この問題を解決するために，各ソフトウェア部品の資源消費プロファイルと，サービスにおけるソフトウェア部品の並行実行度から，機能追加後の資源消費量を予測するためのフレームワークを与える．その上で，画質等の処理品質を変えると消費資源量も変化することに着目し，処理品質をサービスの並行性に応じて動的に制御することにより，実時間応答性を保証する方式を提案する．さらに本設計手法に基づくマルチメディア端末を試作し，初期設計に対して複数のサービス機能追加を試行し，その実現性と有効性を評価する．

第 4 章では，文献 [56] [57] [58] [59] に基づいて，まずサービス提供側の中心となるマルチメディアデータベースサーバにおける性能要件について議論する．この中で，サービス利用者による定型的な検索と，サービス企画者による大規模で非定型的な検索の両方に対して十分な応答性能が要求されることに焦点を当てる．定型的な検索に対しては，従来から用いられている索引付けによる高速化が有効であるが，非定型的な検索を同様の手法で高速化しようとすると，データ量の増大と更新性能の低下を招き，現実的でない．次にこの問題を解決する関係データベースサーバの高速化方式を提案する．具体的には，索引を利用できない非定型的な検索に対して，大容量の主記憶に必要なデータをベクトル形式で動的に展開し，データベース演算専用のプログラマブルプロセッサでパイプライン処理することで処理効率を向上させる．さらに本プロセッサを LSI 実装し，関係データベース管理システムに適用して性能評価を行う．

第 5 章では，結論として本研究で得られた成果を要約し，今後の課題を述べる。

第2章　実時間マルチメディア処理向けシステムオンチップ

2.1 緒言

本章では，文献[47]に基づいて，メディアプロセッサのソフトウェア生産性を向上させる方式を提案し，LSIとして実装してその有効性を評価する．

音声や映像のデジタル化技術は急速に進歩しており，そのコア技術である映像の圧縮方式の変化も予想以上に早く，新方式が次々と開発されている[60]．コア技術の変化が早く，すぐに陳腐化する状況は，その技術を実現するコンポーネントの開発者のみならず，サービスとそれを支えるシステムの設計者にとっても大きなリスク要因となる．例えばモデムや音声の場合は，DSP (Digital Signal Processor) 上のソフトウェアを入れかえることで，新規格に迅速かつ柔軟に対応できるようにし，このリスクを低減している[61]．

映像の場合もモデムや音声と同じ手法での解決を図るべく，デジタル映像処理の実装をソフトウェアで行うことを可能とする，マルチメディア処理専用のプログラマブルプロセッサ（以下メディアプロセッサと呼ぶ）が提案されている[62]．しかし，メディアプロセッサの性能を十分に発揮させるためには，プログラミングにアセンブラ言語の使用が必要であり[33]，そのソフトウェア開発の生産性の低さとともに，変更が困難で再利用が難しいという問題がある．結果として，メディアプロセッサの存在意義であったはずの柔軟性を大きく損ない，デバッグに大きな工数を要するため製品化が遅れる等の困難に直面し，ASIC (Application Specific Integrated Circuit)方式[20]に対する明確な優位性を築くことはできない．

メディアプロセッサのプログラミング効率の問題を解決すべく，いくつかの最適化 C コンパイラの開発が開始されている[29] [31]．しかしながら，コンパイラを適用して十分な性能を得るためには，メディアプロセッサのアーキテクチャ自体もコンパイラによる最適化を支援するよう設計されている必要がある．したがって後付けのコンパイラ設計では最適化コンパイラの開発は困難であり，開発結果は依然ユーザを満足させるに至っていない．

本章では，C コンパイラの最適化可能性を分析評価しながらアーキテクチャを設計することにより，この問題を解決するメディアプロセッサ HMPV (High-performance Media Processor based on VLIW) を提案する．本プロセッサは，演算処理においてはビット列処

理用 RISC プロセッサと 2 次元画素配列処理用の VLIW プロセッサとを組み合わせることで，またデータ転送に対してはデータキャッシュと外部メモリ間の画素データ転送を専用プロセッサで効率化することにより，C 言語でのプログラミングでも十分な性能を発揮する．

以下，2.2 節では従来のメディアプロセッサアーキテクチャの問題点を分析する．2.3 節ではそれを解決する方式について述べる．2.4 節ではハードウェアの実装結果について述べ、2.5 節ではアーキテクチャ性能の評価結果を示し有効性を明らかにする．

2.2 従来アーキテクチャの問題点

2.2.1 メディアプロセッサの基本構成

一般にデジタル映像処理は，データ型と演算種類が異なる次の 4 つの処理から構成される [63]．

① ビット列処理（ハフマン符号化，復号化等）

② 8/16 ビットの 2 次元配列演算（画素の積和等）

③ 2 次元データの転送処理（画面データ等）

④ 外部入出力処理（音声，映像等）

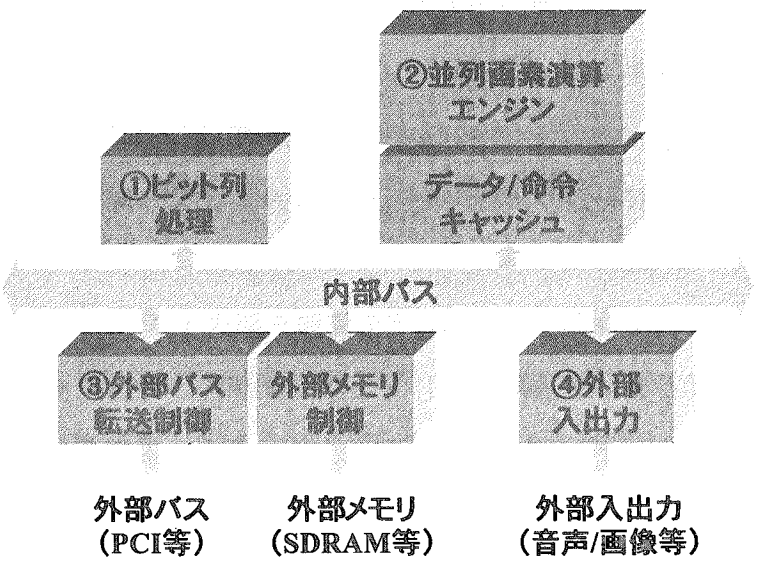

図 2.1 メディアプロセッサの一般的構成

従って，メディアプロセッサでは上記のそれぞれに対して処理ブロックを割り当て，並列に処理する方式が高速化に有効であるため，図 2.1 に示すようなブロック構成が基本となっている．

2.2.2 メディアプロセッサの設計手法

図 2.1 に示すようなメディアプロセッサのアーキテクチャ詳細設計は，以下の手順で行われる．

① ターゲットとするデジタル映像処理応用をベンチマークとして設定する．
② 必要な処理量を評価し，ピーク値をカバーできる演算処理系を設計する．
③ 必要なデータ転送量を評価し，ピーク値をカバーできるデータ転送系を設計する．
④ ターゲット応用の核部分のトライアルコーディングを行い，機能と性能の十分性を確認する．
⑤ LSI としての実装評価を行い，クロック周波数目標と面積目標を満たすことを検証する．

C 言語のような高水準言語でプログラミングしても十分な性能が発揮できるアーキテクチャを得るには，C コンパイラによる最適化容易性を設計の各段階で評価しながらアーキテクチャの詳細化を進めることが不可欠である．以下，この考えに基づき，前記②-④の各設計ステップにおける設計課題について述べる．

2.2.3 演算処理系の課題

いくつかのメディアプロセッサは，ハフマン符号化/復号化，および VLD/VLE (Variable Length Decoding / Variable Length Encoding) といった，JPEG, MPEG 系のビット列処理アルゴリズムに対象を絞ったビット列処理エンジンを備えている[29]．この方式は回路規模面からは最適な構成を取れる反面汎用性に乏しい．特に，新規格に対する C 言語でのプログラミングによる迅速な対応，という本章における目標を達成するのは困難である．

一方，画素演算系では，命令レベルの並列性と画素配列レベルの並列性をそれぞれコンパイラがどう抽出するかが焦点となる．命令レベルの並列化を可能とするアーキテクチャとしては，スーパースカラー[64]と，VLIW (Very Long Instruction Word) [65]がある．命令レベル並列性をハードウェアで動的に検出するスーパースカラーは，コンパイラ設計は

VLIW より容易であるが，並列度を上げようとすると，回路規模が急速に大きくなる．このため，高い並列度と低コストの両立が必須であるメディアプロセッサでは適用が難しい．

VLIW はコンパイラが命令レベル並列性を静的に抽出するため回路規模は小さく，コスト面は優位であるが，命令長が長いためプログラムサイズが大きくなる．特に画素演算では，積和を中心とする演算の種類，8/16/32 ビットといったデータの種類，計算誤差最小化のための丸め処理の種類等，豊富な命令群が必要となる．また，コンパイラが適切な並列化を行うためには，命令内に種々の制御指定フィールドが必要となる．例えば，プレディケーションにより分岐を高速化するための指定ビット，ループアンローリングによる並列化に対応可能な十分なレジスタ数の指定等が必要であり[66]，これらを確保しつつ命令長を 32 ビットに収めることは困難である．従って命令形式の設計は重要な課題である．

一方，画素レベルの並列化には，数値計算向けに開発された，配列演算の並列化コンパイラの手法[67]が適用可能である．上述のようなコンパイラ最適化のために十分な制御情報とレジスタリソースが与えられれば，要素レベルの並列化自体には大きな困難はないと考えられる．数値計算向けコンパイラは並列度の向上を最優先するために，目的プログラムのサイズが大きくなる傾向がある．組み込み応用向けコンパイラでは，ループアンローリング回数の上限をプログラマが指定するといった，プログラムサイズと並列度のトレードオフを調整する機能が有効となる．

2.2.4 データ転送系の課題

C 言語等の高水準言語によるプログラミングの生産性が高い理由は，仮想化により物理リソースの制約からプログラマを解放することが大きい．データ転送に関しては，データキャッシュにより，プログラマは演算器の近くにおかれた小さく高速なローカルメモリパッドにデータを予め明示的に転送しておく，等の苦労なしに比較的最適化されたデータ転送を外部メモリとプロセッサチップの間で行うことができる．

しかしながら，映像データの処理においては，データキャッシュによる転送最適化は期待できない．以下，図 2.2 を用いてその理由について説明する．

映像データは画面スクリーンに対応した画素の 2 次元配列として保持され，スクリーン上の小領域（例えば 8 画素×8 画素の画素ブロック）の各画素に対しある演算を施す，という操作を各画素ブロックに対して繰り返す，というような処理がデジタル映像処理の中心であることが多い[63]．画素ブロックデータの外部メモリからの転送を従来型のデータキ

ャッシュを用いて行うと，データ量が数 MB でかつ処理する画素のアドレス局所性の小さいことから，キャッシング効果は低い．

例えば画素ブロックの行（水平）方向にアドレスが連続しており，各画素が 2B のデータサイズであるとすると行全体は 8 画素で構成されているので 16B となる．図 2.2 に示すように，今この 16B 中のどこか 1 箇所でデータキャッシュラインをまたいでいる確率を α とすると，8 画素の処理中に α 回のライン転送がおきることになる．次の行に移る時は必ずライン転送が起きるとすると，結局 1 画素ブロック処理中に $8\times(1+\alpha)$ 回のライン転送（キャッシュミス）が生ずる．ヒット率は $1-(1+\alpha)/8$ となり，α=1 の時はヒット率 75% と，ブロッキング効果以外は期待できない．キャッシュミス時のライン転送時間等のオーバヘッドを β サイクルとし，1 画素当たりの必要演算サイクル数を γ サイクルとすると，1 画素ブロックの処理には $(16\times\beta+64\times\gamma)$ サイクルを要する．通常 β が 20～50 程度であることを考えると，データ転送時間に要する時間が 320～800 サイクル，演算器を並列化して γ を各画素当たりの必要演算サイクル数を 2～5 サイクル程度まで小さくして演算時間を 128～320 サイクルとすると，データ転送ネックとなることが明らかである．

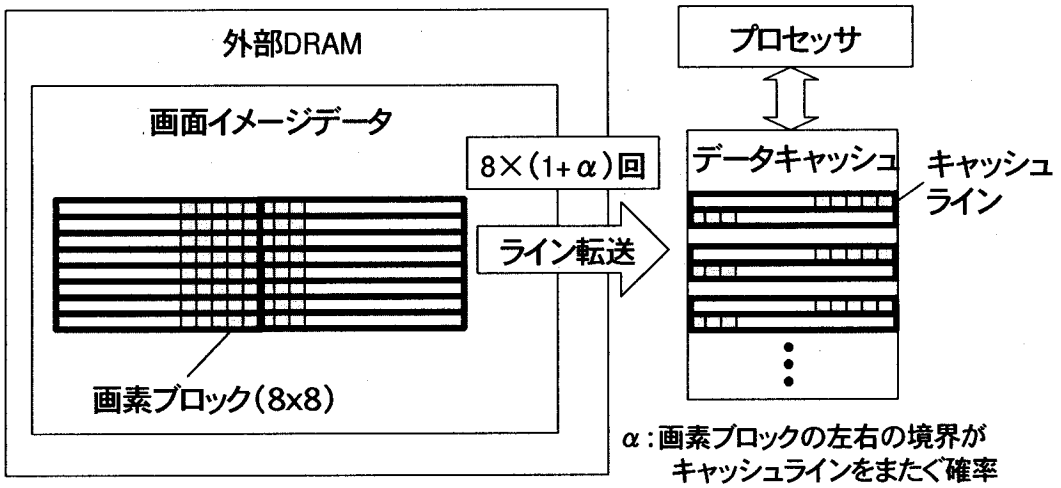

図 2.2 画像処理におけるデータのキャッシング

この問題を回避するために，多くの DSP やメディアプロセッサはデータキャッシュではなく，プログラマが明示的にアドレッシングするオンチップローカルメモリを演算器の近くに配し，外部メモリ中の画像データをプログラマが必要に応じてこのローカルメモリに転送して一時記憶領域として利用できるように設計されている[31]．数値計算向けスーパーコンピュータのベクトルレジスタも同様の目的で導入されたローカルメモリである[68]．プログラマがこれらのローカルメモリを管理してスケジュールされたデータ転送を行うことにより，キャッシュミス時のライン転送による処理中断を避けることが可能となる．しかしながら，物理レジスタの使用を意識して行うアセンブラプログラミングと同様，このようなメディアプロセッサのプログラミングはローカルメモリ資源管理と，演算処理とサイクルベースで完全に同期した転送スケジューリングをローカルメモリの物理アドレスを意識して行う，ローレベルで高いスキルを要するプログラミング作業とならざるを得ず，高水準言語の使用によるソフトウェア生産性向上は困難である．

2.3 VLIW による実時間マルチメディア処理方式

2.3.1 全体構成

本節では，2.2 節で述べたソフトウェア生産性の課題を解決するメディアプロセッサ HMPV を提案する．演算処理系の課題は，命令セットアーキテクチャと命令コードの圧縮方式により解決する。これについては 2.3.2 項で述べる．またデータ転送系の課題については、2.3.3 項で説明するように，データキャッシュと外部メモリ間のデータ転送を制御するプロセッサを導入することにより解決を図る．

HMPV の全体構成を図 2.3 に示す．HMPV では，デジタル映像の基本処理である，①ビット列処理，②画素演算処理，③データ転送，に対して，それぞれ最適な命令セットアーキテクチャを有するプラグラマブルプロセッサを割り当て，並行処理する方式を採用している．外部メモリバスとしては，128M バイトまでの外部 SDRAM に対応する 64 ビット幅のメモリコントローラが用意されている．システムオンチップ化を進めるため，周辺 I/O 回路は映像/音声の入出力を持ち，色々なディスプレイに画像を出力するための画像拡大/縮小を行うビデオフィルタを備えている．これにより，外付けの IO チップなしに STB を始めとする多くの応用システムを構築することが可能となる．また，PCI のような汎用の外部バスを経由して転送される映像等のコンテンツデータに対しては，暗号化/復号化処理[69]も可能である．

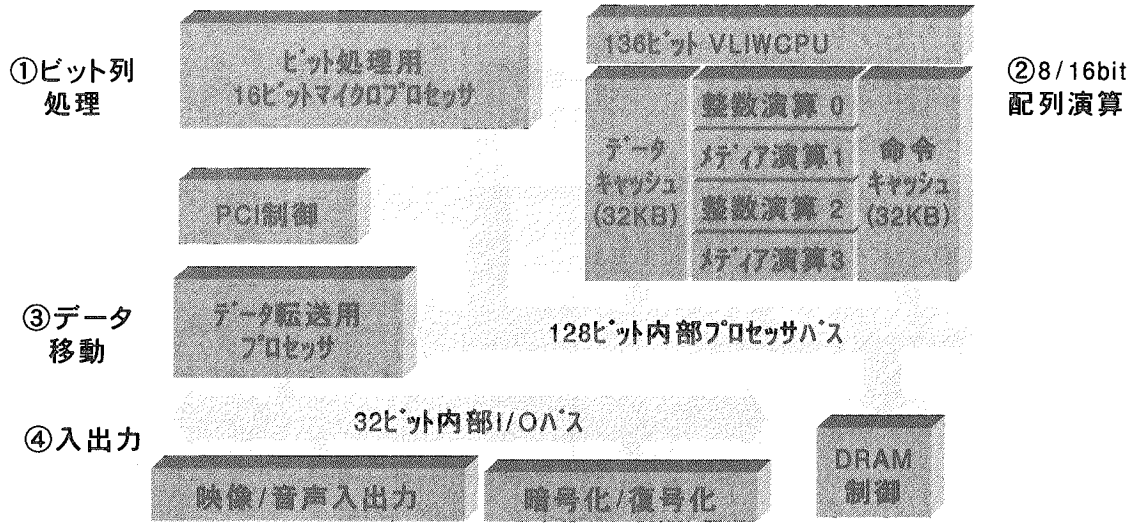

図 2.3 HMPV のブロック図

2.3.2 演算処理系

ビット列処理は汎用 16 ビット RISC マイクロプロセッサコアによって行う．このプロセッサは通常の RISC 命令に，ビットストリームの切りだしやアラインメント等のビット処理命令を付加したものである．汎用プロセッサとそのコンパイラをベースに拡張することで，新規格への対応力を強化しつつ，C コンパイラの適用を容易化している．ビット処理用プロセッサの内部構成を図 2.4 に示す．

画素演算処理には図 2.5 に示すように，VLIW アーキテクチャを用いた CPU コアを採用している．この CPU コアは，演算ユニット，32K バイトの命令キャッシュおよび 32K バイトのデータキャッシュから構成される．演算ユニットは 4 つの演算器から構成され，1VLIW 命令でこれらの演算器を並列動作させることができる．

32 ビット整数ユニットはアドレス変換機能を含む汎用 RISC マイクロプロセッサと同等の機能を持ち，汎用 OS が搭載可能である．これは，STB のようなコスト制約の厳しい家庭用機器に適用することを念頭に置き，汎用マイクロプロセッサが無くてもシステム構築可能なよう配慮したためである．

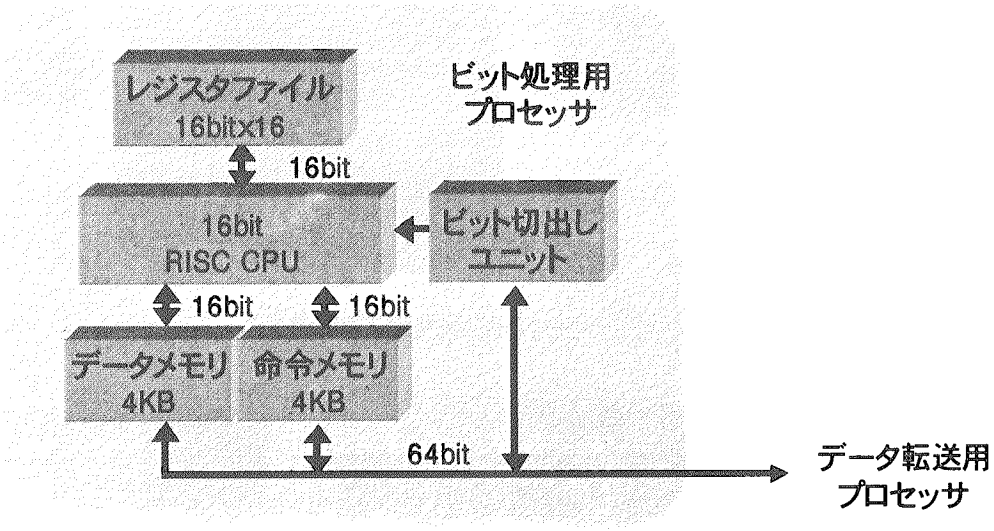

図 2.4 ビット列処理用プロセッサの構成

デジタルメディアデータの処理の中心となる 128 ビットメディア積和ユニットと 64 ビットメディアシャッフルユニットには 64 ビット／128 ビットレジスタを 8 ビット，16 ビットまたは 32 ビット単位に分割し，それぞれに対して同一の演算処理を並列に実行する強力な命令を持たせ，フィルタリング等の並列データ処理を高速化している．代表的な演算命令を以下に示す．

内積：$\sum_{n=0}^{n=7} a(n) * b(n)$　　(16 ビット)

SAD：$\sum_{n=0}^{n=15} | a(n) - b(n) |$　　(8 ビット)

シャッフル：　b(i) = a(j)　(i, j=0, …, 7)　　(8 ビット)

メディアシャッフルのような 64 ビット演算時は連続する 2 本の 32 ビット汎用レジスタ

を 64 ビットレジスタとして使用する．また図 2.5 に示したように，演算ユニットをクラスタ 0 とクラスタ 1 の 2 系統持たせているので，上記 64 ビット／128 ビット演算を 2 演算並列実行可能である．

2.2 節で述べたように，VLIW アーキテクチャでは長形式の命令語中に複数の演算器制御命令が含まれる．このため，動作しない演算器が存在する場合，その演算器の制御に対応する命令（NOP）をどう圧縮するかが課題となる．これを実現する方法として，例えば各演算器制御命令が NOP かどうかを示すフラグビットを，命令データとは別にまとめて保持する方法[65]などが知られている．

HMPV では命令が NOP か否かに加え，条件コードの値に応じて命令実行を抑止するか否か（プレディケーション）といった命令実行の制御を示す情報を独立に保持している．図 2.6 に示すように，命令空間を 256B から成る命令ページに分割し，各命令ページの先頭に命令の拡張部分として制御情報と VLIW 命令の区切りを示すフラグを格納し，その後に NOP を除いた命令の基本部を格納している．命令の基本部と拡張部を合わせて生成された命令パケットが一つの演算ユニットを制御する． さらに 4 つの命令パケットから一つの VLIW 命令

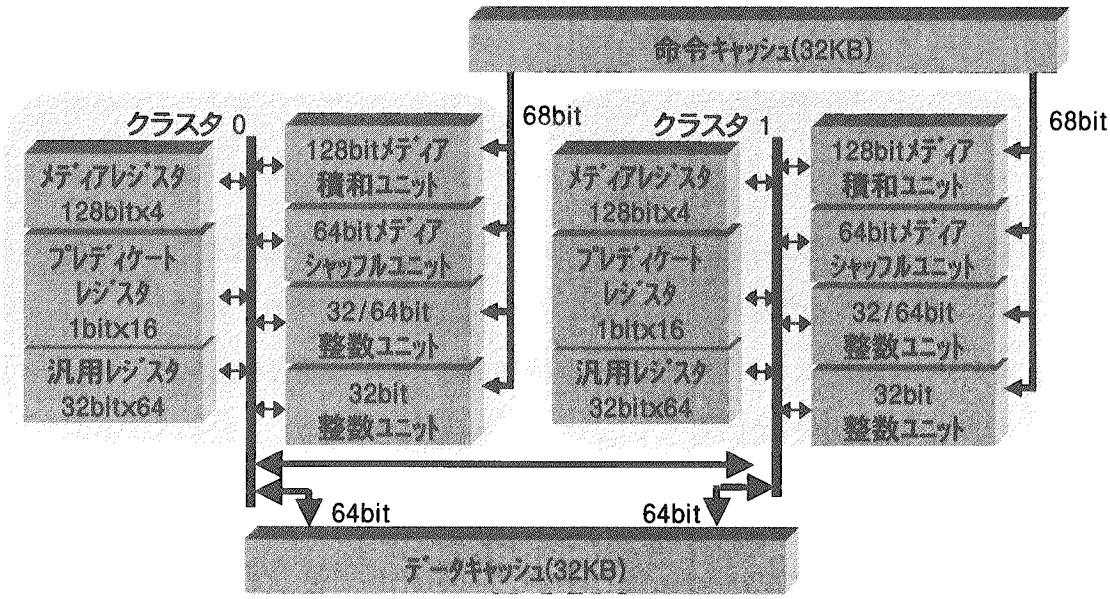

図 2.5 VLIW CPU の構成

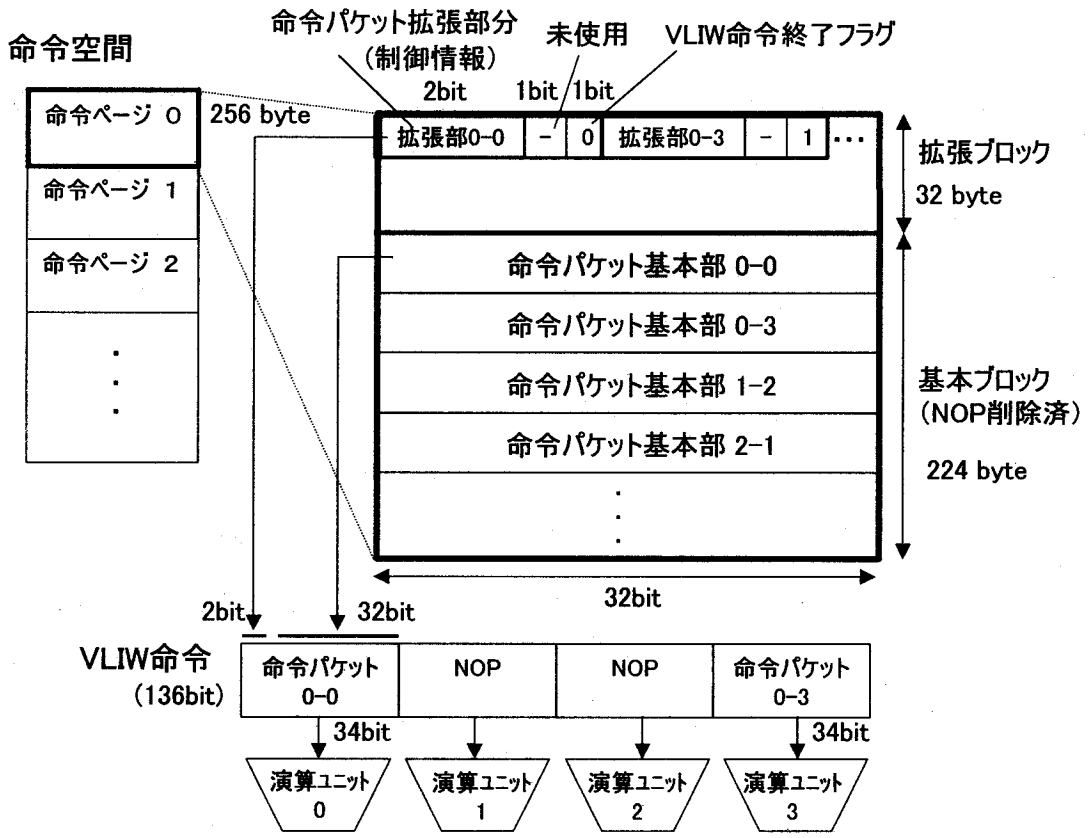

図 2.6 VLIW CPU の命令形式

が構成される．2.2 節で指摘したように，コンパイラによる最適化を支援するにはプレディケーションに代表されるような種々の制御フィールドが必要になり，命令フィールドの設計が難しくなるが，HMPV はこれらの命令制御情報を命令の拡張部分として別に格納することにより，この問題を解決している．

本方式の圧縮効果はプログラムにより異なり，長命令列中の NOP の割合が低い (1/8 未満) 場合は，拡張部のオーバヘッドにより，逆に命令キャッシュの利用効率が低下する可能性もある．実際のアプリケーションで評価したところ，すべてのプログラムで効果が認められた．例えば MPEG2 デコーダでは，圧縮後のプログラムサイズは，圧縮していない状態の約 57%とかなり小さくなっている．

2.3.3 データ転送プロセッサの構成と機能

データ転送プロセッサは，HMPV 上の異なるメモリ資源（外部 DRAM やデータキャッシュ，ビット列処理用プロセッサ内のメモリ等）間や， PCI デバイスや I/O デバイス間で，内部のバッファメモリを介して VLIW CPU とは非同期にデータを転送する一種のプログラマブル DMA (Direct Memory Access) エンジンである．データ転送の制御プログラムは専用のプログラムメモリに格納される．図 2.7 にデータ転送用プロセッサの内部構成を示す．

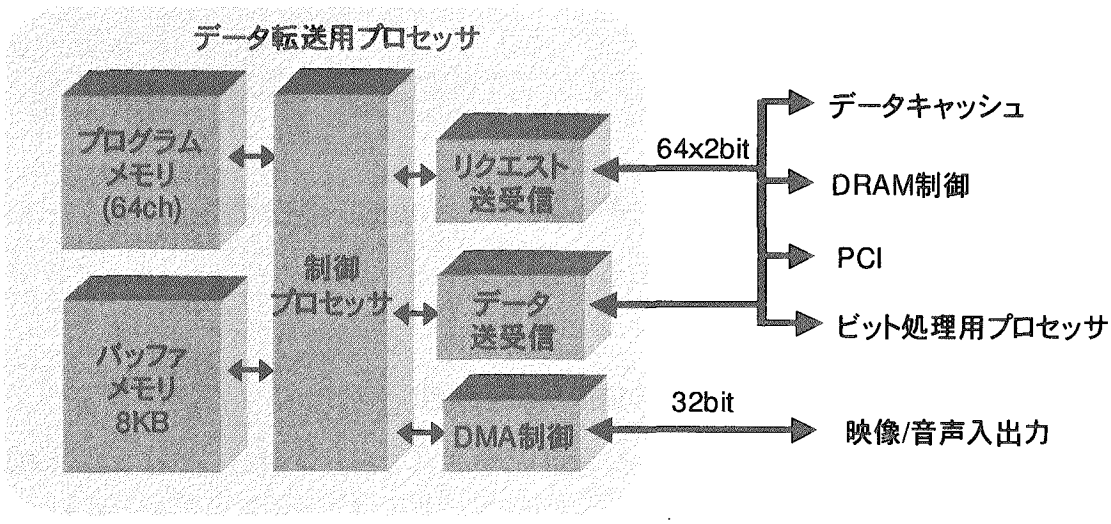

図 2.7 データ転送用プロセッサの内部構成

基本となる転送操作は，ある仮想アドレス空間上の 2 次元領域 Source (I, J) (I=0, …, a-1; J=0, …, b-1) を読み出し，別の 2 次元領域 Target (I, J) (I=0, …, c-1; J=0, …, d-1) に書き込むメモリ間転送である．同時実行可能な転送数は最大で 64 チャネルであり， Source と Target に対してそれぞれ表 2.1 に示すアクセスモードを指定することで，データキャッシュに対する動作を制御する．以下に各モードの代表的な使用方法を示す．

① 処理対象データのキャッシュへの集約 (Gather)

Target の大部分がキャッシュされていないとする．この時 Target について Coherently

表 2.1 データ転送プロセッサのアクセスモード

コヒーレント・モード	動作
Coherently Allocate	データキャッシュにデータを格納するアクセス.キャッシュミス時, データキャッシュに新たに該当キャッシュラインをアロケートする.
Coherently No-Allocate	データキャッシュを参照するがデータは新たには格納しないアクセス. キャッシュヒット時はキャッシュに対して, ミス時は外部メモリに対して読み書きを行う. キャッシュラインのアロケートは行わない
Non-Coherently	データキャッシュを一切参照しないアクセス.

Allocate モードを指定すると，データキャッシュに Target 領域が新たにアロケートされ，処理したい Source データが読み込まれる.

② 処理結果データのキャッシュからの拡散（Scatter）

処理結果の大部分がキャッシュにあるとする．これを Source として Coherently No-Allocate モードを指定すると，キャッシュヒットするデータはキャッシュから，ミスするデータは外部メモリから読み出されて外部メモリ上の Target 領域へ処理結果が書き出される.

③ キャッシュを経由しないデータ転送

Source または Target がキャッシュされていない時，Non-Coherently モードを指定すると，キャッシュミスを起こさずに直接外部メモリへの読み書きが行われる.

2.2.4 項で述べたように DSP やベクトルプロセッサでは，ローカルメモリやベクトルレジスタを Source や Target にした物理アドレスによる集約/拡散操作をサポートしている．本データ転送プロセッサはこれらと異なり，データキャッシュを使って仮想化された集約/拡散操作を実現する．これにより，ローカルメモリやベクトルレジスタ等の物理資源をプログラマが明に管理する必要をなくしている.

データ転送プロセッサのプログラミングは，VLIW CPU からの，①Source/Target の設定，②内部バッファのアロケーション，③転送の起動と停止，④転送状況の調査，等のライブラリ呼び出しを用いて行われる．前述したように Source と Target は仮想アドレス空間上の資源なので，必要な配列領域を自由に宣言して使用するという，高水準言語での普通のプログラミングスタイルが可能となる．

2.3.4 データ転送プロセッサの応用例

先に述べた画素ブロックの処理におけるデータ転送プロセッサの使用方法を図 2.8 により説明する．

まず，仮想アドレス空間上の 1 ブロック分の画素データを格納する小容量の作業領域を Coherently Allocate モードで Target として設定し，データ転送プロセッサで外部メモリ

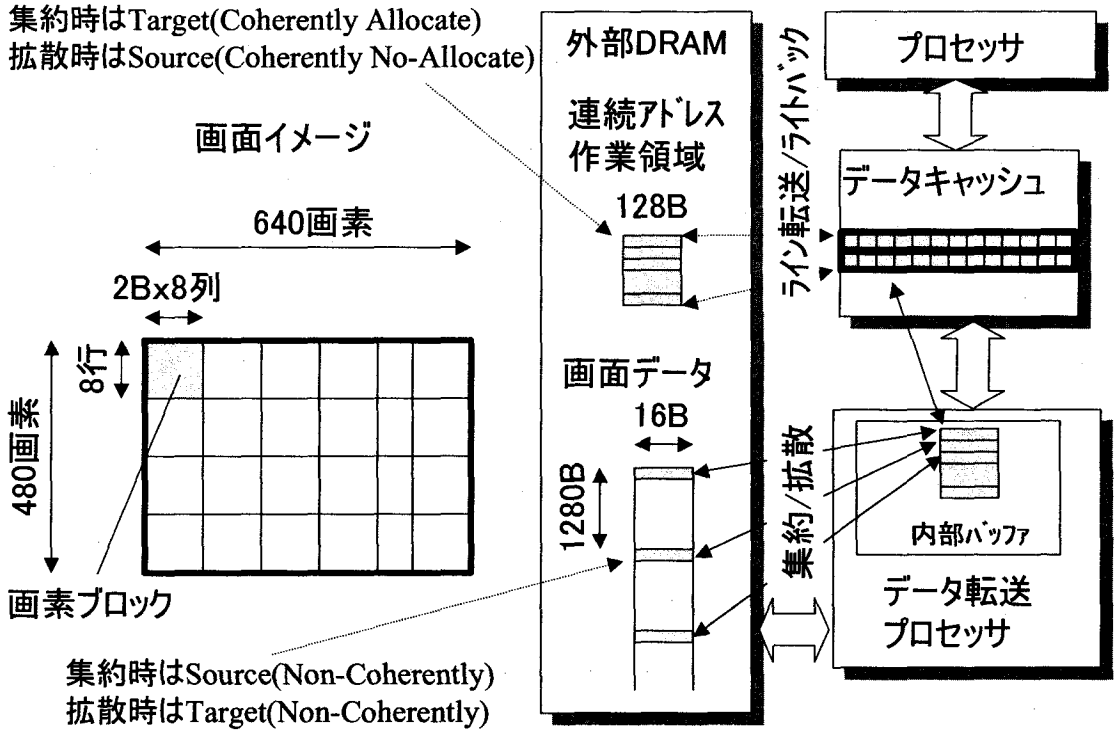

図 2.8 データ転送プロセッサの使用例

上の2次元画面イメージデータから1ブロックをSourceとして読出し，作業領域に対して転送する．これにより，データキャッシュ内にアロケートされた作業領域に画素ブロックデータがコピーされた状態となる．もとの画素ブロックの各行はアドレス的に離れているが，作業領域上では連続アドレスにパックされた形で集約される．

次に，この作業領域上の画素データに対してVLIW CPUで各種演算を施す．今度は逆にこの作業領域をCoherently No-Allocateモードでアクセスする Sourceとし，元の2次元画面イメージデータの画素ブロックをTargetにして拡散することにより，画面データを更新する．上記の集約と拡散のいずれにおいても，作業領域にアクセスする時はキャッシュのミスヒット時のみ外部メモリを参照しているため，データキャッシュとの一貫性が保証される．

一方，外部メモリ上の画面データは，VLIW CPU では直接処理しないのでキャッシングの必要はない．したがって常に No-Coherently モードでアクセスすれば，アドレス局所性が低くかつサイズの大きい画面データがデータキャッシュを汚染するのを防ぐことができる．

画素ブロック処理におけるデータ転送と演算処理のタイミングの例を図 2.9 に示す．ここでは作業領域としてA，B二つの領域を用い，この二つの作業領域をダブルバッファとして，それぞれ64画素からなるブロックデータ A0，B0，A1，B1，…を交互に集約，演算処理，拡散している．処理の流れ全体はVLIW CPUが制御する．　ある時刻例えばt2からt3にかけては，A1の集約と A0の拡散は並行して行われる．外部DRAMが1ポートとするとこれらは競合するので，実際には A1 の各画素を読み出す合間に A0 の各画素の書き込みを適宜メモリポートの空きを待って行う必要がある．内部バッファは書き込みが待たされた時にそのデータを一時保存するのに用いられる．

データ転送プロセッサはチャネル毎に転送状態を示すセマフォアビットを有し，VLIW CPU とデータ転送プロセッサはこのセマフォアにより，互いに排他制御を行う．VLIW CPUとデータ転送プロセッサがデータキャッシュ上の小容量作業領域を繰返しアクセスすることにより，確実にキャッシュヒットする．後述のデジタルTV向けMPEG2デコーダの例ではこのプログラミングスタイルにより，キャッシュヒット率ほぼ100%を達成している．

HMPVはシステムオンチップとして数多くの非同期に動作するプロセッサやIO系から構成され、各機能ブロック内のメモリ資源は単一のメモリアドレス空間にマッピングされている．従ってデータ転送プロセッサによるメモリ間転送は，外部メモリとキャッシュ間だけでなく，システムバスに接続された任意のブロック間で利用できる．例えば入力ビットス

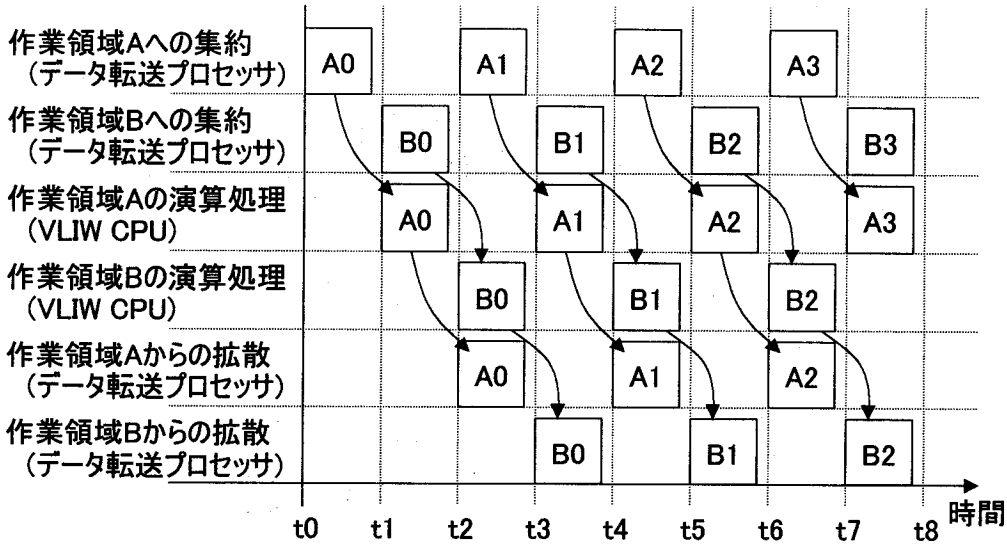

図 2.9 データ転送と演算処理の重畳

トリームをビット列処理用プロセッサに送って処理する，あるいは PCI 外部バスからのデータを直接データキャッシュに送って VLIW CPU で処理する，といった動作が可能となる．

2.4 ハードウェア実装

図 2.10 に 0.18μCMOS プロセスを用いて LSI としてハードウェア実装した結果を示す．本 LSI は 11M トランジスタから成り，最大動作周波数は 360MHz，最大消費電力は約 2.5W である．例えば，オーディオ (AC3)，ビデオ (MPEG2 MP@ML)，システム (OS 他) を含む DVD 再生アプリケーションの必要動作周波数は 120MHz であり，この時の消費電力は約 800mW と，コンシューマ応用に使えるコストと消費電力となっている．

図 2.11 に示すように，VLIW CPU と IO 回路を従来のメディアプロセッサと考え，命令圧縮，ビット列処理プロセッサ，データ転送プロセッサを加えたものを提案アーキテクチャと考えて比較すると 22%の面積増加となっている．

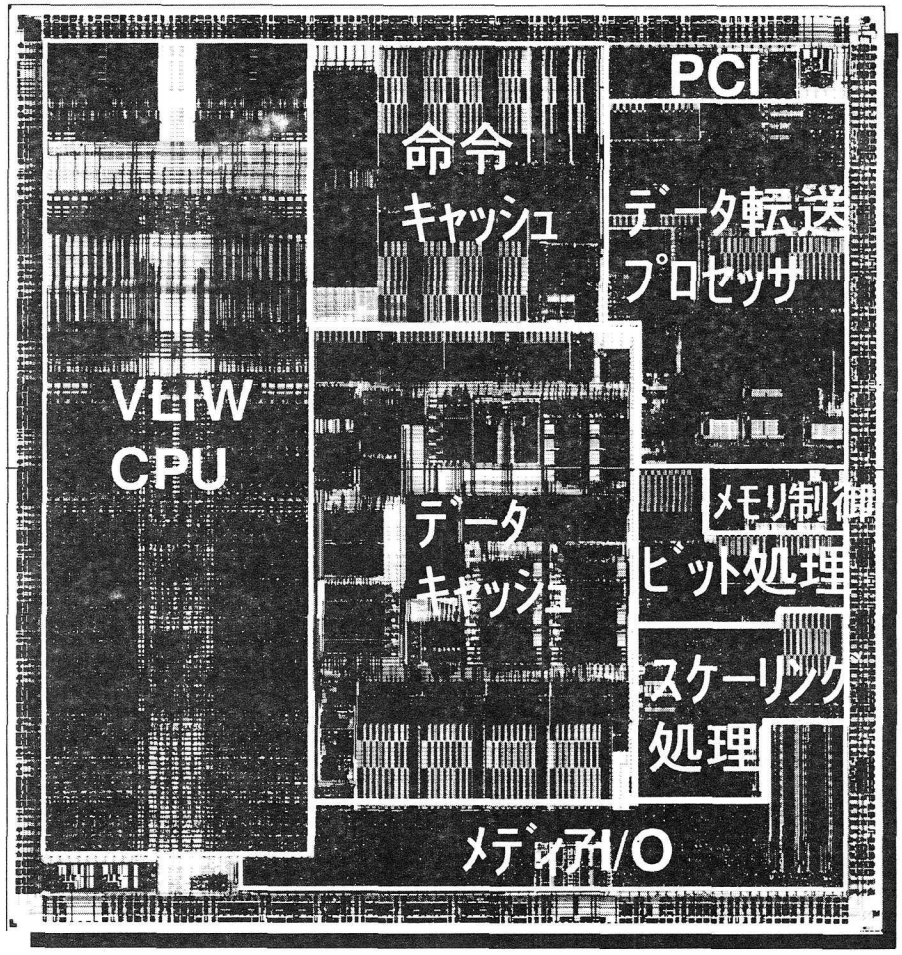

図 2.10 HMPV の実装

消費電力もほぼ面積に比例し，従来に比べ約 20%増加した．一方，VLIW CPU の従来クリティカルパスは分岐処理等で，そのサイクル時間は 2.8ns である．提案回路の付加後も LSI 全体のサイクル時間は変わらず，提案アーキテクチャはサイクル時間には影響を与えていない．

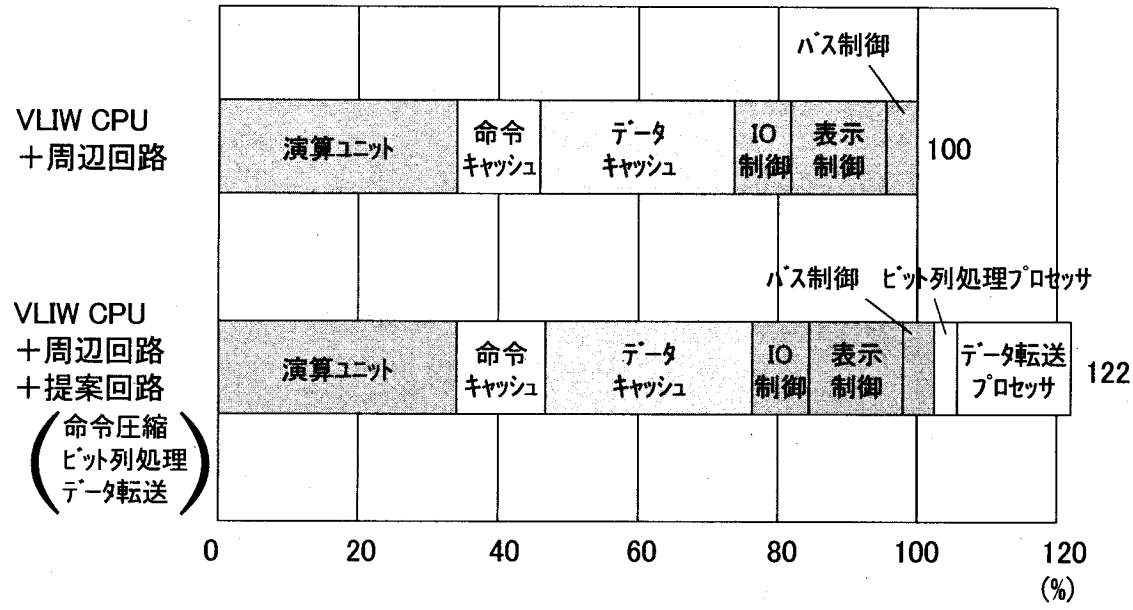

図 2.11 提案回路の実装面積オーバヘッド

2.5 性能評価

HMPV のアーキテクチャの評価を行うために，デジタル TV 向けの MPEG2 (MP@HL) デコーダソフトウェアを開発した．このデコーダソフトウェアは HDTV を含む任意のフォーマットをデコードして標準解像度の SDTV に出力する，AFD (All Format Decoder) であり，製品品質のものである [48]．高水準言語によるプログラミングで十分な性能を得るという設計目的が達成されたか否かを検証するため，プログラミングはすべて C 言語と最適化コンパイラを使用して行った．図 2.12 にその結果を示す．

図 2.12 の一番上の棒グラフは，VLIW CPU コアの汎用整数演算ユニットのみを用いて実行した場合の必要サイクル数（この場合 1300M サイクル）と，AFD の中での処理内訳が示されている．これは 4 命令を並列実行できるスーパースカラー型の汎用プロセッサでの実行に対応する．処理内訳から，動き補償における画素演算の負荷が高いことがわかる．

上から 2 番目の棒グラフはメディア演算ユニットの SIMD(Single Instruction Multi-Data) 演算を用いて，画素演算を並列化した結果を示しており，逆離散コサイン変換

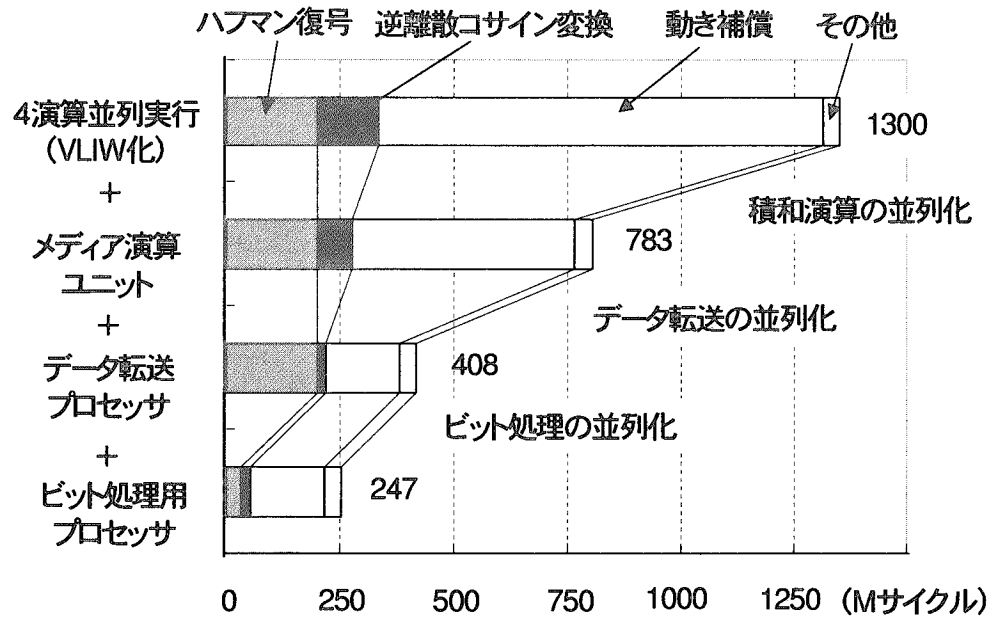

図 2.12 アーキテクチャ評価結果

や動き補償などの画素演算部分が高速化されているのが分かる．メディア処理の高速化を重視して SIMD 演算ユニットを搭載している汎用プロセッサやメディアプロセッサでの実行はこれに相当すると考えられる．

さらに上から 3 番目の棒グラフは，データ転送プロセッサを使って 2.3 節で述べたように，データキャッシュヒット率向上させると共に，演算とデータ転送を並列化した結果を示している．この段階でデータキャッシュヒット率はほぼ 100％を達成している．最後の棒グラフは，逐次ビット処理であるためこれまでほとんど高速化されていないハフマン復号処理をビット列処理用プロセッサにより，VLIW CPU と並行して実行した結果を示している．総サイクル数は 247M サイクルであり，VLIW と SIMD による従来のメディアプロセッサの性能を示す 2 番目のグラフ（783M サイクル）から約 3 倍向上している．

図 2.13 に AFD の主要なカーネルループをアセンブラ言語によるプログラミングにより，人手でさらに最適化した場合の，最適化 C コンパイラとの性能差を示す．アセンブラ化による性能向上は約 12%であり，最適化 C コンパイラでも実用上十分な性能が得られていることがわかる．

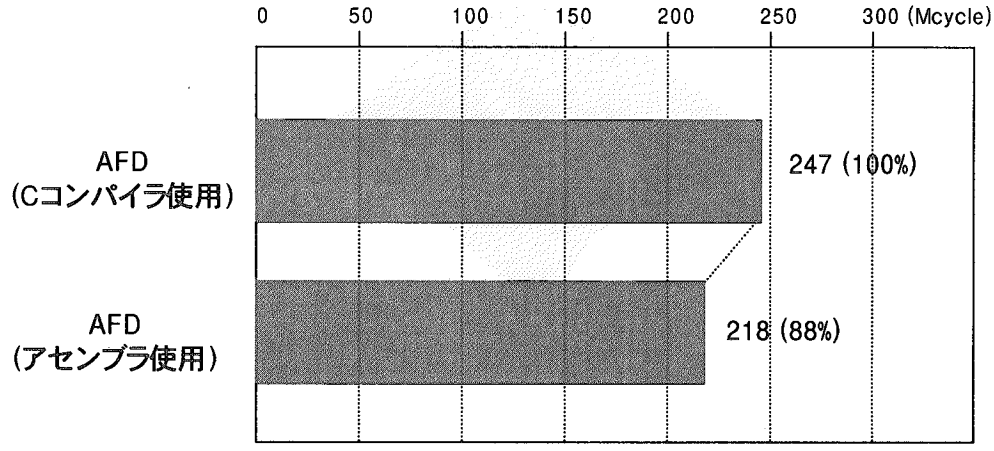

図 2.13　アセンブラ化による性能差

2.6 結言

ソフトウェアによるデジタル映像の実時間処理を目指したシステムオンチップとして，VLIW 型の画素処理プロセッサと，データキャッシュへのデータ転送を制御するプロセッサを中心とするアーキテクチャを提案した．LSI 実装とアプリケーション性能評価を行い，以下の結果を得た．

(1) 論理規模 11M トランジスタ，DVD 再生時の消費電力は約 800mW

(2) 従来アーキテクチャに対し 22%の面積増加で，約 3 倍の性能向上

(3) アセンブラ言語を用いた人手による最適化に比べた C コンパイラの性能差は約 12%．

以上，コンシューマ応用に適用可能なコストと，C 言語で十分な性能が得られることを確認し，提案アーキテクチャの有効性を実証した．

図 2.14 に提案アーキテクチャを応用した製品の一例を示す．提案アーキテクチャによる LSI は，100 万個以上が出荷され，TV 会議システム，デジタル STB，高機能プリンタ，医療用画像処理装置等の実アプリケーションに幅広く適用されている[70]．

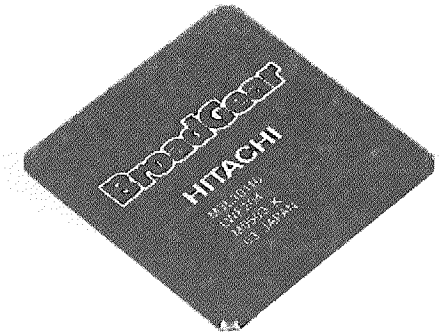

図 2.14 提案アーキテクチャ応用 LSI 製品の例

今後の課題として，以下があげられる．

(1) より多様なアルゴリズムに対する適用性の検証

本研究では MPEG 系の映像処理アルゴリズムにおける処理性能を中心に，有効性を評価しているが，ウェーブレット変換やフラクタルを用いる新しい処理方式も提案されており [16] [17]，これらに対する詳細な適用評価が課題である．

(2) 著作権保護と機器認証への対応

2.3.1 節で述べたように，HMPV は著作権保護のための暗号化と復号化用専用回路を有しているが，著作権保護の新しい手法や規格も次々と提案されており [71] [72]，これに対応する部分もソフトウェアで処理できることが望ましい．また，ネットワークに特殊な機器を接続してサービスを不正に受けるといった行為への対策として，機器認証の必要性が認識されており，プロセッサとしてこれにどう対応するかも課題である [73]．

(3) 入出力回路における柔軟性の向上　音声，映像のネットワーク接続インタフェースの規格も進歩しており [8]，ネットワークへの入出力のための周辺回路にも柔軟性が求められている．対応する方法としては，FPGA (Field Programmable Gate Array) 技術 [74] の適用などが考えられ，今後の研究課題である．

第3章　マルチメディア端末のソフトウェア設計手法

3.1 緒言

本章では，文献[51]に基づき，メディアプロセッサを用いたマルチメディア端末において，ソフトウェア更新後も実時間応答性を保証する設計方式を提案し，端末試作を通して有効性を評価する．

ADSL や FTTH に代表される家庭へのブロードバンド接続が急速に普及している．IP 電話，映像配信，高画質 TV 電話といったサービスが期待を集め、そのためのブロードバンド専用端末も盛んに開発が行われている[75]．これら音声や映像を中心とするサービスのコア技術である音声や映像のデジタル化技術も急速に進歩している．特に音声や映像の圧縮伸張方式の進化は予想以上に早く，圧縮効率が高く、より低ビットレートで高音質、高画質を実現する新規格が次々と開発されている[60]．

規格の変化が早く，すぐに新しい規格への対応が必要となる状況は，その規格を実現するコンポーネントの開発者のみならず，サービス開始に向けて早期に仕様を確定する必要のある端末の設計者にとっても大きなリスク要因となっている．また利用者から見ても新規格に対応したサービスを受けるためには新しい端末が必要となり、古い端末に対する投資が無駄になる。

新規格に迅速かつ柔軟に対応するため，デジタル映像処理をソフトウェアで実装することを可能とする，前章で提案したメディアプロセッサ HMPV を開発している[47]．これにより，初期設計時に想定していなかった規格への対応等、端末出荷後に機能を追加したい場合にも，新しい映像処理ソフトウェアを動的にダウンロードして対応できる．

端末ソフトウェアの動的な追加自体は，PC においてはユーザ責任によるアプリケーションソフトの追加として行われている．しかしながら，映像処理は高負荷であるため機能追加後の応答性保証は難しく，ユーザに端末ソフトウェアの管理を委ねた PC でのソフトウェア更新モデルは，誰にでも使えることが重要となる情報家電分野では適用が困難である．

本章では，メディアプロセッサを活用したブロードバンド端末におけるソフトウェア設計手法を提案する．本手法の目的は，端末ユーザにソフトウェア管理の負担をかけずに端末機能の追加変更を可能とすることにある。これを実現するために，端末機能を実装する各

ソフトウェア部品の消費ハードウェア資源量と同時実行可能性をデータベースで管理する。機能の追加変更時にはこのデータベースを参照して必要な資源量を計算し、不足する場合には各ソフトウェア部品の使用資源量を制御して実時間応答性を保つ.

以下，3.2 節では従来のブロードバンド端末におけるソフトウェア設計手法とその問題点について述べ，3.3 節で消費資源量の管理に基づく解決手法を提案する．3.4 節では適用例として機能追加型のデジタルセットトップボックスの試作について述べ，3.5 節で提案手法の評価結果を報告する.

3.2 従来の端末機能追加手法の問題点

従来のブロードバンド端末の代表的な開発フローを図 3.1 に示す．製品仕様決定後は開発期間短縮を目的としたハードウェア，ソフトウェア並行設計により，通常 1-2 年の期間で開発されている [34].

製品出荷後に発生するバグ修正などの機能追加変更要求には，更新されたソフトウェアを

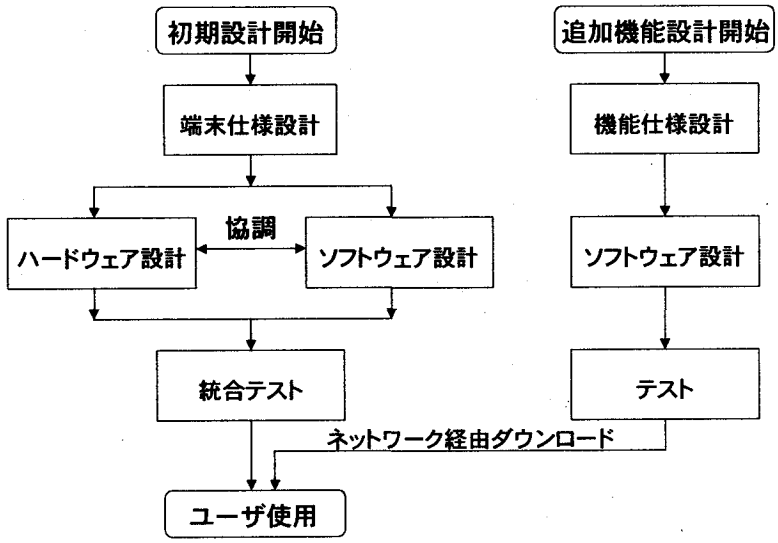

図 3.1 情報端末の開発フロー

ネットワーク経由でダウンロードすることにより対応している[76]．従来のブロードバンド端末は，汎用のマイクロプロセッサを制御用に用い，処理負荷の高いデジタル映像処理は専用 LSI である ASIC (Application Specific Integrated Circuits) を使用している．この構成では，マイクロプロセッサ上で動作している制御ソフトウェアは更新できるが，映像処理部は機能固定のハードウェアであるため機能の追加は困難である．以下では，映像処理をソフトウェア化して機能追加を可能とするため，前章で述べた HMPV を使用することとする．従来の端末設計手法にメディアプロセッサを適用すれば，音声や映像処理といった各端末機能をメディアプロセッサ上のソフトウェア部品によって実現できる．したがって，端末出荷後も単なるバグ修正にとどまらず，ソフトウェア部品のダウンロードにより新たな映像サービス機能を追加することができる．

ここで問題となるのは，従来の設計手法では新しいソフトウェア部品の追加変更後に端末の実時間応答性を保証することが困難な点である．機能の追加変更後に並行動作するソフトウェア部品群が消費するハードウェア資源量が，端末が有しているハードウェア資源を超えると応答性が低下し，音声や映像が途切れる等の破綻を生ずる．実際映像処理を汎用プロセッサ上のソフトウェアで行っている PC の場合には，新しいソフトウェアのインストール後に実時間応答を保証できていないケースが多くみられる．PC と異なりより幅広いコンシューマによる家電品的な使用を前提とするブロードバンド端末では，実時間応答の保証は必須の要件と考えられる．

3.3 資源プロファイル管理による実時間応答性保証方式

3.3.1 基本的な考え方

本節では，3.2 節で述べた問題点を解決するブロードバンド端末設計手法を提案する．本設計手法の目的は製品出荷後の映像サービス機能の追加においても，実時間応答性を保証することにある．

まず従来手法と同様に，端末設計を初期設計と変更設計の二つのフェーズに分けて考える．初期設計では統合テスト後に，端末のソフトウェア構成情報と使用している各ソフトウェア部品がハードウェア資源をどう使っているかを示すプロファイル情報をデータベース化しておく．

一方，変更設計フェーズでは初期設計において得られたこのデータベースを参照して機

能追加後の資源使用量を評価する．どれかのハードウェア資源を過剰に使用すると予測される場合，実時間応答性を保証できないので，ソフトウェア部品の使用資源量等を調整してこれを解消する．この結果を資源プロファイルに反映するとともに，ソフトウェアのダウンロード後は、新しいソフトウェア構成にしたがって端末を動的に再構成する．この設計フローを図 3.2 に示す．

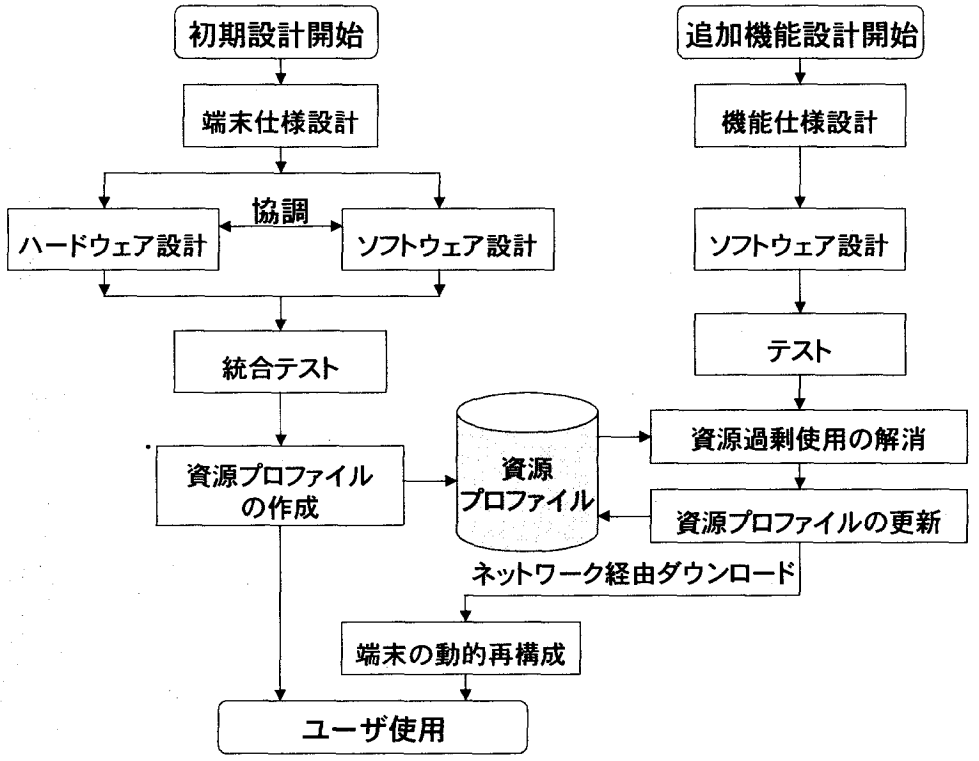

図 3.2 提案手法

3.3.2 資源プロファイル

以下の議論では，端末は複数のサービスを提供し，サービスは複数のアプリケーションから構成され，さらに各アプリケーションは複数のソフトウェア部品に分解されるものとする．この端末ソフトウェアモデルのもとで，各サービスにおける消費資源量を評価する

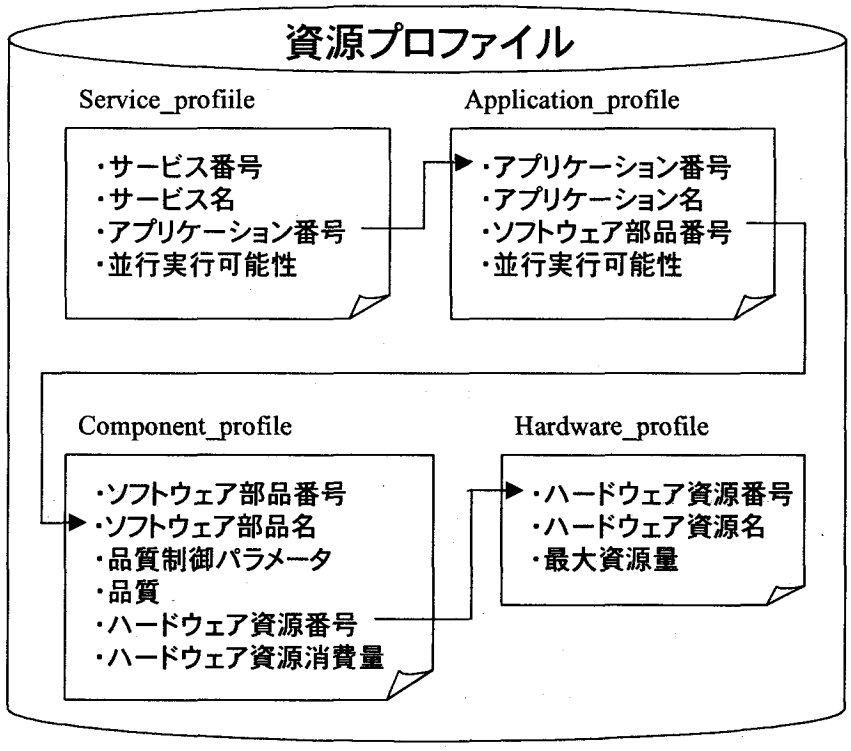

図 3.3 資源プロファイルデータベースの構成

には，ソフトウェア部品の資源消費量に加えて，アプリケーションとソフトウェア部品の実行多重度が必要となる．これらを含む資源プロファイルの構成例を図 3.3 に示す．

資源プロファイルのうち，Service_Profile は各サービスにおけるアプリケーションの並行実行可能性を，同様に Application_Profile は各アプリケーションにおけるソフトウェア部品の並行実行可能性を保持する．すなわち，サービス番号を i，アプリケーション番号を j，ソフトウェア部品番号を k とすると，次のように定義される．

Service_Profile(i, j) =

0: i 番目のサービスは j 番目のアプリケーションを実行しない．

1: i 番目のサービスは j 番目のアプリケーションを実行する．

Application_Profile(j, k) =

0: j 番目のアプリケーションは k 番目のソフトウェア部品を実行しない.

1 : j 番目のアプリケーションは k 番目のソフトウェア部品を実行する.

上記二つの並行動作情報は，仕様設計から得られる端末の機能構成を表現するものである.

Component_Profile は各ソフトウェア部品がある品質を実現するのに消費するハードウェア資源量を示す．これは音声や映像を処理するソフトウェア部品は，音質や画質といった品質に関わるパラメータを制御すると消費資源量が変化するためである.

Component_Profile(k, p, q, l) =

k 番目のソフトウェア部品の品質制御パラメータが p で得られる品質が q である時の，m 番目のハードウェア資源の消費量.

この Component_Profile(k, p, q, m) は机上評価あるいは実走行状態のモニタリングによって得られる．ピーク消費量に対しては机上評価が、平均消費量には実機のモニタリングが適している.

Hardware_Profile は端末に用意されているハードウェア資源量の情報である.

Hardware_Profile(m) =

端末で実装されているメディアプロセッサの m 番目のハードウェア資源の最大量

図 2.3 に示した HMPV の場合，VLIW CPU，ビット列処理プロセッサ，データ転送用プロセッサの消費サイクル数やバスのデータ転送量，外部 DRAM の使用量といったハードウェア資源に対するプロファイルが重要である.

資源プロファイルの情報を用いて，i 番目のサービスにおける m 番目のハードウェア資源の消費量 Consumption(i, m) は，並行実行されるソフトウェア部品の資源消費量をすべて加算することにより以下のように近似的に計算できる.

$$
\begin{aligned}
&\mathrm{Consumption}(i,m) = \\
&\quad \Sigma_j [\ \mathrm{Service\_Profile}(i,j) \\
&\qquad \times \Sigma_k [\ \mathrm{Application\_Profile}(j,k) \\
&\qquad\quad \times \mathrm{Component\_Profile}(k,p,q,m)\ ]\ ]
\end{aligned}
$$

3.3.3 消費資源量の調整

Consumption(i, m)が実装されているハードウェア資源量 Hardware_Profile(m)を超えた場合にはリソース競合により，実時間応答性が失われる．これを防止するためには，並行性や品質を落としてハードウェア資源消費量を抑える必要がある．したがって実時間応答性を保ってi番目のサービスに対して端末のソフトウェア構成を最適化するとは，「すべてのmに対してConsumption(i, m)≦Hardware_Profile(m)を満たしつつ各ソフトウェア部品の品質できるだけ大きくなる解を探す」という多目的最適化問題となる[77]．

もし，どのソフトウェア部品も品質を制御できないとすると，サービスに意味がある範囲で並行実行されるアプリケーション数を減らす以外に，実時間性を保証する方法はない．資源の最大量をナップザックの容量，各アプリケーションにおける消費資源量を荷物の体積と考えればこれはナップザック問題[78]と等価となる．したがってNP完全であり，同時実行アプリケーション数が少ない場合は総当りで，そうでない場合には遺伝的アルゴリズムなどのヒューリスティックな手法で解を求める[79]．また予め各アプリケーションに優先順位を与えておくことにより問題を単目的化することが可能な場合もある．消費量が実装量を超過する際には，優先度の低いアプリケーションから順に実行を抑止すれば良い．

一方，品質の制御が可能な場合には，選択肢の数は全品質制御パラメータの組み合わせとなる．さらに映像圧縮時のビットレートのように，品質とそれに対応する資源消費量は離散的でなく連続的に変化するものもあり，この場合総当り計算ではない多目的最適化方式が必要となる．本節ではそのような方式の例について述べる．最適化の基本方針として，図 3.4 (a)に示すような各ソフトウェア部品の品質がバランスした解を，(b)のような解よりも優位として探索することにする．

品質の定量的評価手法としては，主観評価と客観評価が知られており，国際標準を含む種々の規格が与えられている[80]．以下では簡単のため，表 3.1 に示すような主観評価を用いて品質を定義する．

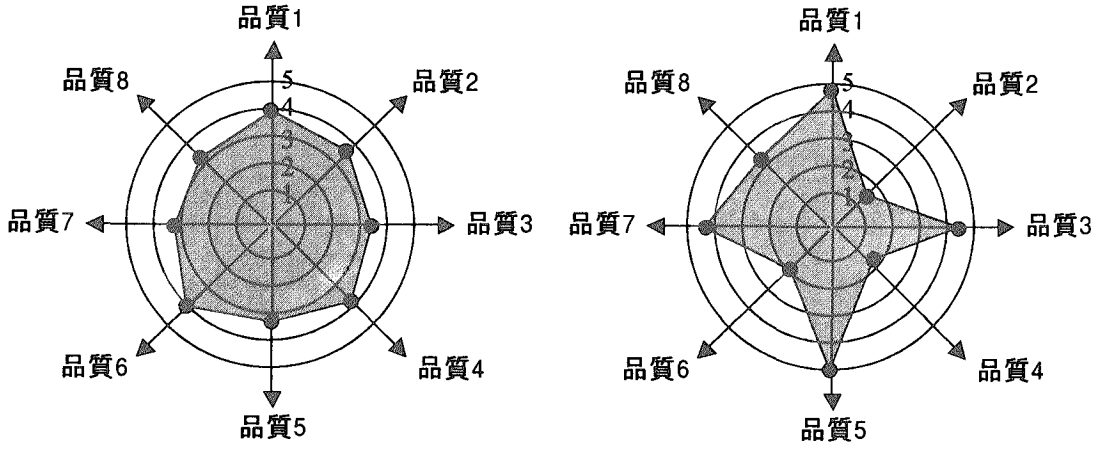

図 3.4 . 最適化の方針

表 3.1　品質の定義

品質	意味
5	非常に良い (オリジナルの品質を忠実に再現)
4	良い (専門家は品質の低下に気付く)
3	普通 (一般ユーザも品質の低下に気付く)
2	やや悪い(顕著な品質低下が認められる)
1	悪い(内容の理解が何とか可能なレベル)

ソフトウェア部品に関しては，適切な実装を行えば，品質は消費資源量の増加に対して単調に増加する．この増加曲線を直線で近似して，図 3.5 に示すように，各ソフトウェア部品の品質が等しくなるように資源を初期配分する．その後，資源の余裕があればさらに最適化する．具体的には以下の手順で品質を決定する．

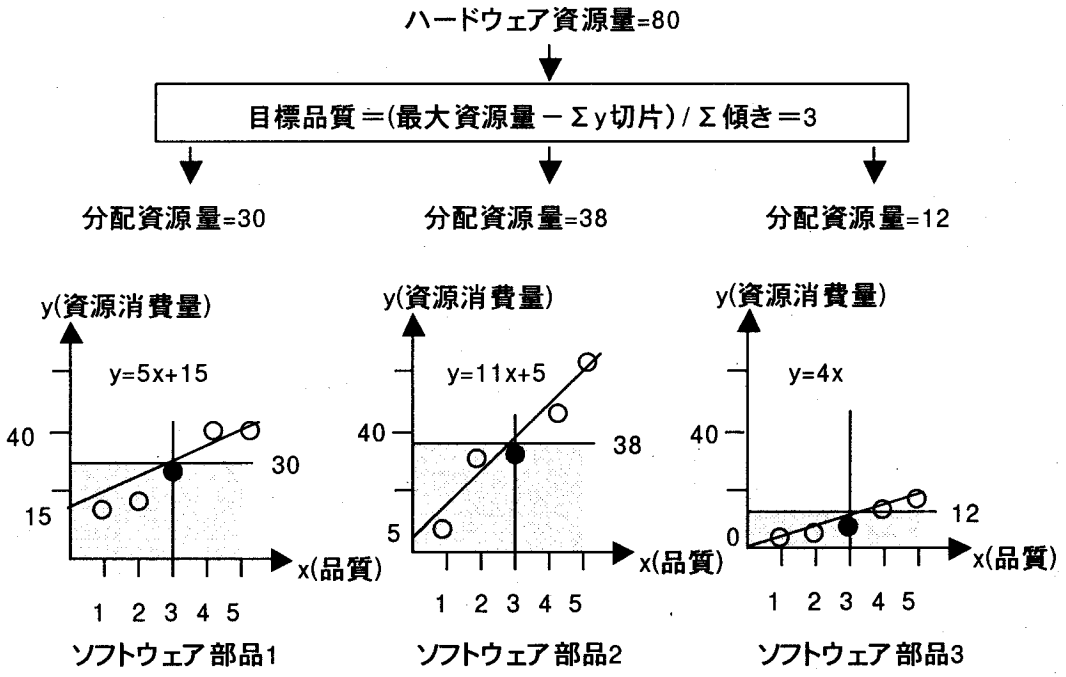

図 3.5 品質の決定方式

① k 番目のソフトウェア部品の品質 x と m 番目のハードウェア資源に対する消費量 y の関係を下記の直線で近似する.

$y = a(k, m)\ x + b(k, m)$

② アプリケーションの品質を，実行されるソフトウェア部品の品質の平均と定義すると，j 番目のアプリケーションの品質 X の，m 番目のハードウェア資源に対する消費量 Y も以下の直線で近似される.

$Y = A(j, m)\ X + B(j, m)$

$A(j, m) = \Sigma_{\text{j 番目のアプリケーション中の全ての k}}\ a(k, m)$

$B(j, m) = \Sigma_{\text{j 番目のアプリケーション中の全ての k}}\ b(k, m)$

③ 各アプリケーションの品質が等しくなるように，ハードウェア資源を分配する．この時の目標品質 Q(m) は次式で与えられる．

$Q(m) = (Hardware\_Profile(m) - \Sigma_j B(j,m)) / \Sigma_j A(j,m)$

従って j 番目のアプリケーションに配分される資源量 App_Allocation(j,m) は以下となる．

$App\_Allocation(j,m) = A(j,m)\ Q(m) + B(j,m)$

④ 各アプリケーションに配分された資源量を今度はソフトウェア部品に配分する．k 番目のソフトウェア部品に配分される資源量 Comp_Allocation(k,m) は以下で与えられる．

$Comp\_Allocation(k,l) = a(k,m)\ Q(m) + b(k,m)$

⑤ 分配された資源量を超えない範囲で，最も高い品質を各ソフトウェア部品に対して選択する．

⑥ 決定した品質に対して，資源量の総和 (Consumption) に余裕があれば，他に比べて品質の低いソフトウェア部品に対して，再配分を行う．

⑦ すべての m に対して上記①－⑥を行って，最終的な品質を決定する．

3.4 提案手法にもとづく端末の試作

3.4.1 試作の概要

本提案の設計手法を評価するため，図 3.2 に示したフローにしたがってマルチメディア端末を試作した．メディアプロセッサとしては，動作周波数 400MHz の HMPV を使用し，周辺 I/O としてデジタル放送のチューナー，イーサネット，映像と音声の入出力を付加している．記憶装置としては，256M バイトの SDRAM (Synchronous Dynamic Randam Access Memory) と 20G バイトの HDD (Hard Disk Drive) を使用した．試作端末のハードウェア構成を図 3.6 に示す．また試作した端末の内部と外観を図 3.7 と図 3.8 にそれぞれ示す．

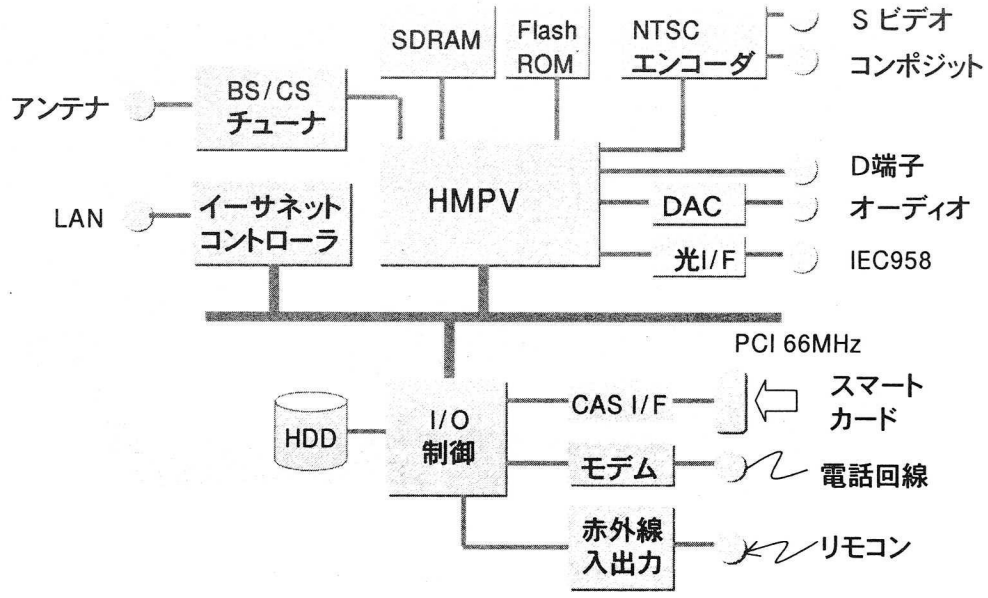

図 3.6 試作端末のハードウェア構成

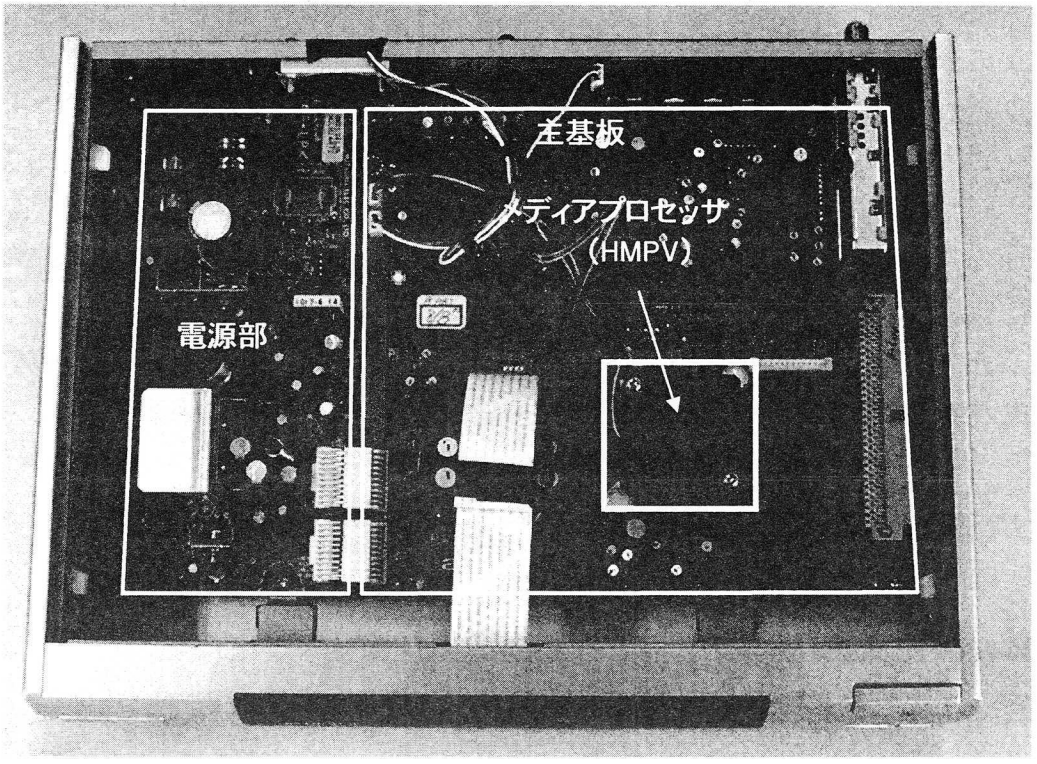

図 3.7 試作端末の内部

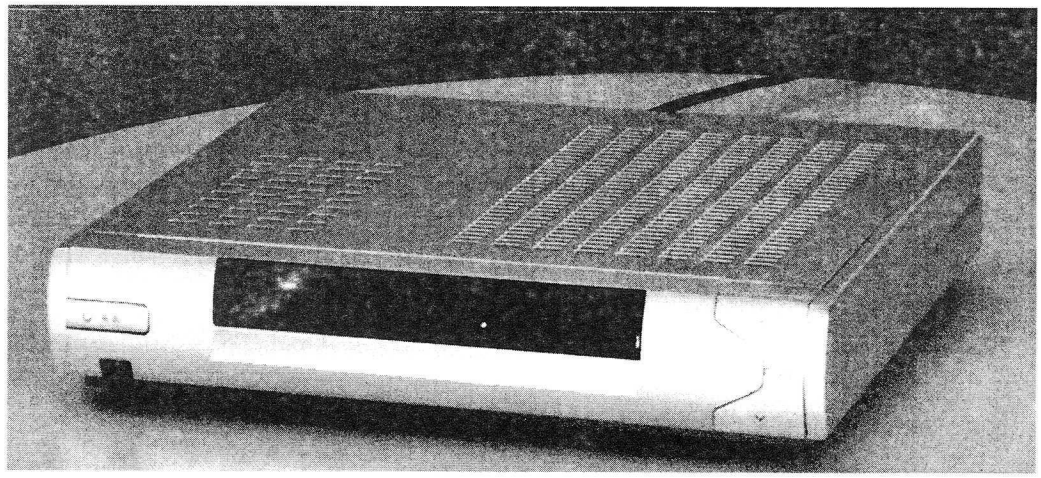

図 3.8 試作端末の外観

ソフトウェア部品として，MPEG2 や MPEG4 などの音声と映像のデコーダとエンコーダ，リアルタイム OS，ネットワークアクセスやハードディスク制御等の IO 制御を開発した．ソフトウェアの全体構成を図 3.9 に示す．

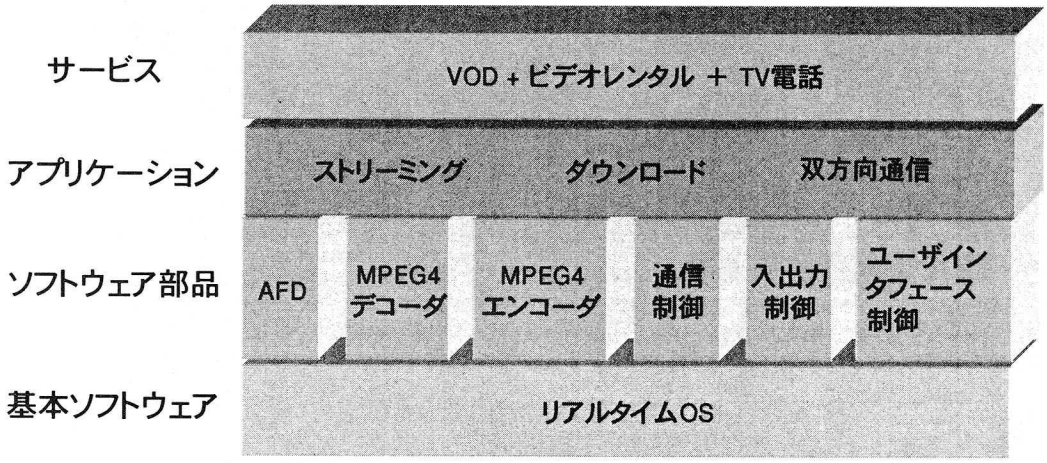

図 3.9 試作端末のソフトウェア構成

表 3.2 に資源プロファイルの Hardware_Profile データベースの内容例を示す

表 3.2　Hardware_Profile データベース

資源番号	資源名	最大資源量
1	VLIW CPU	400Mcycle
2	ﾋﾞｯﾄ列処理ﾌﾟﾛｾｯｻ	400Mcycle
3	外部ﾒﾓﾘ量	256MByte

端末の初期サービスは，FTTH 経由の MPEG2 の高精細ストリーミングによる VOD (Video On Demand) サービスとした．初期サービスの開発完了後に，まず MPEG4 のデコーダソフトウェアの追加による HDD へのダウンロード録画ファイルの再生機能を追加して，デジタルビデオのレンタルサービスを実現した．さらに MPEG4 のエンコーダソフトウェア追加による TV 電話サービスの追加を行い，提案設計手法のフィージビリティを調べた．VOD とビデオレンタルは同時には実行しないものとし，TV 電話機能はこれらのサービスと同時に使用可能とした．本シナリオにおける，資源プロファイル中の Service_Profile と Application_Profile をそれぞれ表 3.3 と表 3.4 に示す．

表 3.3　Service_Profile データベース

ｻｰﾋﾞｽ番号	ｻｰﾋﾞｽ名	ｽﾄﾘｰﾐﾝｸﾞ	ﾀﾞｳﾝﾛｰﾄﾞ	双方向通信
1	VOD	1	0	0
2	ﾋﾞﾃﾞｵﾚﾝﾀﾙ	0	1	0
3	TV 電話	0	0	1
4	ｽﾄﾘｰﾐﾝｸﾞ＋TV 電話	1	0	1
5	ﾋﾞﾃﾞｵﾚﾝﾀﾙ＋TV 電話	0	1	1

表 3.4　Application_Profile データベース

ｱﾌﾟﾘｹｰｼｮﾝ番号	ｱﾌﾟﾘｹｰｼｮﾝ名	AFD	MPEG4 ﾃﾞｺｰﾀﾞ	MPEG4 ｴﾝｺｰﾀﾞ
1	ｽﾄﾘｰﾐﾝｸﾞ	1	0	0
2	ﾀﾞｳﾝﾛｰﾄﾞ	0	1	0
3	双方向通信	0	1	1

3.4.2 資源プロファイル

初期機能である MPEG2 デコード処理では，対象とするビットストリームの解像度，フレームレート，ビットレート等は外部から一意に与えられるので，単純な実装では消費資源量を制御することは困難である．そこで本試作では，AFD (All Format Decoder) [48] として知られる方式に基づく，資源消費量が制御可能なビデオデコーダを実装した．AFD は圧縮ビットストリームを水平または垂直方向にダウンサンプリングして，既存の SDTV で HD 放送を楽しむことを目的に考案された方式である．本試作ではダウンサンプルレートを変えることで出力解像度とハードウェア資源消費量を制御できるように実装した。表 3.5 に AFD の Component_Profile の内容を示す（このプロファイルならびに後述の MPEG4 のプロファイルは OS や IO 制御等の周辺ソフトウェアが消費する資源量も含む）．本実装ではダウンサ

表 3.5　AFD の Component_Profile

部品番号	ｿﾌﾄｳｪｱ名	ﾀﾞｳﾝｻﾝﾌﾟﾙﾊﾟﾗﾒｰﾀ (水平×垂直)	品質	VLIWCPU 消費量 (cycle/sec)
1	AFD	1×1	5	305M
		1×1/2	4	250M
		1/2×1/2	3	210M

ンプルレートの制御により，デコードに必要なメモリ量の削減も可能となる．例えば，1080i フォーマットのビットストリームをデコードする場合，1フレームあたりのフレームバッファサイズは，通常デコーダでは約 3MB 必要となるが，水平方向および垂直方向にともに 2 分の 1 にダウンサンプリングした場合のフレームバッファ使用量は約 0.8MB である．一方，ビット列処理プロセッサの消費サイクル数は，入力解像度のみに依存するので，ダウンサンプルによって制御することはできない．本 MPEG2 デコーダのビット列処理プロセッサ消費量は約 240M サイクル/秒である．消費資源量はどのハードウェア資源に関しても同様に分析できるので，以下では単純化のため，クリティカルな資源の一つである，第 1 番目のハードウェア資源 VLIW CPU を例に議論を進める．

表 3.6 に MPEG4 デコーダの VLIW CPU 資源消費量と品質の評価結果を示す [50]．表 3.6 に示すように，ビデオレンタルサービスにおける MPEG4 デコーダは解像度の異なる複数のファイルフォーマットに対応するよう設計した．一方，TV 電話機能を付加するために用意した MPEG4 エンコーダでは，出力するフレームレートやビットレートを変えることで，消費するハードウェア資源量の制御が可能である。

表 3.6 MPEG4 デコーダの Component_Profile

部品番号	ソフトウェア名	ファイル形式（解像度）	品質	VLIWCPU 消費量 (cycle/sec)
2	MPEG4 ﾃﾞｺｰﾀﾞ	D1	5	125M
		Half D1	4	75M
		CIF	3	50M
		QCIF	2	35M

表 3.7 に本試作で使用した MPEG4 エンコーダ [81] の，解像度，フレームレート，ビットレート等のパラメータを変化させた時の消費 VLIW CPU サイクル数の変化を示す．

表 3.7　MPEG4 エンコーダの Component_Profile

部品番号	ソフトウェア名	出力解像度	フレームレート (fps)	ビットレート (kbps)	品質	VLIWCPU 消費量 (cycle/sec)
3	MPEG4 エンコーダ	D1	30	2048	5	345M
		Half D1	30	1024	4	210M
		CIF	30	384	3	90M
		QCIF	15	128	2	50M

3.5 評価

3.5.1 実時間応答性保証方式の評価

3.3.3 で説明した，品質制御による実時間応答性保証方式を適用した結果を表 3.8 に示す．

表 3.8 の第 2 列から第 5 列は，3.3 節に述べたステップ①－④から得られる，サービスで実行される各ソフトウェア部品に対する目標品質（各欄の上段）と資源配分（各欄の中段），および目標品質に最も近い品質を与えた時の資源消費量の予測値（各欄の下段）を示している．また第 6 列は最終的に得られたサービス品質の主観評価値（各欄の上段）と，資源消費量の実測値（各欄の中段）と資源プロファイルから得られる予測値（各欄の下段）を示す．ここでの実測値はメディアプロセッサの動作周波数を 400MHz から順次下げていき，実時間応答性が損なわれる直前の値とした．

以下 VOD と TV 電話の同時実行サービスを例にとって品質制御の過程を説明する．本サービスは，ストリーミングと双方向通信の二つのアプリケーションを含む．VOD は AFD，双方向通信は MPEG4 のデコーダとエンコーダをそれぞれソフトウェア部品としている．

表 3.8　ソフトウェア部品への資源配分

サービス名	目標品質と資源配分				最終品質と実資源消費量
	ストリーミング	ダウンロード	双方向通信		
	AFD	MPEG4 デコーダ	MPEG4 デコーダ	MPEG4 エンコーダ	
VOD	q=5.0 400 [305]	-	-	-	q=5 313 [305]
ビデオレンタル	0	q=5.0 400 [125]	-	-	q=5 128 [125]
TV 電話	-	-	q=4.4 108 [75]	q=4.4 291 [210]	q=4 297 [285]
VOD ＋TV 電話	q=2.8 203 [210]	-	q=2.8 60 [50]	q=2.8 134 [90]	q=3 366 [350]
ビデオレンタル ＋TV 電話	-	q=3.8 88 [75]	q=3.8 88 [75]	q=3.8 223 [210]	q=4 375 [360]

[　]内は Component_Profile からの予測値

ステップ①：

AFD と MPEG4 デコーダ，エンコーダの品質と消費資源量の関係を，表 3.5，表 3.6，表 3.7 をもとに直線で近似すると，y=47.5x+67.5，y=30x-25，y=98.3x-146.6 を得る．

ステップ②：

これより，ストリーミングと双方向通信の品質と消費資源量の関係は，y=47.5x+67.5，y=128.3x-171.6 となる．

ステップ③：

最大資源量は 400Mcycle であるので，目標品質 Q=(400-67.5+171.6)／(47.5+128.3)=2.86

が得られる．したがってストリーミングへの資源配分は，47.5×2.86+67.5=203Mcycle，双方向通信に対しては，128.3×2.86-171.6=195Mcycle となる．

ステップ④：

双方向通信の二つのソフトウェア部品である MPEG4 デコーダとエンコーダへの資源配分値も同様に，30×2.86-25=60Mcycle，98.3×2.86-146.6=134Mcycle となる．

ステップ⑤：

表 3.5 よりわかるように，AFD にはこれを満足する品質が存在しない．一方，表 3.6，表 3.7 に示すように，MPEG4 のデコーダ，エンコーダについては，それぞれ CIF 解像度（品質は 3）に対して，資源消費量が 50Mcycle と 90Mcycle で条件を満たすので，これを選択する．

ステップ⑥－⑦：

この時の資源の余裕は 260Mcycle であり，これを AFD に再配分すると，今度は 1/2×1/2 のダウンサンプリングでの品質 3 において，消費資源量 210Mcycle となり，条件を満たす．この段階での資源の余裕は 50Mcycle である．これを MPEG4 デコーダに再配分して解像度を Half D1（品質 4）に上げることができる．資源の残りは 25Mcycle となり，これ以上の品質向上はできない．すなわち，すべてのソフトウェア部品について，この品質を上回る解は存在しないという意味において，この解は最適解の一つとなっている．ここでは省略するが，実際には同様な品質制御を表 3.2 に示した他資源についても行う必要がある．

表 3.8 より明らかなように，各サービスの目標品質の±10%以内での実時間応答を確認した．また，TV 電話と他サービスの同時実行の場合など，複数ソフトウェア部品が並行動作する際には資源使用の待ちにより，資源消費量の予測値より実測値の方が大きくなる傾向がある．したがって資源量には誤差を吸収できる程度の余裕が必要である．本試作の場合，全サービスにおいて予測値と実測値の差は 5%以下であった．

3.5.2 設計開発期間の評価

図 3.10 に示すように，AFD や MPEG4 エンコーダ，デコーダ等の主要なソフトウェア部品自体の開発には，メディアプロセッサ上での実装にそれぞれ約 40 人月を要した．これはこれ

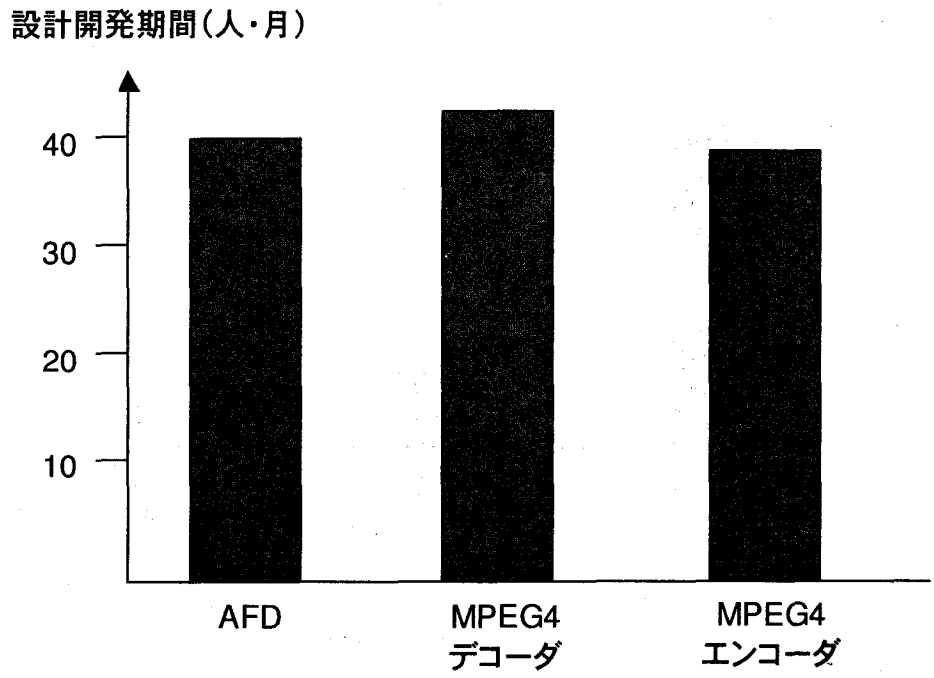

図 3.10 ソフトウェア部品の開発工数

らのソフトウェア部品に対する性能最適化に工数を要しているためである.

図 3.11 に初期設計開発と，機能追加変更に要した期間を示す。適切なソフトウェア部品が揃っている場合，変更に要する工数は 2-3 人月であった．新しい機能を持ったブロードバンド端末を新規に設計，開発，配布することに比べて，サービス提供者とユーザのメリットは大きく，メディアプロセッサを搭載した端末の実時間応答性を保証しつつデジタルメディア処理機能をソフトウェアで追加するという，提案手法の有効性を確認した.

3.6 結言

ブロードバンド端末設計の問題点である，新規格の出現などによる仕様確定における困難を解決する設計手法を提案した．端末機能をソフトウェア化して機能の追加変更が可能となるというメディアプロセッサのメリットを活かせるよう，出荷後の機能追加変更も考慮した設計フローを提案した．機能の追加変更によりソフトウェア実行時のハードウェア消費資源量が実資源量を越えて実時間性が損なわれる問題に対し，消費資源量を管理するた

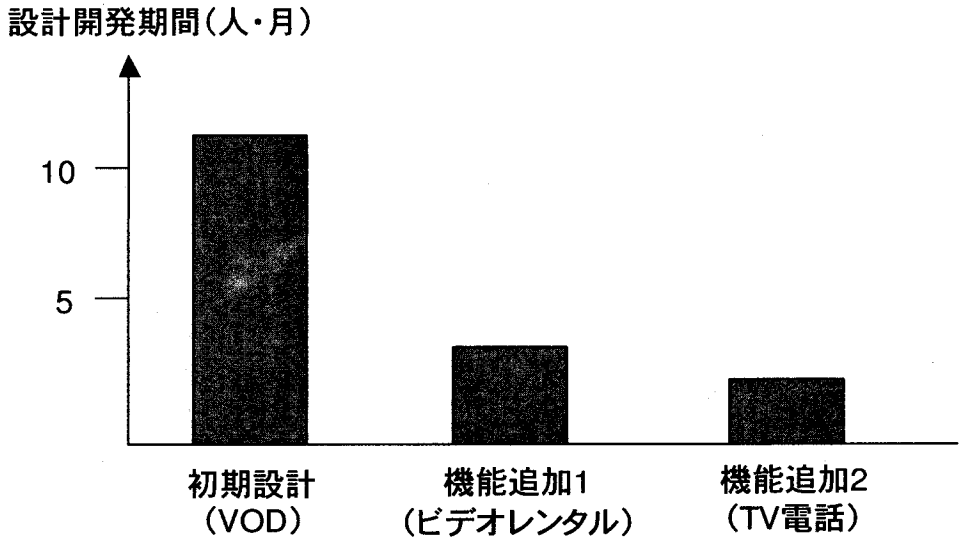

図 3.11　変更に要した期間

めのフレームワークと，画質等の品質パラメータをサービスの並行性に応じて動的に変更して実時間応答性を維持する方式を提案した．本設計手法に基づき，ブロードバンド端末を試作評価して以下の結果を得，その有効性と実用性を確認した．

(1) 与えられたハードウェア資源制約下で，目標品質の±10%内での実時間応答を確認
(2) 消費資源量の予測値と実測値の差は 5%以下
(3) 主要なソフトウェア部品の開発工数は約 40 人月
(4) ソフトウェア部品が揃っている場合，新サービスの追加工数は約 3 人月

現在，本設計手法の本格的な適用に向けて，ソフトウェア部品のレパートリー拡大と資源管理プロファイルの整備を図っている．今後の課題を以下に示す．

(1) プロファイリングの自動化

本研究における品質と実時間応答性の評価はまだ人手に頼っている部分が多く，その自動化による効率向上が重要な課題である．

(2) プログラミング環境の高度化

ソフトウェア生産性をさらに向上させるためには，GUI (Graphical User Interface) を有効に用いた，視覚的で統合化されたプログラミング環境を実現する必要がある [82] [83] [84]．特に，マルチメディア処理では，データ構造の可視化に基づくプログラミング支援が有望と考えられ，今後の研究課題である．

(3) セキュリティ対策

サービスプロバイダからのソフトウェアダウンロードが一般化すると，種々のセキュリティ上の問題が顕在化する可能性がある．例えば，サービスプロバイダのダウンロード用のサーバになりすまされて，アクセスした端末がウィルスに感染するといった脅威である [85]．本章で述べたサービス，アプリケーション，ならびにソフトウェア部品を認証するフレームワークを提供する必要がある [86]．コンピュータの高速化に伴い，核となるソフトウェア部品のセキュリティホール対策としては，プログラムの正当性を形式的に証明する方式 [87] [88] [89] も適用可能となりつつあり，今後の研究の進展が期待される．

第4章　マルチメディアデータベースサーバの高速化方式

4.1 緒言

本章では，文献[56] [57] [58] [59]に基づいて，定型的検索と非定型的検索の両方の高速化が可能なデータベース管理システムの高速化方式を提案し，システムの試作を通して有効性を評価する．

マルチメディアサービスシステムにおけるデータベース管理サーバの役割は，顧客情報やコンテンツの書誌情報など，サービスで必要となるすべての情報を管理することである．音声や映像等のストリームデータの送出はそれに特化したストリームサーバを用いることが多い[13]．図4.1に典型的なマルチメディアデータベースサーバのシステム構成を示す．

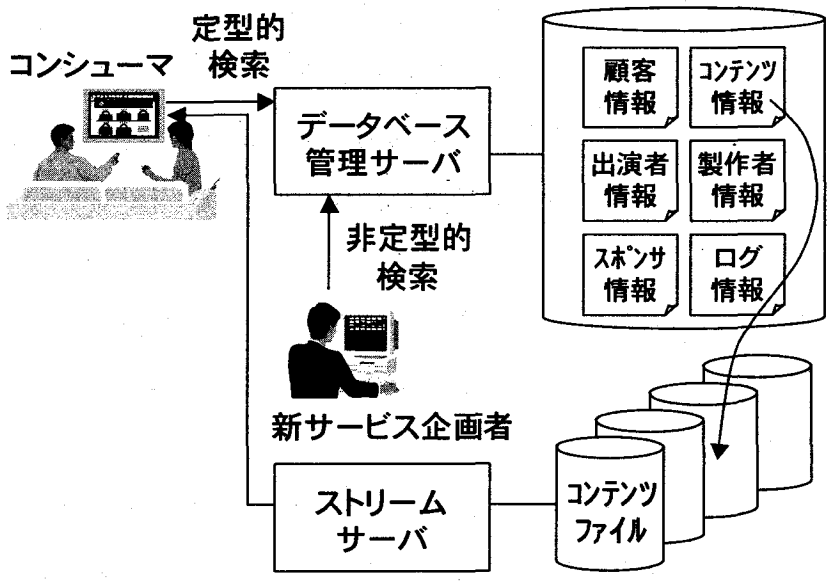

図4.1 マルチメディアデータベースサーバのシステム構成

顧客の嗜好に合った新サービスを迅速に提供するには，サービスに対する顧客の反応を常に蓄積，分析する必要がある．したがって，データベース管理サーバにおいては，見たいコンテンツの一覧を表示する，といったコンシューマからの定型的な検索要求を高速に処理するだけでは不十分である．顧客のコンテンツ利用状況の分析を目的とする，新サー

ビスの企画者からの大規模で非定型的な検索要求に効率良く対応できることが重要である.

データベース管理システムとしては，関係データベース管理システム[36]が機能的に十分であるため広く適用されている．関係データベース管理システムは平明な論理構造と強固な数学的基盤を背景に幅広い注目を集めてきた．1970年代の理論研究から80年代初頭の実験システム開発を経て，今日データベース管理システムの標準としての地位を確立している．関係データベースシステムは問い合わせパターンが決まっている定型的な検索に対しては，必要な部分に索引付けを予め行うなどの工夫により，十分な性能を実現できる．しかしながら，索引付けによる高速化はデータ量の増大と更新性能の低下を招くため，非定型的な処理に対しては有効でなく，性能向上が課題となっている．

この課題を専用アーキテクチャにより解決しようとする試みとして，データベースプロセッサの研究開発が行われてきた[90]．商用マシンの開発例もあるが，いくつかの基本的課題が残されていることも事実である．そのうちの一つにＩ／０隘路の問題がある．実際，CPUの演算速度と磁気ディスクのＩ／０速度のギャップは拡大する一方であり，問い合わせ処理の大半はディスクＩ／０が占めるといっても過言ではない.

この問題に対し，二つのアプローチが有望視される[91]．一つは小規模なディスク装置を多数台用意し，これらを並列アクセスしてＩ／０速度の向上をはかる方式である．この考え方にそったデータベースプロセッサとしては，GRACE[41]，DBC/1012[42]，GAMMA[92]などがあげられる.

一方，急速な半導体技術の進展により，主記憶装置のような大容量の半導体メモリ上にコンテンツ本体を除いたデータベースの大部分を常駐する方式も現実的なものになりつつある[93]．この主記憶データベースのアプローチに，さらにスーパーコンピュータとして効果をあげているベクトル処理の考え方を加えた新しい関係データベースシステムのアーキテクチャが提案されている[94] [95]．本方式は検索対象をベクトル形式で主記憶上に動的に展開し，これをパイプライン処理して高速な関係演算の実現を目指すものである.

しかしながら従来の数値計算用のベクトル処理装置では，探索演算やソート演算などの関係データベースにおける重要な基本演算のベクトル化が困難である．本章ではこの問題に対処するため，データベース処理向けのプログラマブルな集合演算ユニットを提案する．以下4.2節では従来の関係データベースサーバの問題点について述べ，4.3節で集合演算ユニットによる非定型的検索の処理方式を提案し，4.4節では集合演算ユニットのハードウェア実装について述べ，4.5節では性能評価結果について論じる.

4.2 従来方式の問題点

ソフトウェアで実現された従来の関係データベース管理サーバの多くはテーブルの横一行（以下レコードと呼ぶ）を処理の単位（粒度）としている．ディスク上の物理的データ構造についてみると，System R にみられるようにページ内にレコードを単位として格納する形式をとっている[12]．また内部の処理もすべて1時点1レコード(one record at a time)を原則とする．つまり各プログラムモジュールは単一のレコードのみを操作するよう設計されており，システムは1レコードずつ逐次的に処理を進めている．

レコード単位処理方式の利点として以下があげられる．

(1) 数キロバイト程度の主記憶さえあれば任意の大きさのデータベースが処理できる．

(2) 少量のレコードを検索する場合，適当なインデクスを用意することによってディスクI／O回数を削減できる．

(3)　レコードをそれへのポインタで管理する形式をとれば，レコードを連続配置する必要がなくなるため，レコードの追加や更新あるいは削除が容易となる．

(4) レコードやページをロック単位とした高並列度のトランザクション制御が可能となる．

逆に欠点としては以下があげられる．

(1) 問い合わせ処理で大量のレコードにアクセスする場合には，不連続なディスクI／Oが多発し平均ディスクアクセス速度が低下する．またレコード型のチェックやモジュール間遷移処理など，レコード数に比例するオーバヘッドのため，CPU処理時間も増大する．

(2) 論理的には単純なはずのテーブル構造がポインタを多用したチェイン構造で実現されることになり，処理の複雑化を招く．

(3) 専用プロセッサを用いたパイプライン処理あるいは並列処理による高速化の適用が困難である．

すなわちレコード単位処理方式は各トランザクションが比較的少数のレコードを処理する場合に適した方式であるといえる．

一方，データベース処理専用プロセッサ（以下データベースプロセッサと呼ぶ）に目を転じると，その多くはテーブル結合のような大量レコード処理の高速化を目指して設計されており，より大きな粒度の処理単位を採用している．ストリーム[96]はその好例である．実際，いくつかの実験システムではテーブル結合の性能においてレコード単位処理方式の商用システムを大きく上回る結果を報告している．しかしながら，例えば単一レコード検

索のような負荷の軽い問い合わせの処理ではむしろ不利である.

現実のデータベース環境では，通常は単一レコード検索が行われ，時に大量レコード処理要求が発生するといった形態が多くみられる．したがって 1 レコードから数十メガレコードに至る検索要求をともに効率良く処理する必要がある．いいかえると，処理単位をどう設定してこの相反する要求に応えるかがデータベースプロセッサの実用化に向けた大きな課題である.

4.3 集合演算ユニットによる非定型的検索の高速化方式

4.3.1 関係データベースにおける検索処理のベクトル化

図 4.2 に動的ベクトル化方式に基づく関係データベースシステムの構成を示す.

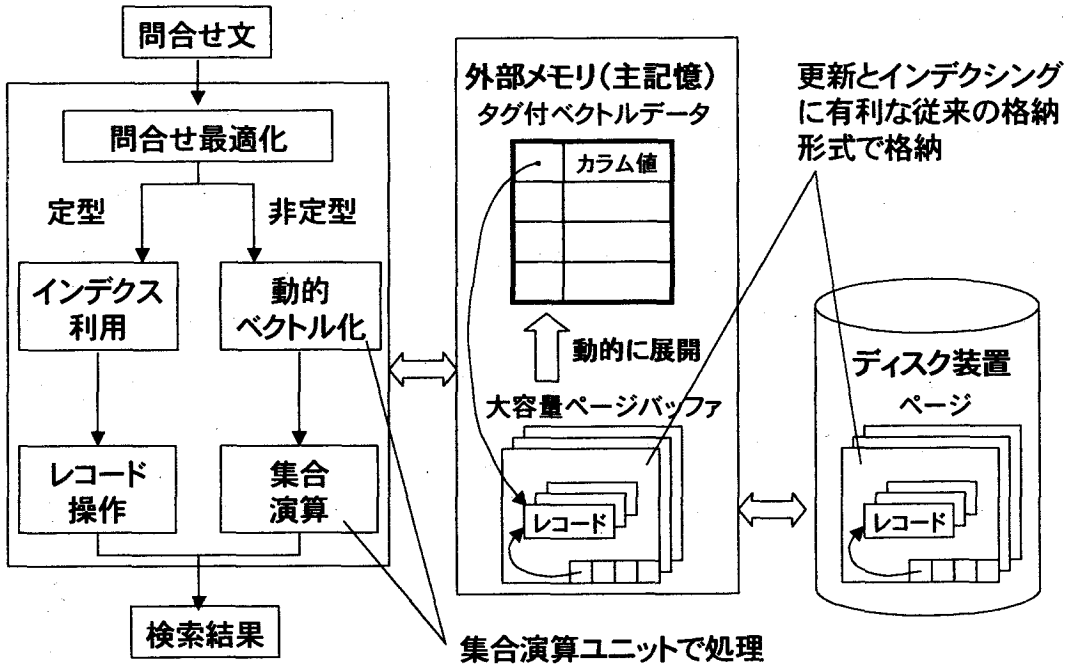

図 4.2 ベクトル型関係データベースシステムの構成

本方式の特徴は次のようにまとめられる.

(1) ディスク上のデータ格納形式は従来のレコード学位処理方式と同様とする.

(2) 主記憶上のデータは2通りの形式とする．一つはページバッファ，すなわちディスク上のデータページ群の写しである．もう一つはテーブルの縦一欄（以下カラムと呼ぶ）を後述のタグ付きのベクトル形式（デュアルベクトル）で配列したものである．図4.2に示されるようにデュアルベクトルの各要素は二つの部分から成っており，この例では前半部にタグとしてレコードへのポインタが，後半部にはカラムの値がそれぞれ格納されている．

(3) システムは検索要求を受け取ると，まず処理単位としてレコードとカラムのどちらが有利かを判定する．レコードが選ばれた場合，検索要求は従来のレコード単位処理方式で処理される．カラムが選ばれた時には，必要なカラムのデュアルベクトルが動的に生成されて集合演算ユニットによりパイプライン処理される．

以下では具体的な検索要求の処理手順を例として動的ベクトル化方式について説明する．検索要求としてはSQL言語で記述された次のようなテーブル結合を考える[36]．

```
SELECT
  PARTNAME, CODEA,
  MAKER, PRICE, TEL
    FROM A, B
      WHERE
    A. CODEA=B. CODEB
```

この問い合わせを例としてベクトル化による処理手順を図4.3に示す．

(1) デュアルベクトル生成

まず，従来のデータ構造を以下の手順で動的にベクトル形式に展開する．

(a) A, B の各テーブルについて，その中のレコード識別子（RID）を後半部とするデュアルベクトルを作成する．RIDはそのレコードが格納されているページの番号とページ内の何番目のレコードであるかを示すスロット番号とのペアである．デュアルベクトルの前半部はこの時点では空であり，後で代入される．

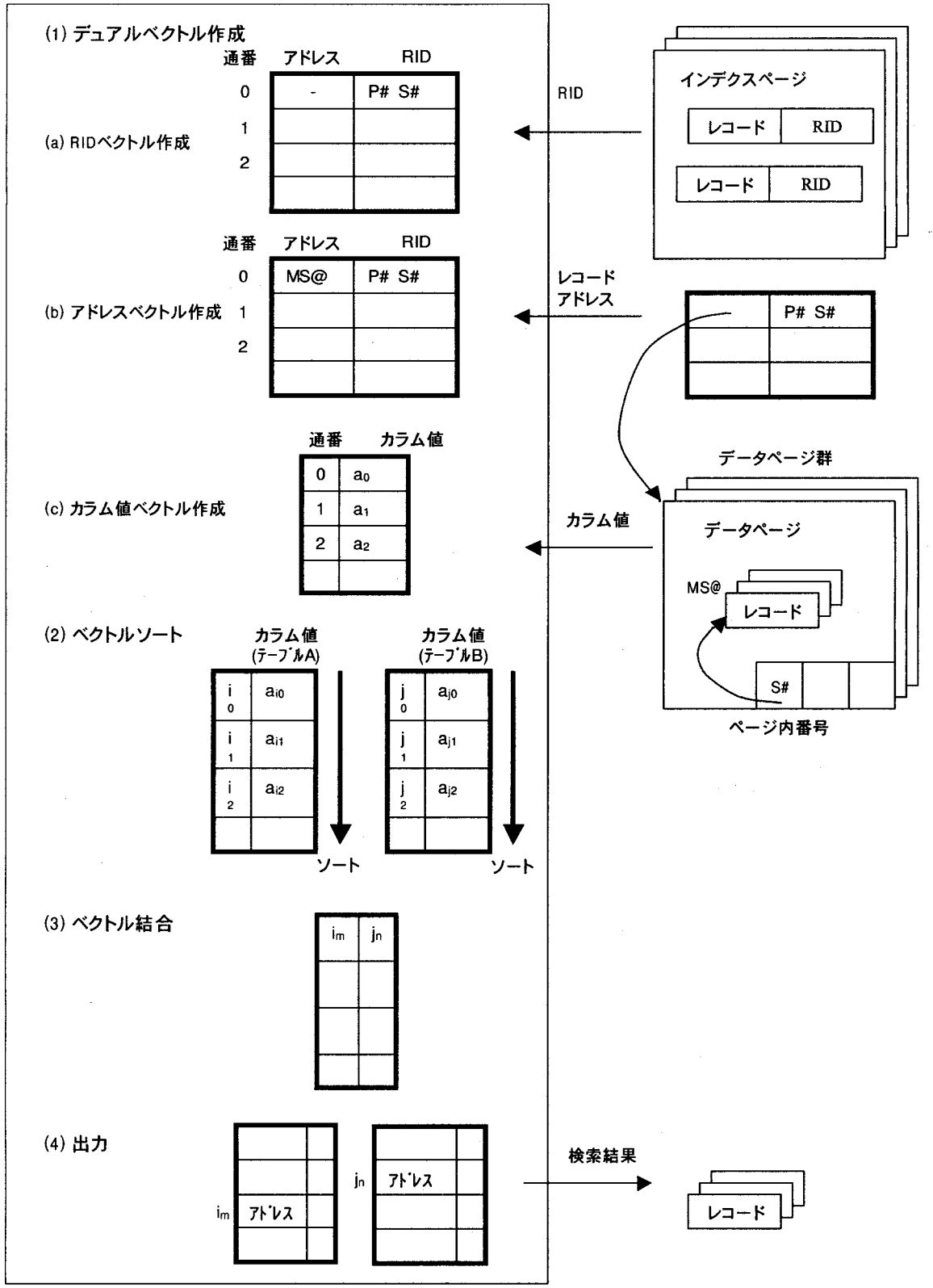

図 4.3 結合処理のベクトル化

(b) 各レコードに対して，その主記憶アドレスをデュアルベクトルの前半部に格納する．

(c) 得られた主記憶アドレスを用いてテーブル A，B 中の各レコードからその結合カラムの値を取り出し，別のデュアルベクトルの後半部に格納する．前半部には，通番を格納する．

(2) ベクトルソート

作成したデュアルベクトルを，その後半部についてソートする．

(3) ベクトル結合

ソートされたベクトルをマージしながら結合する．結果として後半部（カラム値）が等しいデュアルベクトル要素の前半部〈通番〉の対が得られる．

(4) 出力

最終的な結合結果を，(3) で得られた通番の対を使って作成し応答する．結合すべきレコードの主記憶アドレスは，通番をインデクスとしてアドレスベクトルを参照することにより知ることができる．

動的ベクトル化方式の長所として以下があげられる．

(1) 問い合わせの性質に応じて，最適な処理単位を選択できる．

(2) 従来の商用関係データベースシステムからみると，ベクトル化されたデータベースアクセスは新たな高速アクセスパスとなる．したがって商用システムがもつ豊富な機能はそのままにして高速化がはかれる．

(3) 全データベースを最初からベクトル形式で主記憶上に常駐する静的ベクトル化方式と比べ，主記憶の利用効率が高い．

(4) 二次記憶中を含めテーブルをすべてカラム単位に格納する方式では，各カラムの値を結果のレコードへと組み上げる処理の負荷が重い．動的ベクトル化方式ではレコードとカラム値ベクトルの両方をもっているのでこの処理が不要となる．

動的ベクトル化方式の性能向上には，ベクトル生成処理の高速化が重要である．これについては 4.5 節で定量的に評価する．

4.3.2 集合演算ユニットのアーキテクチャ

大規模数値計算専用と考えられてきたベクトル計算機は，記号処理の高速化にも高い潜

在能力を有していることが認識されつつある．例えばPrologなどの論理型言語で記述された記号処理プログラムに対しても，ベクトル計算機による高速実行の可能性が示されている[97] [98] [99] [100]．

データベース処理においては，従来のベクトル計算機がもつ機能（算術・論理演算，比較演算，マスク付き演算，圧縮・伸張演算，および間接アドレッシングなど）に加え，集合演算や O（n log n）であるような効率の良いソートアルゴリズムを用いたソート演算が必要となる[101]．

アーキテクチャ的にみると，集合演算ユニットは従来のベクトル演算に対して以下の点を拡張するものである．

(1) 従来のベクトル演算では，演算されるベクトル要素対は演算の開始前に静的に指定される．例えばベクトル A と B の加算（for i＝0 to N−1 do C(i)：＝A(i)＋B(i)）を例にとると，演算される要素はその要素番号が等しいもの同士に限られている．一方，集合演算の基本となるマージ処理では要素A(i)とB(j)の比較結果に応じて次にA(i＋1)とB(j)，あるいはA(i)とB(j＋1)との比較が行われる．すなわち，演算対象が演算の進行に応じて動的に変化する．そこで演算要素を示すベクトル要素カウンタを各ベクトルオペランド対応に設け，独立に更新するよう拡張する．

(2) 従来のベクトル計算機では，ベクトル要素のデータ型は，整数，浮動小数点数，論理データ，インデクス等のアトミックなものに限っている．集合演算ユニットでは，タグ付きデータ等，データベース処理で必要となるデータ構造を柔軟に表現するため，基本データ型としてデュアルベクトル形式を新設する．4.2節でも述べたように，デュアルベクトルの各要素をフロント部とリア部の二つの部分から構成する．フロント部とリア部にはそれぞれ異なった型のデータを格納できる．前節の図 4.3 は，フロント部に要素番号，リア部にはキー値を格納してソートする等の，デュアルベクトルの典型的使用例のいくつかを示している．

デュアルベクトルは任意の 2 項関係を表現できるため，多項関係である表データを分解してベクトル化するのに有効なデータ構造である．またデュアルベクトル構造は Lisp の cons セルをベクトル形式に配列したものともみることができ，より一般の記号処理のベク

トル化においても有効と考えられる．例えば記号処理用 SIMD 型計算機であるコネクションマシン[102]も，Xector（ゼクタ）と呼ばれるデュアルベクトルと類似性の高い基本データ構造を採用している．

図 4.4 に集合演算ユニットのデュアルベクトル命令の一覧を示す．機能的には，ソート，結合，逐次探索，および集合演算を行う命令群を用意している．各命令と関係演算との対応は以下のとおりである．

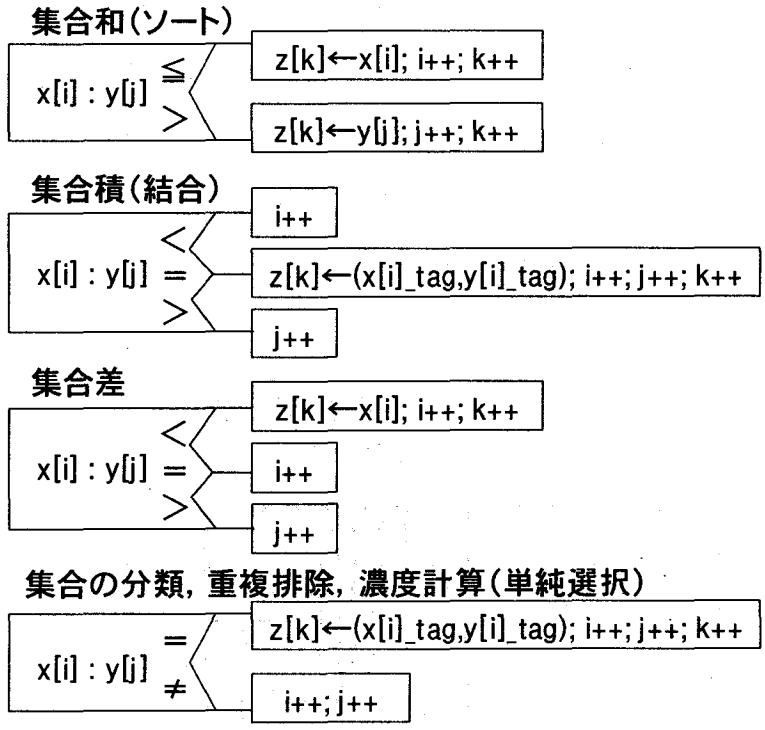

図 4.4 集合演算ユニットの命令体系

(1) 選択，制限，ソート，結合

これらの関係操作については直接に対応する命令を用意している．すなわち逐次探索命令が選択ならびに制限に，マージソート命令がソートに，また結合命令が結合にそれぞれ対応している．

(2) 集合和，集合積，集合差

これらの基本集合演算にはそれぞれ，マージソート命令，結合命令，集合差命令が直接に対応する．

(2) 集合の重複排除

逐次探索命令を用いて，射影演算などにおける中心的な処理である重複排除を行うことができる．例えば図 4.5 に示すように，ソート済みのカラム値ベクトル X の重複排除を行うには，X と X の開始アドレスを 1 要素分ずらしたベクトル Y とを対象に，異なるリア部をもつ要素対を逐次探索命令により探索すれば良い．この逐次探索命令では，リア部が異なるキー値をもつデュアルベクトル要素のフロント部の対のみがベクトル Z に出力される．したがって Z のフロント部にはXの中で等しいキー値をもつ要素群（r 1, r2), (r3, r4, r5), (r6) の先頭要素 r1, r3, r6 が得られ，逆にリア部には最後の要素である r2, r5 が得られる．

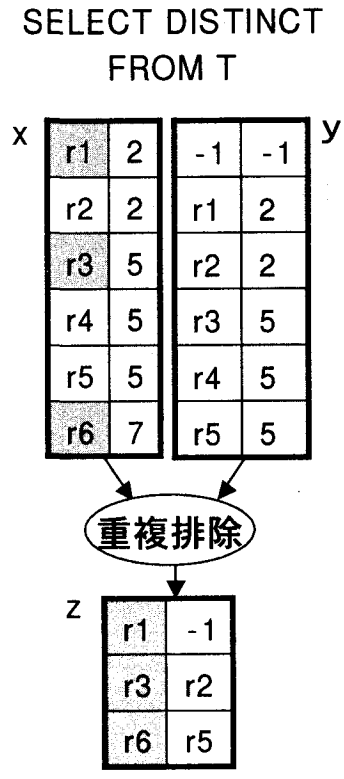

図 4.5 集合演算ユニットによる重複排除

(3) 集合の分類と濃度計算

SQL における GROUP BY 指定のような統計的な処理も，重複排除と同様の手法で図 4.6 に示すようにベクトル化できる．ここでは逐次探索命令に加えて二つの従来型のベクトル命令を使用している．一つは数列生成用の命令（Vector Element Increment）である．この命令の機能は（for i=0 to N−1 do X(i+1)=X（i）+A（i））であり，X（1）を 0，A（i）を 1 とすればXに 0 から始まる整数列が作業用デュアルベクトル V1 のフロント部に得られる．もう一つはベクトル要素間の差を求めるベクトル減算命令（Vector Elementwise Subtract）である．減算命令により結果ベクトル Z のリア部に重複要素の個数が得られる．

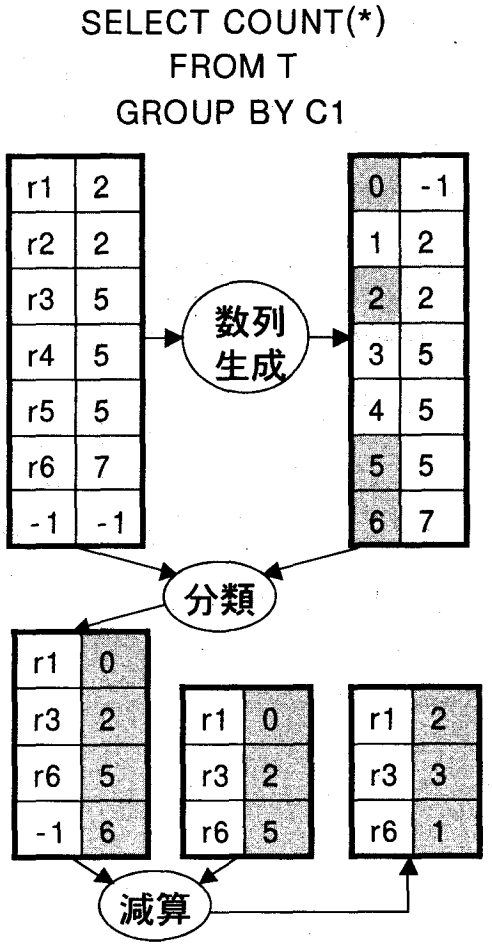

図 4.6 分類と集合濃度計算のベクトル化

4.2節で説明したように，上述のデュアルベクトル操作の前処理として，カラム値ベクトルの生成が行われる．したがって問い合わせ処理全体を効率良くベクトル化するには，関係の動的ベクトル化処理そのもののベクトル化が必要となる．新設したデュアルベクトル命令はこの目的にもかなうものである．図 4.7 に集合演算ユニットによるアドレスベクトル作成処理の概要を示す．この例はデータページ番号のリストVrを受け取ってその主記憶上

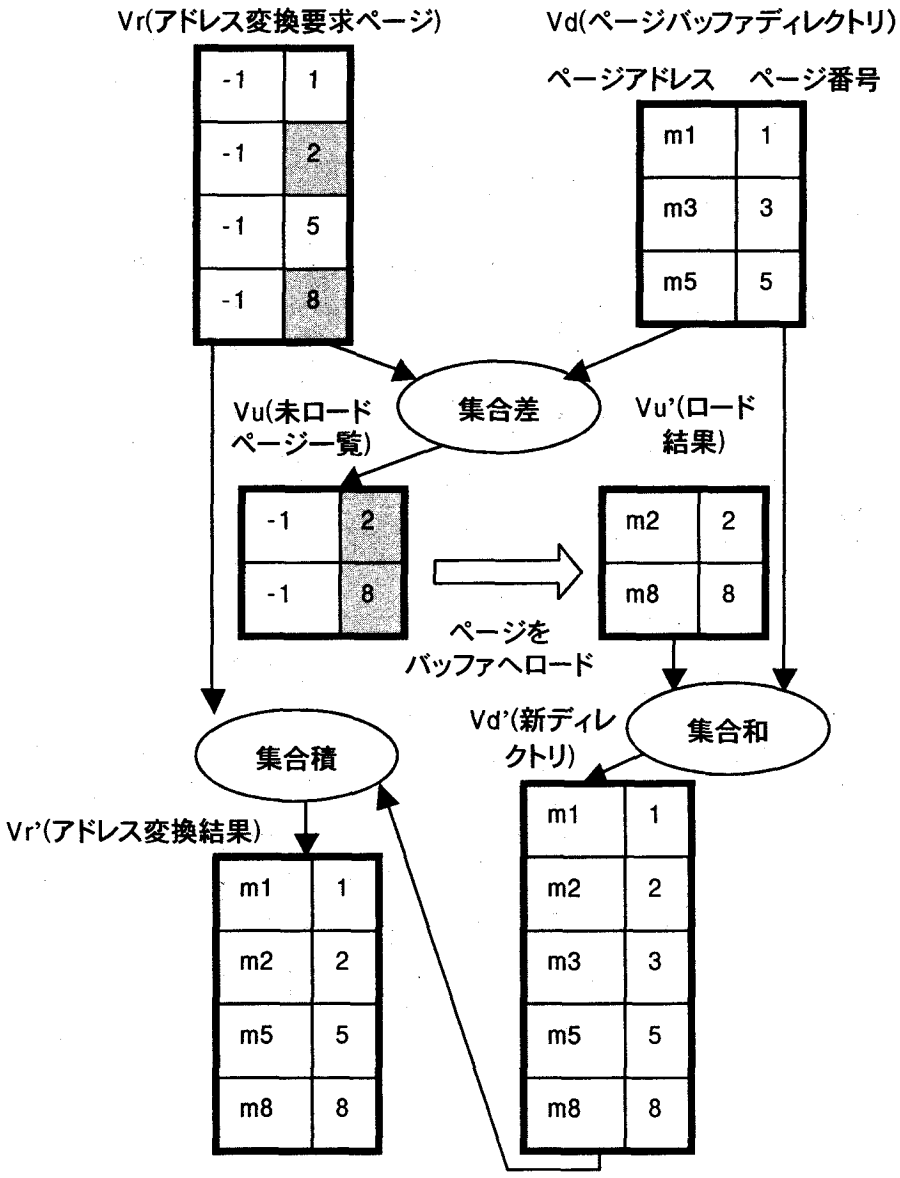

図 4.7 アドレスベクトルの作成処理

アドレスの一覧Ｖｒ′を得る，一種のアドレス変換処理である．主記憶にあるページバッファ上にロードされているデータページのアドレスはページ番号順にソートして，デュアルベクトル形式のページバッファディレクトリ Vd に格納しておく．変換は以下の手順で進められる．

(1) 集合演算ユニットの集合差命令により，Ｖｒと Vd の集合差 Vu を作成する．Vu は変換を要求されたページのうち，まだページバッファにロードされていないページの番号のリストである．

(2) スカラ命令を使って，未ロードページをロードし，その主記憶アドレス Vu のフロント部に格納して Vu′ を作成する．

(3) Vd と Vu′ をソート命令によりマージしてページバッファディレクトリを Vd′ に更新する．ここではソート命各は集合和を計算するのに使っている．

(4) Vr と Vd′ の集合積を結合命令を用いて計算し，変換結果 Vr′ を得る．

4.4 ハードウェア実装

データベースプロセッサはホストマシンとの結合形態から，スタンドアロン型，バックエンド型，および統合型の 3 種に大別される．スタンドアロン型は通信制御等を含めてすべての処理を独立して行うマシンである．スタンドアロン型においては，設計の自由度が大きいので，高速化のために最適な構成を追求することができる．反面，設計コストは高く，すべてのデータベース機能を実現するのは決して容易ではない．バックエンド型のマシンは，チャネルやネットワークを介してホストマシンと通信しながらデータベース処理を行う形態をとる．バックエンド型のデータベースプロセッサの主要な問題点はホストとの通信オーバヘッドにある．通信オーバヘッドを削減するには，ホストとのインタフェースの水準を上げてコマンドやデータのやり取りを減らす必要がある．しかしながらインタフェースを高水準化するほど，バックエンドマシンに必要な機能は多くなる．すなわちスタンドアロン型に近くなり，その性能価格比を損なう要因となる．

ここでは，通信オーバヘッドの少なさと低コストであることを重視し、ホストマシンの CPU の内部に付加して性能を向上させるアクセラレータとして集合演算ユニットを実装する．具体的には(株)日立製作所の大型汎用コンピュータである M-680H [103] の CPU に内蔵型の集合演算ユニット IDP (Integrated Database Processor) を付加した．図 4.8 に M-680H/IDP

の外観を示す．M-680H には，IAP（Integrated Array Processor）[104]と呼ばれる数値計算の四則演算用のベクトル演算ユニットも付加機構として用意され，IDP とともにデータベース処理用ベクトル演算ユニットを構成している．関係データベース処理のベクトル化にあたって，従来型のベクトル命令が必要な際，例えば前節図 4.6 中の四則演算は IAP を使用できる．集合演算ユニット IDP の論理規模は約 140kgate で，M-680H CPU 本体の約 5%程度である．

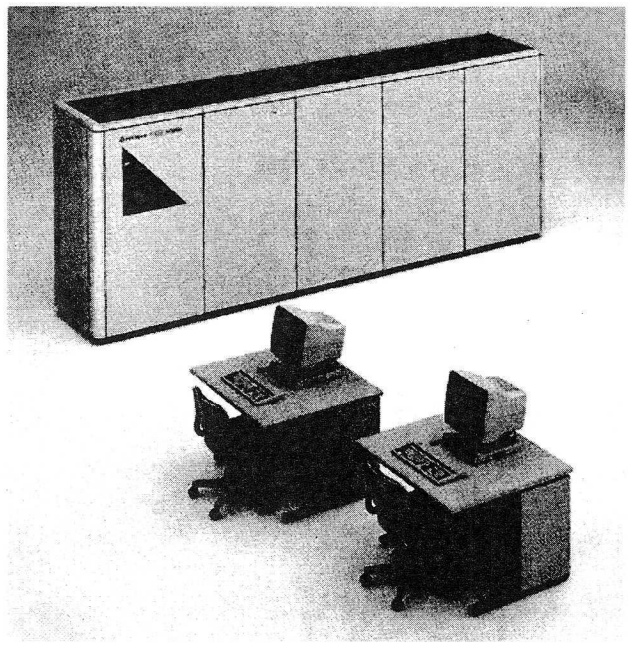

図 4.8 M680H/IDP の外観

図 4.9 に集合演算ユニットを付加した M-680H CPU の構成を示す．M-680H の記憶階層は，一次キャッシュ，二次キャッシュ，および外部メモリである主記憶装置の 3 階層から成っている．集合演算ユニットによるデュアルベクトル要素アクセスは高速な一次キャッシュに対して行われる．なお， IDP は M-680H とその次機種 M-880H [105] において専用ハードウェアとして実装され，それ以後の機種では，並列化したスカラ演算ユニットをファームウェアで並列に動作させることによってソフトウェア的に集合演算を処理している．

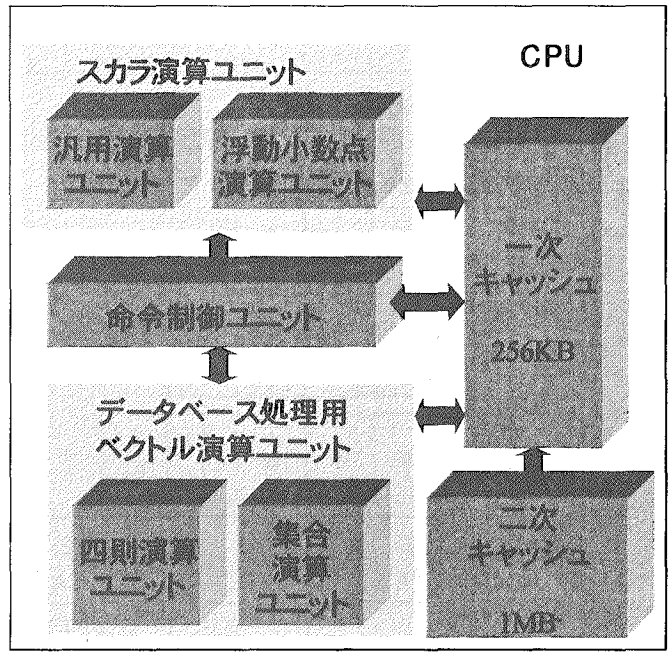

図 4.9 M-680H/IDP の CPU 構成

図 4.10 は集合演算ユニットのパイプラインによるベクトル処理の流れを示している．単一のベクトル要素に注目するとその処理は以下の順で行われる．

(1) ベクトル要素のフェッチ要求を発行するとともにフェッチアドレスを更新する．

(2) 集合演算ユニットへのベクトル要素到着を待つ．

(3) ベクトル要素を比較し，結果に応じてベクトル要素カウンタを更新する．

(4) 結果のストア要求を発行するとともに，ストアアドレスを更新する．

各要素の処理は 1 マシンサイクルごとに開始されるので，キャッシュのミスヒットなどのパイプライン擾乱がない限り，集合演算ユニットは 1 サイクルピッチで演算結果を生成する．集合演算ユニット命令機能を使わずに通常のスカラ命令で実現した場合，その最内側ループは約 20 命令から成る．したがって，ピークで 20 倍以上の加速率が得られる．

集合演算ユニットのベクトル要素に対するアクセス制御は従来のベクトル命令に比べてより復雑である．その理由は既に述べたように，集合演算では，直前のベクトル要素に対する比較結果に応じて次にアクセスされるベクトル要素が決まるからである．したがって，効率の良いパイプライン化を実現するには，あるベクトル要素に対する演算結果が確定す

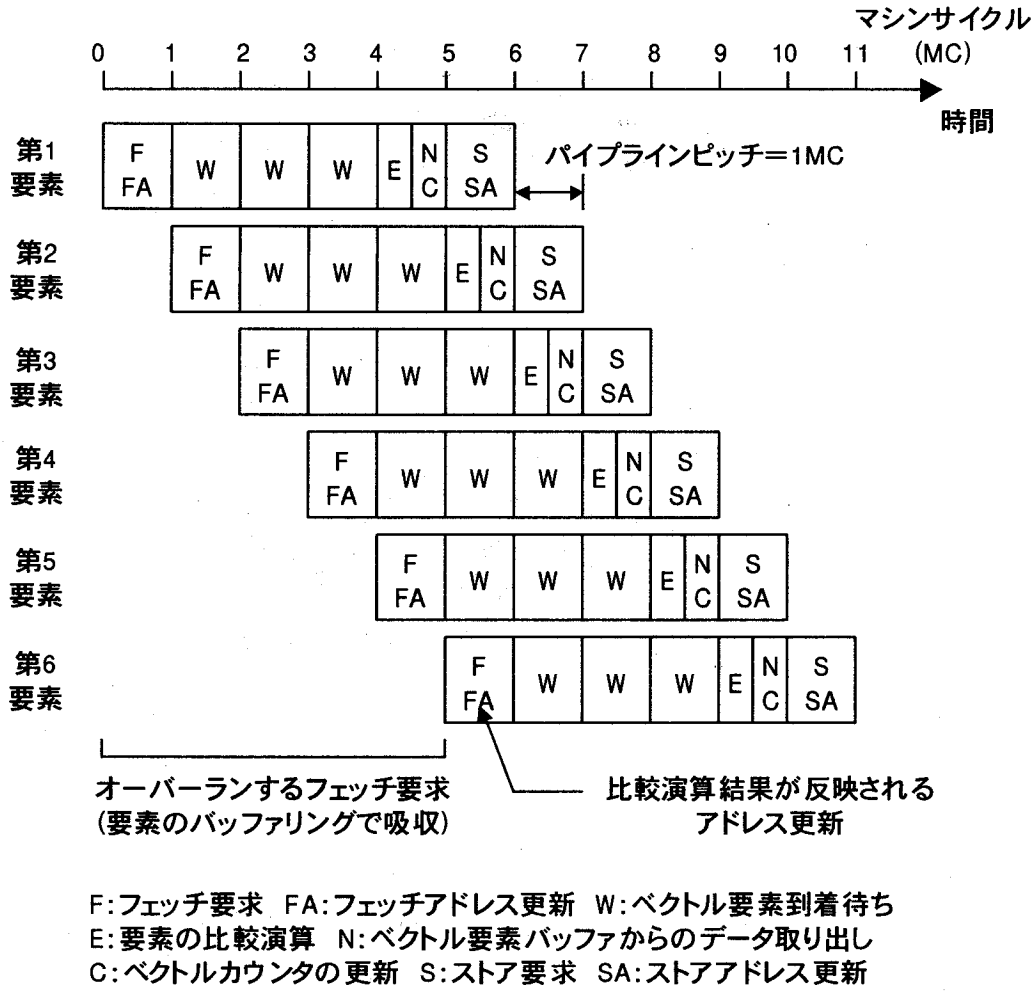

図 4.10　集合演算ユニットのパイプライン制御

る以前に，次のベクトル要素のアクセスを開始する必要がある．図 4.10 が示すように 1 サイクルピッチのパイプライン処理では，本来第 2 要素のアクセスに反映されるべき第 1 要素の演算結果を第 6 要素へと遅れて反映される．集合演算ユニットではこの制御遅れにより先行して集合演算ユニットに到着するベクトル要素を専用のメモリにバッファリングすることで，1 サイクルピッチのパイプライン化を実現している．

4.5 性能評価

4.5.1 命令レベル評価

本節では集合演算ユニットの命令レベルの性能実測結果について述べる．命令としてはソート命令を例にとり，集合演算ユニットを使用していない，種々のアルゴリズムを採用した通常のソートプログラムとの性能比較を行った．

図 4. 11 はその結果を示すグラフである．スカラ命令によるソートのアルゴリズムとしては，単純マージソート，自然マージソート，およびクイックソート [106] を用いた．図 4. 11 のグラフの横軸にはソート対象データの個数 n をとり，縦軸には各ソートアルゴリズムのオーダである n log n の係数をとっている．ソート対象データは，乱数的に発生させた 4 バイトのキー値を含む 8 バイトデータの列である．図 4. 11 から明らかなように，データ数が十分に多い場合，集合演算ユニットによるソートはソフトウェアによるソートプログラムに比べて約 10 倍高速である．集合演算ユニットの性能曲線はデータ数 10^4 を超えたあたりで二つに分かれている．上の線はデータ量がホストマシンのバッファ記憶あるいはワーク記憶に入りきらなくなって，ヒット率が低下しパイプラインが乱れることによる性能低下を示している．下の線は入力データをバッファ記憶やワーク記憶に入りきるように分割

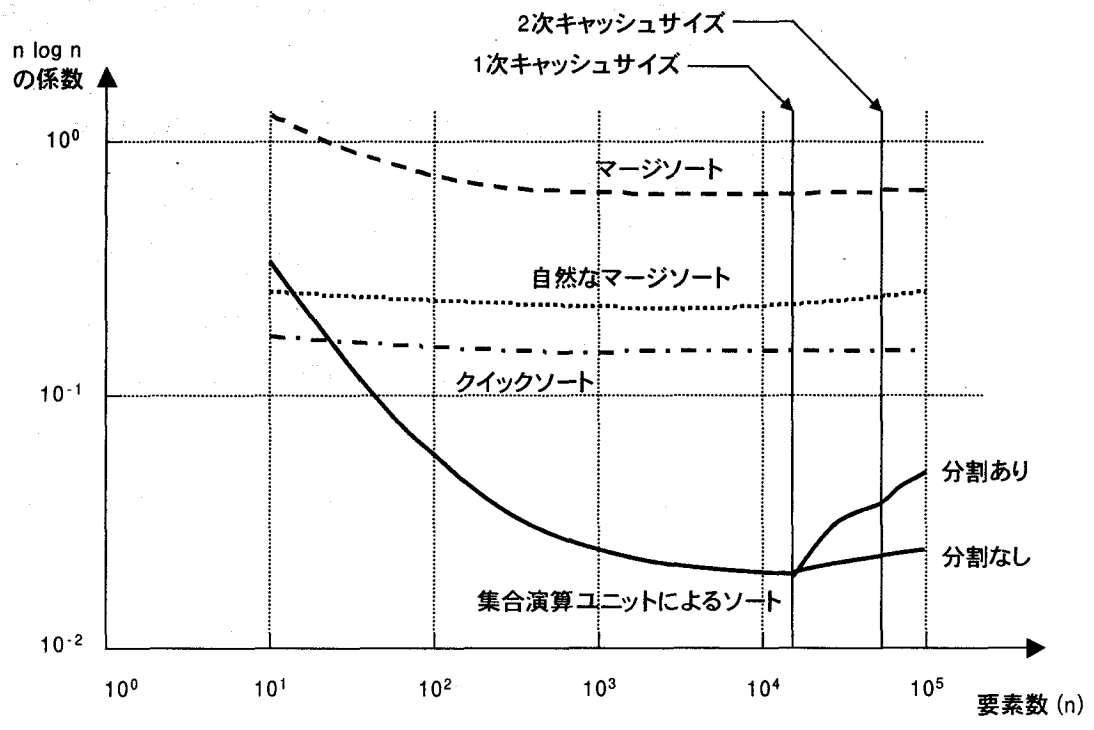

図 4. 11 集合演算ユニットの命令レベル性能

して処理した場合の性能を示している．この結果から大規模なテーブルをベクトル処理する際には，キャッシュに入る程度の大きさにテーブルを分割して処理するのが有効であるといえる．

4.5.2 コマンドレベル性能評価

SQL のコマンドレベルでの集合演算ユニットの性能を評価するにあたって，性能測定用のデータベースとして二つのテーブル T1，T2 を用意した．各テーブルは 6，000 レコードを格納し，各レコードの長さは 200 バイトである．次の 10 個の検索コマンドから成る連続するデータベースアクセスをベンチマークとして使用した．

(1) 非定型的な結合処理 2 件（インデクスのない環境での T1 と T2 の結合．結合結果は 100 レコードである．）
(2) 非定型的な全件検索 1 件（インデクスのない環境で，T1 の全レコードを走査して 100 レコードを出力する．）
(3) 定型的インデクス検索 7 件（インデクスを用いて T1 から 100 件を選択する．）

一般にベクトル処理による高速化においては，図 4.12 に示すような関係が成立する．図 4.12 の上段の帯グラフは，汎用計算機のスカラ命令による実行時間を，また下段はベクトル計算機のベクトル命令による実行時間をそれぞれ表している．スカラ命令による実行時間はさらにベクトル化不可能な部分 Tu と，ベクトル化可能な部分 Tv とに分けられる．ベクトル化可能な部分 Tv はベクトル計算機のパイプライン処理ハードウェアにより高速化される．パイプライン処理ハードウェアの加速率を r とすれば，ベクトル計算機による実行時間は Tu と Tv/r の和となる．全体としての性能向上率は次式で表される．

性能向上率＝（Tu＋Tv）／（Tu＋Tv／r）

したがって，ベクトル化により高い性能向上率を得るためには，加速率を高めるとともにベクトル化可能な部分を多くすることが必要である．このベクトル化可能な部分の比率 Tv/(Tu＋Tv) は重要な因子であり，ベクトル化率と呼ばれる．本性能評価では Tv の測定は集合演算ユニットのソフトウェアシミュレータを作成して行った．すなわちシミュレータ

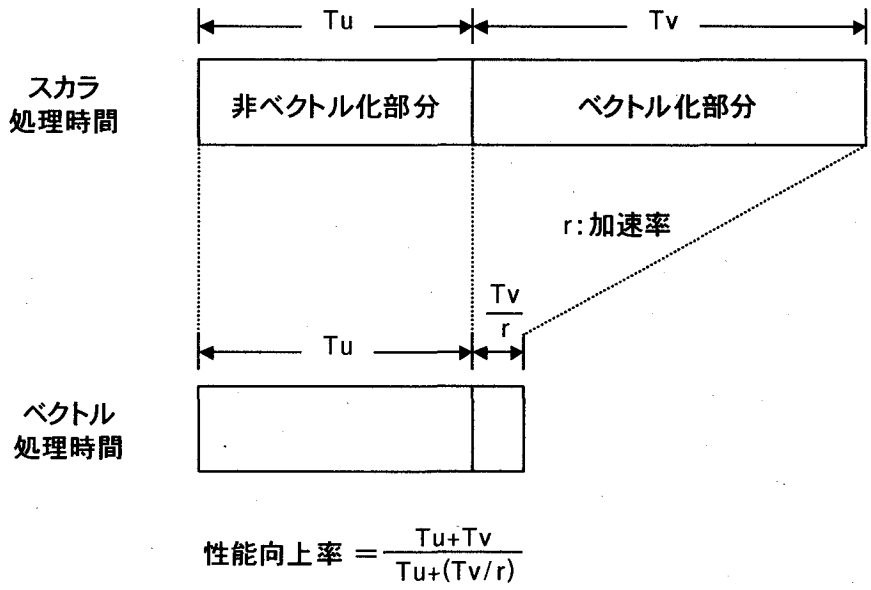

図 4.12 ベクトル処理による高速化効果

を使ったベクトル型関係データベース管理システムにおける SQL コマンド処理時間（Tu+Tu）の中で，シミュレータのみの走行時間を測定しこれを Tv としてベクトル化率を算出している．

前記ベンチマークによる性能実測結果を図 4.13 の四つのグラフにより示す．最初のグラフは，10 コマンド全体の処理に要する CPU 時間を，また 2 番目から 4 番目のグラフは各検索コマンド毎の CPU　処理時間を示している．各グラフにおいては，レコード単位処理方式の従来型の商用関係データベース管理サーバでの処理時間，動的ベクトル化方式を採用したベクトル型関係データベース管理サーバを集合演算ユニットのシミュレータを使って動作させた場合の処理時間，およびベクトル型関係データベース管理サーバを集合演算ユニットを使って動作させた場合の三つが比較されている．集合演算ユニットのソフトウェアシミュレータを用いての測定も行ったのは，前述のベクトル化率の測定と，動的ベクトル化方式における処理粒度拡大効果とを把握するためである．

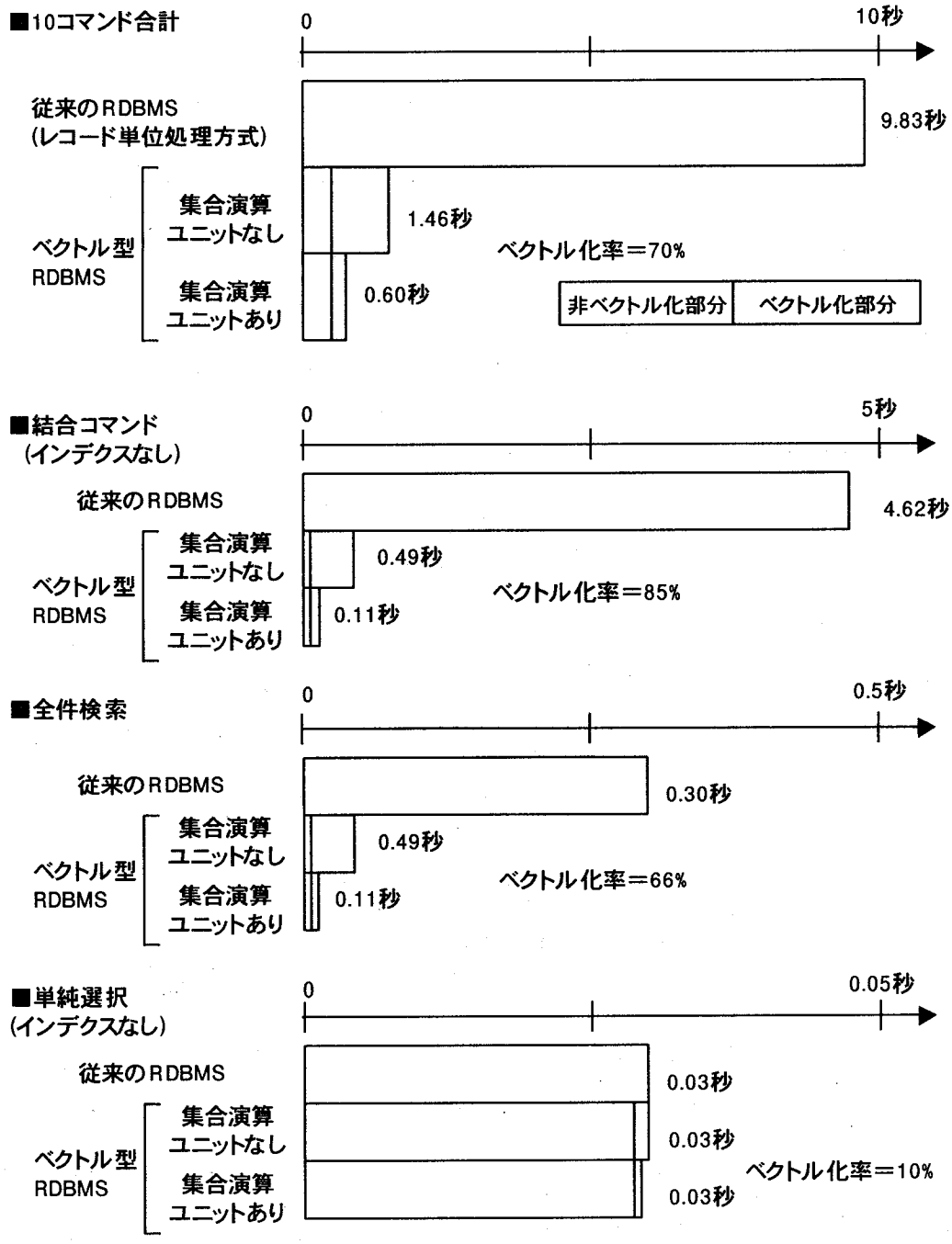

図 4.13 コマンドレベル性能

なお，主記憶上のデータベースバッファ（ページバッファ）のサイズは従来型とベクトル型で等しく，各テーブルおよびインデクスが入りきる容量とした．したがって本ベンチマークにおいては最初の検索時に必要なテーブルおよびインデクスがすべて二次記憶から

データベースバッファ上にロードされ，以後常駐化（キャッシング）される．すなわちキャッシング効果については従来型とベクトル型の間で差が生じない環境で測定した．

図 4.13 が示すように，このベンチマークでは処理粒度拡大のソフトウェア的効果は大きく，集合演算ユニットの効果と合わせベクトル型関係データベース管理サーバは従来の関係データベース管理サーバに比べ CPU 時間で約 15 倍の高速化を達成している．特に結合処理はベクトル化率も 85%と高く，性能改善の度合が著しい．反面，ベクトル型関係データベース管理サーバはインデクス検索に対しては効果が少ない．この原因は，インデクス検索では対象レコードがインデクスによりあらかじめ絞り込まれるので粒度拡大効果がほとんどないためと，ベクトル化率が 10%程度と低いのが原因である．ベクトル化率が低い理由はインデクスが B-tree 形式で，その探索のベクトル化が困難なためである．ベクトル化率を向上させるにはよりベクトル化に適したインデクス構造の採用を検討する必要がある．

図 4.14 にベクトル型関係データベース管理サーバの CPU 処理の内訳を示す．集合演算ユニットのソフトウェアシミュレータを使った場合では，全処理の 58%を前処理であるデュ

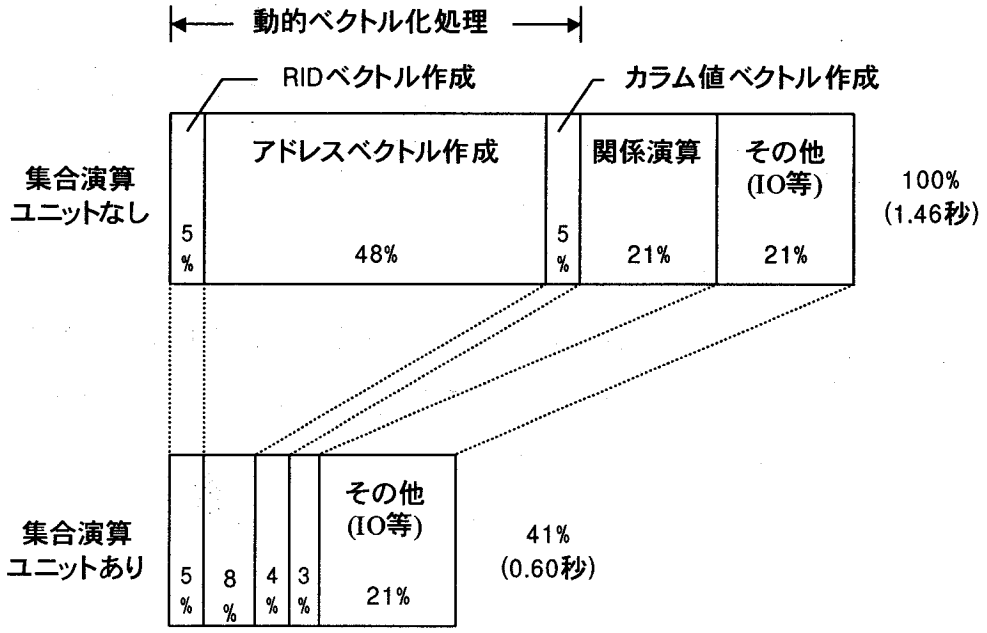

図 4.14 ベクトル型関係データベースの処理内訳

アルベクトル作成処理に要しており，そのオーバヘッドは大きい．集合演算ユニットを使った場合，デュアルベクトル作成オーバヘッドは約 1/3 に削減されており，4.3 節で述べたデュアルベクトル作成処理自体のベクトル化は有効な手法といえる．

4.6 結言

関係データベース処理を大容量主記憶に動的にベクトル展開してプログラマブルな集合演算ユニットによって一括処理する，マルチメディアデータベースサーバの高速化方式について述べた．集合演算ユニットの命令セットアーキテクチャを提案するとともに，データベース処理用の CPU 内蔵型のアクセラレータとして実装した．さらに提案方式の性能評価を行い，以下の結果を得，本高速化方式の有効性を確認した．

(1) 集合演算ユニットは命令レベルでは通常のスカラ処理に比べ約 10 倍高速である．

(2) テーブル結合検索を含む非定型的検索処理のベンチマークでは，集合演算ユニットを用いたベクトル型関係データベース管理システムは従来のレコード単位処理方式のシステムに比べ，約 15 倍の性能改善を達成した．ベクトル化率としては約 70％を実測した．

IDP とベクトル型関係データベース管理システムは商用システムとして出荷され，国内 300 システム以上の稼動実績を有する [107]．今後の課題として以下があげられる．

(1) 集合演算ユニットの機能向上

性能をさらに改善するには，集合演算ユニットの適用率を向上させる必要がある．例えば，適用の対象となるデータ構造を木やグラフに拡張してインデクス処理に対する適用率を向上させたり [108]，演算の種類を単純な探索からパターンマッチング処理へと拡張してより広範囲な検索に対応させる，といったことが考えられる [109]．

(2) 大型ストレージシステムへの適用

RAID (Redundant Array of Inexpensive Disks) と SAN (Storage Area Networking) 技術の進歩により，ストレージシステムの大容量化と高機能化が進んでいる．特にストレージシステムは大容量のディスクキャッシュを有しており [110]，ストレージシステム内に関係データベース管理システムを常駐させ，コンピュータ側の主記憶でなく，ストレージシステ

ムのディスクキャッシュ上にデータを展開して集合演算を適用するアプローチも有望と考えられる.

(3) プライバシ保護

マルチメディアデータベースサーバ内には，コンシューマの個人情報が格納されるため，プライバシの侵害への対策が重要な課題となっている[111]．脅威の種類としては，ネットワーク経由で外部からデータベースサーバに侵入される場合と，サービスプロバイダの内部から不当にアクセスされる場合が想定される．対策としては格納データの暗号化や，検索要求を発行する機器やプログラムをデータベース管理システムとして認証する機能の追加が有効と考えられる．これらの機能追加は検索性能を低下させる要因となる可能性が高く，耐タンパ性の実現を含め，ハードウェアによるプライバシ保護の支援方式の研究も今後重要になると予想される.

第5章　結論

5.1　本研究のまとめ

サービスプロバイダにとって付加価値の高いマルチメディアサービスの提供を容易化することを目的として，新たな専用プログラマブルプロセッサと，それを用いたマルチメディア処理方式を提案した．具体的には，次の2点の達成を目標とした．

(1) メディアプロセッサの高性能とプログラマビリティを活かし，マルチメディア端末における新機能追加や新規格対応を迅速に行うことで，新サービス提供を加速化する．

(2) サービス企画者による，大量データのマイニングのような非定型的検索を，データベースプロセッサを用いて高速化することで，十分な顧客分析を可能とし，コンシューマの新サービスに対する満足度を向上させる．

第 1 章において，従来研究について議論し，上記目標の達成に向けて解決すべき課題として以下が重要であることを明らかにした．

(1) メディアプロセッサにおけるソフトウェア生産性の向上

メディアプロセッサの並列性を引き出すためには，アセンブラ言語による高いスキルを要する複雑なプログラミング作業が必要で，生産性は低く変更が困難で再利用も難しい．

(2) メディアプロセッサ応用端末におけるソフトウェア更新後の実時間応答性の保証

メディアプロセッサ用のソフトウェアをネットワークからダウンロードして新サービスに対応する場合，映像圧縮のような高負荷処理の追加においては，端末のハードウェア資源量の制約から，機能追加後に実時間応答性を保証するのが難しい．

(3) データベースプロセッサにおける定型的検索と非定型的検索の両方の高速化

データベースプロセッサは，データベースの関係する領域を一括して並列に処理するため，非定型的で大量データをアクセスする検索に対しては高速であるが，少数のデータを定型的に処理するような並列度の低い検索に対しては効率が低い．

本論文ではこれらの課題を解決する方式の提案とその評価を，第 2 章から第 4 章に分けて述べた．

第 2 章では，メディアプロセッサにおけるソフトウェア生産性の課題を解決するメディ

アプロセッサ HMPV を提案した．具体的にはメディアプロセッサ用の C コンパイラの最適化可能性を分析評価し，その結果からコンパイラによる最適化を支援するための情報を命令フォーマットに埋め込むと同時に，データ転送専用プロセッサを導入してデータキャッシュのヒット率を向上させる方式を提案した．提案方式を LSI として実装し，C 言語で記述したデジタル TV のデコーダソフトウェアと最適化コンパイラを用いて評価を行い，以下の結果を得た．

(1) 論理規模 11M トランジスタ，DVD 再生時の消費電力は約 800mW

(2) 従来アーキテクチャに対し 22%の面積増加で，約 3 倍の性能向上

(3) アセンブラ言語を用いた人手による最適化に比べた C コンパイラの性能差は約 12%．

以上により，コンシューマ応用に適用可能なコストと，C 言語によるプログラミングで十分な性能を得られることを確認し，提案アーキテクチャの有効性を実証した．

第 3 章では，メディアプロセッサ応用端末におけるソフトウェア更新後の実時間応答性保証の課題を解決する端末ソフトウェアの設計手法を提案した．前章で提案した HMPV を適用して主要な端末機能をソフトウェア部品化し，各ソフトウェア部品の資源消費プロファイルと，サービスにおけるソフトウェア部品の並行実行度から，ソフトウェア更新後の資源消費量を予測するためのフレームワークを与えた．その上で，画質等の処理品質を変えると消費資源量も変化することに着目し，処理品質をサービスの並行性に応じて動的に制御することにより，実時間応答性を保証する方式を提案した．提案設計手法に基づいてマルチメディア端末を試作し，初期設計に対して複数のサービス機能追加を試行した．その結果は以下であり，提案設計手法の有効性と実用性を確認した．

(1) 与えられたハードウェア資源制約下で，目標品質の±10%内での実時間応答を確認

(2) 消費資源量の予測値と実測値の差は 5%以下

(3) 主要なソフトウェア部品の開発工数は約 40 人月

(4) ソフトウェア部品が揃っている場合，新サービスの追加工数は約 3 人月

第 4 章では，データベースプロセッサにおける定型的検索と非定型的検索の両方の高速化という課題を解決する，マルチデータベースサーバの高速化方式を提案した．提案方式は，索引を利用できる定型的な検索についてはレコード単位処理方式を適用し，索引が使えない非定型的な検索に対しては，大容量の主記憶上に必要なデータをベクトル形式で動

的に展開し，集合演算専用のプログラマブルプロセッサでパイプライン処理することで処理効率を向上させた．本方式に基づくデータベースプロセッサを大型汎用コンピュータに付加する集合演算ユニットとして実装し，関係データベース管理システムに適用して以下の結果を得て本方式の有効性を確認した．

(1) 集合演算ユニットは命令レベルでは通常のスカラ処理に比べ約 10 倍の処理性能を実測

(2) テーブル結合検索を含む非定型的検索処理のベンチマークでは，集合演算ユニットを用いたベクトル型関係データベース管理システムは従来のシステムに比べ，ベクトル化率約 70%で，約 15 倍の性能改善を達成した．

5.2 今後の課題

インターネットや広帯域無線通信技術の進展により，「いつでも，どこでも，誰にでも」サービスが提供できるいわゆるユビキタス情報社会が到来しつつある [112]．このような流れの中，今後マルチメディアサービスがより広範に普及するためには，本研究で目標とした「質の高いサービスを低価格で提供する」，ということに加え，まず，「安心してサービスが利用できる環境を整備する」ことが重要である．既にインターネットにおいては，さまざまな著作権侵害やセキュリティ，プライバシの問題が顕在化しており [85]，それらを解決することがマルチメディアサービスシステム全体として次の最も大きな課題である．

メディアプロセッサについては，著作権保護と機器認証への対応が重要である．著作権保護の新しい手法や規格が次々と提案されており，これらに柔軟に対応できるアーキテクチャの開発が課題である．特により安全性の高い楕円暗号や高速なストリーム暗号 [113]，あるいは電子透かし [72] といったアルゴリズムへの適合性を高めることが重要と考えられる．また，ネットワークに特殊な機器を接続してサービスを不正に受けるといった行為への対策として，機器認証の必要性が認識されており，プロセッサの効率の良い認証方式と耐タンパ性の向上 [73] も研究課題である．

端末ソフトウェアの設計手法に関しても，サービスプロバイダからのソフトウェアダウンロードが一般化すると，種々のセキュリティ上の問題が顕在化すると予想される．例えば，サービスプロバイダのダウンロード用のサーバになりすまされて，アクセスした端末にウィルスが送り込まれるといった事態が想定できる．サービス，アプリケーション，ならびに個々のソフトウェア部品を認証するフレームワークを提供する必要がある．企業内

で閉じてセキュリティを確保するとの観点からは，既にビジネス向けのPCにおいて標準化も含めて提案が行われている[86]．よりオープンなネットワーク環境への対応が必要となるコンシューマ向けの端末でどういう枠組みを作るのが良いかは今後の研究課題である．

マルチメディアデータベースサーバにおいては，多数のコンシューマの個人情報が蓄積されるため，プライバシ保護が最大の課題である．既にいくつかのサービスプロバイダにおいて個人情報の流出が実際に起き，大きな社会問題となっている．これらはいずれも内部の人間が顧客情報を直接的に読み出したものであるが，データベース内の複数のデータから顧客情報を演繹的に合成される可能性も考慮する必要があり[114]，今後のデータベース管理システムにとって主要な研究課題である．

マルチメディアサービスシステムがコンシューマに広く利用されるための，もう一つの大切な課題は，いうまでもなく「誰にでも簡単に使える」ことである．PCや携帯電話に代表される高機能なデジタル機器は，多くのコンシューマにとって使いこなすのが難しく，使い勝手の向上が強く望まれている．これを実現するアプローチとして，各種の自動化によってコンシューマの機器操作を簡略化するものと，現在のGUI(Graphical User Interface)にこだわらないユーザインタフェースを用いて操作をより直感的にわかり易くするものがあげられる[115] [116]．また，高齢者や子供などの情報弱者に対する配慮も課題となっており，コンシューマにとっての使い勝手を中心として，デジタル機器の機能やインタフェースを根本的に見直す時期に来ていると考えられる．

謝辞

本研究の全過程を通じ，懇切なるご指導，ご鞭撻を賜った大阪大学大学院情報科学研究科マルチメディア工学専攻 薦田憲久教授に心から感謝の意を表します．

本研究をまとめるにあたり，貴重なお時間を割いていただき，丁寧なるご教示を賜りました大阪大学大学院情報科学研究科情報システム工学専攻 尾上孝雄教授，マルチメディア工学専攻 岸野文郎教授に謹んで深謝の意をささげます．

大学院博士後期課程において，マルチメディア工学全般に関して，親切なるご指導とご助言を賜わりました大阪大学大学院情報科学研究科マルチメディア工学専攻　西尾章治郎教授，藤原融教授，下條真司教授に深く感謝申し上げます．

本研究を進めるにあたり，第 2 章の実時間マルチメディア処理向けシステムオンチップに関する研究ならびに第 3 章のマルチメディア端末のソフトウェア設計手法に関する研究は，(株) 日立製作所システム開発研究所およびマイクロデバイス事業部において多数の方々のご指導，ご助力を得て実施したものである．本研究の機会を与えて頂くとともにその方向性についてご指導いただいたシステム開発研究所の元所長 春名公一氏，元所長 片岡雅憲氏（現 (株) 日立 INS 社長），元所長 和歌森文男氏（現 (株) 日立情報システムズ），坂東忠秋主管研究長，元主管研究員 森文彦氏（現法政大学客員教授），マイクロデバイス事業部の元主管技師長 細坂啓氏，元事業部長 安斎昭夫氏，　小高正則事業主管，河路幹規本部長に心からお礼申し上げます。

また第 4 章のマルチメディアデータベースサーバの高速化に関する研究は，（株）日立製作所中央研究所とエンタープライズシステム事業部，およびソフトウェア事業部において多数の方々のご指導，ご助力を得て実施したものである．本研究の方向性についてご指導いただいた中央研究所の元所長 堀越彌氏（現 (株) 日立情報システムズ社長），元主管研究長 千葉常世氏，エンタープライズシステム事業部 元本部長 河辺峻氏（現中央研究所），ソフトウェア事業部 元主任技師 高橋政美氏に深く感謝いたします．

本研究の共同研究者として多大なるご協力を頂きました (株) 日立製作所研究開発本部デジタルアプライアンス研究センタの西岡清和部長，野尻徹主任研究員，川口敦生主任研究員，藤井由紀夫主任研究員，田代卓主任研究員，藤川義文主任研究員，田中和彦主任研究員，山本一道研究員，東嶋重樹研究員，藤平達研究員，石黒正雄研究員，中田啓明研究員，廣井和重研究員，細木浩二研究員，江濱真和研究員に，また（株）日立製作所中央研究所の元主任研究員 鳥居俊一氏（現技術研修所シニアプランニングマネージャ），三科雄介主任研究員，高本良史主任研究員に

心から感謝致します．本研究を進めるにあたり，データ提供，助言やディスカッションをして頂きました(株)日立製作所マイクロデバイス事業部 岩渕真人部長，山岸幹生統括主任技師，宮崎健司主任技師，作田俊之主任技師，鍬田真主任技師に感謝致します．

本研究の推進に賛同くださり，大阪大学大学院博士後期課程に在学することへのご配慮と援助を賜りました(株)日立製作所システム開発研究所 小坂満隆所長ならびに舩橋誠壽主管研究長に感謝申し上げます．

さらに，本論文の執筆にあたり，何かと便宜を図って頂き，有益なるご助言，お心遣いを頂きました大阪大学大学院情報科学研究科マルチメディア工学専攻 大川剛直助教授に心から感謝の意を表します．また，文献の調査や原稿作成を支援いただいた（株）日立製作所インターネットプラットフォーム事業部の諸星晴美氏に感謝致します．

最後に，本論文の執筆にあたり，常に最高の環境を提供してくれた妻 小島陽子に心から感謝します．

参考文献

[1] 総務省："IT 社会発展の基礎となるブロードバンド化の進展，" 平成 13 年度版情報通信白書，pp. 3-45 (2001).

[2] ITU-T Rec. J. 112, "Data-Over-Cable Service Interface Specification" (2003).

[3] 八木伸行，吉村俊郎，加井謙二郎：放送技術読本－BS/CS/地上デジタル放送，オーム社 (2002).

[4] TIA/EIA/IS-856, "Cdma2000 High Rate Packet Data Air Interface Specification" (2003).

[5] ITU-R Rec. M. 1457, "Detailed specification of the radio interfaces of International Mobile Telecommunications-2000 (IMT-2000)" (2003).

[6] 上原実，丸山隆，渡辺健治，樋口正明："無線 LAN アクセスに対応したソリューションへの取り組み，" 日立評論，Vol. 84, No. 11, pp. 683-686 (2002).

[7] 安部保範，篠田雪久，桑原昌史，浅尾芳久，徳丸亀鶴，弘津研一，大埜廣治，下口剛史，小崎武嗣，矢田勝啓，三田雅樹，桑原雅裕，坂元広史："高速電力線通信装置の開発，" SEI テクニカルレビュー，第 164 号，pp. 67-71 (2004).

[8] 瀧塚博志："ホームネットワーク，" 情報処理，Vol. 41, No. 12, pp. 1356-1361 (2000).

[9] 田中真愉子，大野千代，山下洋史："簡単・安全を目指したホームネットワーク技術，" 日立評論，pp. 809-812, Vol. 86, No. 11 (2004).

[10] 総務省："デジタルネットワーク文化の発展とコンテンツの流通，" 平成 15 年度版情報通信白書，pp. 95-113 (2003).

[11] 尾関孝介，松澤俊彦，高田春樹，山本俊："ハイビジョンプラズマテレビの新ラインアップ，" 日立評論， Vol. 84, No. 11, pp. 673-678 (2004).

[12] Date, C. J. : *An Introduction to Database Systems*, 3rd ed., pp. 171-180, Addison-Wesley (1981).

[13] 竹内理，Damien, L. M. : "HiTactix-BSD 連動システムを応用した大規模双方向ストリームサーバの設計と実装，" 情報処理学会論文誌，Vol. 43, No. 1, pp. 137-145 (2002).

[14] ISO/IEC 13818, "Information Technology - Generic coding of moving pictures and associated audio information" (1999).

[15] 鈴木輝彦："MPEG-4 AVC | H. 264 の概要と標準化動向，" 情報処理学会オーディオビジ

ュアル複合情報処理研究会, No. 38-13, pp. 69-73 (2002).

[16] Cheung, M. L., Kin, M. L., and Wan, C. S. : "A fast fractal image coding based on kick-out and zero contrast conditions," *IEEE Trans. on Image Processing*, Vol. 12, No. 11, pp. 1398- 1403 (2003).

[17] Claypoole, R. L., Davis, G. M., Sweldens, W., and Baraniuk, R. G. : "Nonlinear wavelet transforms for image coding via lifting," *IEEE Trans. on Image Processing*, Vol. 12, No. 12, pp. 1449-1459 (2003).

[18] Horowitz, M., Joch, A., Kossentini, F., and Hallapuro, A. : "H. 264/AVC baseline profile decoder complexity analysis, " *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 13, No. 7, pp. 704- 716 (2003).

[19] 小柳滋, 久保田和人, 仲瀬明彦 : "Matrix Clustering: CRM 向けの新しいデータマイニング手法," 情報処理学会論文誌, Vol. 42, No. 8, pp. 2156-2165 (2001).

[20] 白石肇, 西尾誠一, 伊藤聡 : "ASIC 手法を活用したシステム VLSI の開発," 情報処理学会計算機アーキテクチャ研究会, No. 69-12, pp. 89-95 (1988).

[21] 桜井優, 澤繁隆, 石川達也, 吉岡健次, 甲斐直行, 名古屋哲雄, 池田一雅, 真中重之, 奥山武彦, 新舟剛夫 : "BS デジタルハイビジョンテレビ用 LSI," 東芝レビュー, Vol. 55, No. 8, pp. 44-57 (2000).

[22] 月岡陽一, 白田和己, 山田有 : "IPv6 ネットワークへの移行ソリューション," 日立評論, Vol. 86, No. 8, pp. 551-553 (2004).

[23] Adiletta, M., Rosenbluth, M., Bernstein, D., Wolrich, G., and Wilkinson, H. : "The Next Generation of Intel IXP Network Processors, " *Intel Technology Journal*, Vol. 6, No. 3, pp. 6-18 (2002).

[24] Gwennap, L. : "New PA-RISC Processor Decodes MPEG Video, " *Microprocessor Report*, Vol. 8, No. 1, pp. 16-17 (1994).

[25] Peleg, A., Wilkie, S., and Weiser, U. : "Intel MMX for Multimedia PCs, " *Comm. of the ACM*, Vol. 40, No. 1, pp. 25-38 (1997).

[26] Chen, Y., Holliman, M., Debes, E., Zheltov, S., Knyazev, A., Braranov, S., Belenov, R., and Santos, I. : "Media Applications on Hyper-Threading Technology, " *Intel Technology Journal*, Vol. 6, No. 1, pp. 47-57 (2002).

[27] Gunther, S. H., Binns, F., Carmean, D. M., and Hall, J. C. : "Managing the Impact

of Increasing Microprocessor Power Consumption, " *Intel Technology Journal*, 2001 Q1 issue (2001).

[28] Guttag, K., Gove, R. J., and Van Aken, J. R.: "A Single-Chip Multiprocessor For Multimedia: The MVP, " *IEEE Computer Graphics & Applications*, Vol. 12, No. 6, pp. 53-64 (1992).

[29] S. Rathnam and G. Slavenburg, "An Architectural Overview of the Programmable Multimedia Processor TM-1, " in *Proc. of 41st IEEE International Computer Conference*, pp. 319-326 (1996).

[30] 吉田豊彦:"VLIW 型メディアプロセッサ," 情報処理, Vol. 38, No. 6, pp. 499-506 (1997).

[31] P. Kalapathy: "Hardware-Software Interactions on Mpact, " *IEEE Micro*, Vol. 17, No. 2, pp. 20-26 (1997).

[32] 森下宏之, 平井誠, 木村浩三, 清原督三:"HDTV 対応メディアプロセッサ MCP2 におけるストリーム処理部の開発," 情報処理学会ハイパフォーマンスコンピューティング研究会, No. 85-6, pp. 31-36 (2001).

[33] 小島啓二, 西岡清和, 野尻徹, 宮崎健司:"メディアプロセッサ (MAP) のビジョンとアーキテクチャ," 第 59 回情報処理学会全国大会, 情報家電とホームネットワーク特別セッション, 1C-1, pp. 87-92 (1999).

[34] 岩崎裕江, 落合克幸, 長沼次郎, 遠藤真, 小倉武:"マルチメディア用大規模システム LSI のコンカレントデザイン環境," 電子情報通信学会論文誌, Vol. J84-DI, No. 6, pp. 548-557 (2001).

[35] 植村俊亮:"CODASYL 方式のデータベース・システム," 情報処理, Vol. 17, No. 10, pp. 892-903 (1976).

[36] 上林弥彦:"関係データベース言語 －データベースの基礎理論 (4) －," 情報処理, Vol. 24, No. 2, pp. 176-187 (1983).

[37] 溝口徹夫:"B-tree によるデータ管理," 情報処理, Vol. 21, No. 7, pp. 769-776 (1980).

[38] Babb, E.:" Implementing a Relational Database by means of Specialized Hardware, " *ACM Trans. on Database System*, Vol. 4, No. 1, pp. 1-29 (1979).

[39] Ozkarahan, E. A.:" Evolution and Implementations of the RAP Database Machine, " *New Generation Computing*, Vol. 3, No. 3, pp. 237-271 (1985).

[40] 田中譲:"データフロー制御データストリーム処理方式データベースマシン (基本構

想),” 情報処理学会データベース管理システム研究会, No. 28-3, pp. 1-8 (1982).

[41] Kitsuregawa, M., Tanaka, H., and Moto-oka, T.: “Architecture and Performance of the Relational Algebra Machine GRACE,” in *Proc. of International Conference on Parallel Processing*, pp. 241-250 (1984).

[42] Neches, P. M. and Shemer, J.: “The Genesis of a Database Computer,” *IEEE Computer*, Vol. 17, No. 11, pp. 42-56 (1984).

[43] 疋田定幸, 川上英, 岸田巧, 羽生田博美:“リレーショナルデータベースマシン FREND,” 情報処理学会データベースシステム研究会, No. 39-2, pp. 1-8 (1984).

[44] 喜連川優:“データベースマシン,” 情報処理, Vol. 28, No. 1, pp. 56-67 (1987).

[45] 田中和彦, 鍬田真, 野尻徹, 西岡清和:“VLIW プロセッサにおける命令圧縮方式,” 第63回情報処理学会全国大会, 5K-5, pp. 37-38 (2001).

[46] 細木浩二, 東嶋重樹, 田代卓, 川口敦生, 西岡清和:“メディアプロセッサ MAPCA のデータ転送方式,” 情報処理学会計算機アーキテクチャ研究会, No. 146-16, pp. 91-95 (2002).

[47] 小島啓二, 西岡清和, 川口敦生, 細木浩二, 薦田憲久:“ソフトウェアによる実時間デジタルメディア処理向けシステムオンチップ,” 電子情報通信学会論文誌 D-I, Vol. J87-D-I, No. 12, pp. 1051-1059 (2004).

[48] 廣井和重, 田代卓, 川口敦生, “VLIW 型メディアプロセッサを用いた MPEG-2 オールフォーマットビデオデコーダ,” 映像情報メディア学会誌, Vol. 56, No. 5, pp. 102-111 (2001).

[49] 石黒正雄, 藤川義文, 山口宗明, 廣井和重, 鈴木教洋, 川口敦生:“メディアプロセッサにおける可変長復号処理の高速化,” 電子情報通信学会技術研究報告, CQ2002-68, pp. 19-24 (2002).

[50] 石黒正雄, 藤川義文, 山口宗明, 廣井和重, 鈴木教洋, 川口敦生: “VLIW 型メディアプロセッサを用いた MPEG-4 ビデオデコーダ,” 電子情報通信学会技術研究報告, IE2001-120, pp. 31-36 (2001).

[51] 小島啓二, 西岡清和, 川口敦生, 薦田憲久: “メディアプロセッサを応用したブロードバンド端末のソフトウェア設計手法,” 電子情報通信学会論文誌 D-I (投稿中).

[52] 鳥居俊一, 小島啓二, 坂田明治, 井門徳安, 吉住誠一:“ベクトル型高速データベースプロセッサ -ソフトウェア方式-,” 情報処理学会第 32 回全国大会, pp. 917-918

(1986).

[53] 吉住誠一，鳥居俊一，小島啓二，坂田明治，井門徳安："ベクトル型高速データベースプロセッサ -基本構想-，" 情報処理学会第 32 回全国大会，pp. 915-916 (1986).

[54] 小島啓二，鳥居俊一，坂田明治，吉住誠一："ベクトル型高速データベースプロセッサ - アーキテクチャ - ，" 情報処理学会第 32 回全国大会，pp. 919-920 (1986).

[55] 小島啓二，鳥居俊一，坂田明治，高本良史，吉住誠一，河辺峻："ベクトル型高速データベースプロセッサ IDP - ソート性能評価 - ，" 情報処理学会第 33 回全国大会，pp. 871-872 (1986).

[56] Kojima, K., Torii, S., and Yoshizumi, S.: "IDP - A Main Storage Based Vector Database Processor -," in Kitsuregawa, M. and Tanaka, H. eds. "*Database Machines and Knowledge Base Machines,*" pp. 47-60, Kluwer Academic Press (1988).

[57] 小島啓二，鳥居俊一，吉住誠一："ベクトル型データベースプロセッサ IDP，" 情報処理学会論文誌，Vol. 31, No. 1, pp. 163-173 (1990).

[58] 小島啓二，鳥居俊一："データベース処理方法，" 日本特許第 1998039 号 (1995).

[59] 小島啓二，鳥居俊一，井門徳安："ベクトル処理装置，" 日本特許第 2080498 号 (1996).

[60] Conklin, G. J., Greenbaum, G. S., Lillevold, K. O., Lippman, A. F., and Reznik, Y. A.: "Video coding for streaming media delivery on the Internet," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 11, No. 3, pp. 269-281 (2001).

[61] 矢幡明樹："DSP のモデムへの応用，" 情報処理，Vol. 30, No. 11, pp. 1315-1323 (1989).

[62] Turley, J.: "Multimedia chips complicate choices," *Microprocessor Report*, No. 10, pp. 14-19 (1996).

[63] Fritts, J., Wolf, W., and Liu, B. : "Understanding multimedia application characteristics for designing programmable media processors," *Proc. of SPIE*, vol. 3655, pp. 2-13 (1998).

[64] Hennessy, J. L. and Patterson, D. A. : "*Computer Architecture: A Quantitative Approach,*" Morgan Kaufmann Publishers, Inc. (1990).

[65] Corwell, R. P., Nix, R. P., O' Donnell, J. S., Papworth, D. B., and Rodman, P. K. : "A VLIW architecture of a trace scheduling compiler, " *IEEE Trans. on Computers*, Vol. 37, No. 8, pp. 967-979 (1988).

[66] Muchnick, S. : "Advanced compiler design and implementation, " Morgan Kaufmann,

San Francisco (1997).

[67] 田中義一, 岩澤京子："ベクトル計算機のためのコンパイル技術," 情報処理, Vol. 31, No. 6, pp. 736-743 (1990).

[68] 西田政人, 中村俊彦, 西垣泰洋, 有泉武仁, 多賀谷聡："SX-6 の本体系ハードウェア," NEC 技報, vol. 55, No. 9, pp. 11-15 (2002).

[69] Aikawa, M., Takaragi, K., Furuya, S., and Sasamoto, M. : "A lightweight Encryption Method Suitable for Copyright Protection, " *IEEE Trans. on Consumer Electronics*, Vol. 44, No. 3, pp. 902-910 (1998).

[70] 近藤伸和, 角田元泰, 川口敦生, 廣井和重："ユビキタスサービスシステムを支えるクライアント技術," 日立評論, Vol. 84, No. 11, pp. 711-714 (2002).

[71] 櫻井紀彦, 木俵豊, 高嶋洋一, 谷口展郎, 難波功次："コンテンツ流通における著作権保護技術の動向," 情報処理学会論文誌, Vol. 42, No. SIG15, pp. 63-76 (2001).

[72] 越前功："著作権保護と電子　透かし技術," ユビキタス時代の情報セキュリティ技術, 瀬戸洋一編, 日本工業出版, pp. 153-176 (2003).

[73] 北原潤："暗号のハードウェア実装技術," ユビキタス時代の情報セキュリティ技術, 瀬戸洋一編, 日本工業出版, pp. 96-106 (2003).

[74] 佐藤幸一："量産は FPGA を使うのか, ASIC を使うのか －ユーザが見る ASIC の課題と FPGA の使い方－," 情報処理学会システム LSI 設計技術研究会, No. 108-9, pp. 49-53 (2003).

[75] 一之瀬進："ネットワークアプライアンス," NTT の光新世代ビジョン「RENA」の全貌, 日経ニューメディア別冊, pp. 187-199 (2003).

[76] 電波産業会："デジタル放送におけるアプリケーション実行環境標準規格," ARIB STD-B2 (2003).

[77] 渡邉真也, 廣安知之, 三木光範："近傍培養型遺伝的アルゴリズムによる多目的最適化," 情報処理学会論文誌, Vol. 43, No. SIG 10, pp. 183-198 (2002).

[78] 守屋悦朗："チューリングマシンと計算量の理論," 情報数理シリーズ, 培風館 (1997).

[79] 坂和正敏, 田中雅博："遺伝的アルゴリズム," ソフトコンピューティングシリーズ 1, 朝倉書店 (1995).

[80] ITU-R Rec. Bt. 500-11, "Methodology for the subjective assessment of the quality of television pictures" (2002).

[81] 石黒正雄, 藤川義文, 山口宗明, 廣井和重, 川口 敦生, 鈴木教洋: “MAPCA 応用 MPEG-4 シンプルプロファイルビデオエンコーダ,” 2001 年映像情報メディア学会年次大会講演予稿集, pp. 93-94 (2001).

[82] 小島啓二, 松田芳樹 :“視覚的プログラミングシステム LIVE,” 情報処理学会第 30 回プログラミングシンポジウム, pp. 9-17 (1989).

[83] Kojima, K., Matsuda, Y., and Futatsugi, S.: “LIVE - Integrating Visual and Textual Programming Paradigms -,” in *Proc. of the 1989 IEEE Workshop on Visual Languages*, pp. 80-85 (1989).

[84] Kojima, K. and Myers, B. A.: “Parsing Graphic Function Sequences,” in *Proc. of the 1991 IEEE Workshop on Visual Languages*, pp. 112-117 (1991).

[85] 佐々木良一 :“インターネットセキュリティ入門,” 岩波新書 (1999).

[86] Trusted Computing Group, “TCG TPM Specification Version 1.2” (2003).

[87] Hagino, T., Honda, M., Koga, A., Kojima, K., Nakajima, R., Shibayama, E., Yuasa, T.: “The IOTA Programming System” in Nakajima, R. and Yuasa, T. eds. “*Lecture Notes in Computer Science*,” Vol. 160, Springer-Verlag (1983).

[88] 湯浅太一, 小島啓二, 柴山悦哉, 中島玲二, 萩野達也, 本田道夫 :“イオタ入門,” 情報処理学会記号処理研究会, No. 016-001, pp. 1-6 (1981).

[89] 湯浅太一, 小島啓二, 中島玲二 : “モジュラー・プログラミングのための支援環境,” 情報処理, Vol. 23, No. 5, pp. 433-441 (1982).

[90] Hsiao, D. K. : “Data Base Computers,” *Advances in Computers*, Vol. 19, pp. 1-64, Elsevier (1983).

[91] Boral, H. and Dewitt, D. J.: “Database Machines: An Idea Whose Time Has Passed?,” in Leilich, H. and Missikoff, M. eds., *Database Machines*, Springer-Verlag (1983).

[92] Dewitt, D. J., Gerber, R. H., Graefe, G., Heytens, M. L., Kumar, K. B. and Muralikrishna, M. : “GAMMA－A High Performance Dataflow Database Machine,” in *Proc. of Twelfth International Conference on Very Large Data Bases*, pp. 228-237 (1986).

[93] Dewitt, D. J., Katz, R., Olken, F., Shapiro, D., Stonebraker, M. and Wood, D. : “Implementation Techniques for Main Memory Database Systems,” in *Proc. of the 1984 SIGMOD Conference*, pp. 1-8 (1984).

[94] Torii, S., Kojima, K., Yoshizumi, S., Sakata, A., Takamoto, Y., Kawabe, S.,

Takahashi, M. and Ishizuka, T. : "A Relational Database System Architecture Based on a Vector Processing Method, " in *Proc. of the Third International Conference on Data Engineering*, pp. 182-189 (1987).

[95] Torii, S., Kojima, K., Yoshizumi, S., and Sakata, A. : "A Relational Database System Architecture Based on a Vector-Processing Method," *Information Sciences*, No. 48, pp. 135-155 (1989).

[96] Tanaka, Y. : "A Data Stream Database Machine with Large Capacity, " in Hsiao, D. K. ed. *Advanced Database Machine Architectures*, pp. 168-202, Prentice-Hall (1983).

[97] 金田泰, 小島啓二, 菅谷正弘 : "ベクトル計算機のための探索問題の計算法「並列バックトラック計算法」, " 情報処理学会論文誌, Vol. 29, No. 10, pp. 985-994 (1987).

[98] Kanada, Y., Kojima, K., and M. Sugaya: "Vectorization Techniques for Prolog," in *Proc. of 1988 ACM International Conference on Supercomputing*, pp. 539-549 (1988).

[99] Torii, S., Kojima, K., Kanada, Y., Sakata, A., Yoshizumi, S., and M. Takahashi: "Accelerating Non-Numerical Processing by an Extended Vector Processor," in *Proc. of the Fourth IEEE International Conference on Data Engineering*, pp. 194-201 (1988).

[100] 鳥居俊一, 小島啓二, 金田泰, 坂田明治, 高橋政美 : "マージ型ベクトル演算機構を用いた非数値処理の高速化方式," 情報処理学会論文誌, Vol. 34, No. 1, pp. 109-119 (1993).

[101] 石浦菜岐佐, 高木直史, 矢島脩三 : "ベクトル計算機上でのソーティング, " 情報処理学会論文誌, Vol. 29, No. 4, pp. 378-385 (1988).

[102] Hills, W. D. : "The Connection Machine, " The MIT Press (1985).

[103] Wada, K., Nagashima, S. and Odaka, T. : "Design for a High Performance Large-Scale General Purpose Computer, The HITACHI M-680H Processor" in *Proc. of IEEE International Conference on Computer Design: VLSI in Computers*, pp. 481-484 (1985).

[104] 掘越彌, 梅谷征雄 : "汎用計算機のための内蔵ベクトル演算方式, " 情報処理学会論文誌, Vol. 24, No. 2, pp. 191-199 (1983).

[105] 若井勝郎, 小川哲二, 安部秀一, 竹田克己, 久保完次, 和田健一, 木下俊之 : "大型

コンピュータ M-880 の処理方式とハードウェア,”日経エレクトロニクス, 1990 年 12 月 10 日号, No. 515, pp. 210-223 (1990).

[106] Knuth, D. E. :“Searching and Sorting, ” *The Art of Computer Programming*, Vol. 3, Addison-Wesley (1973).

[107] 鳥居俊一, 小島啓二, 吉住誠一, 河辺峻, 高橋政美, 久代康雄 :“リレーショナル・データベースの処理速度向上を図る CPU 内蔵型データベースプロセッサ, ” 日経エレクトロニクス, 1987 年 2 月 9 日号, No. 414, pp. 185-206 (1987).

[108] 安村通晃, 小島啓二 :“スーパーコンピュータ上でのグラフ問題の高速化,” 情報処理学会第 31 回プログラミングシンポジウム, pp. 9-17 (1990).

[109] Kojima, K. : “A Pattern Matching Algorithm in Binary Trees,” in R. Nakajima ed. “*Lecture Notes in Computer Science*,” Vol. 147, pp. 99-114, Springer-Verlag (1982).

[110] 大谷功, 宇田のぞみ, 高橋一敏 :“Harmonious Computing を支えるハードウェア,” 日立評論, Vol. 86, No. 6, pp. 431-436 (2004).

[111] 梅澤健太郎, 斎藤孝道, 奥乃博 :“プライバシーを重視したアクセス制御機構の提案,” 情報処理学会論文誌, Vol. 42, No. 8, pp. 2067-2076 (2001).

[112] 総務省 :“世界に拡がるユビキタスネットワーク社会の構築,” 平成 16 年度版情報通信白書, pp. 2-120 (2004).

[113] 佐藤尚宜, 古屋聡一 :“デジタル社会を支える暗号技術,” ユビキタス時代の情報セキュリティ技術, 瀬戸洋一編, 日本工業出版, pp. 25-46 (2003).

[114] 木村憲史, 橋本誠志, 井上明, 金田重郎 :“ネットワーク上での情報統合に対するプライバシー保護,” 情報処理学会論文誌, Vol. 41, No. 11, pp. 2985-3000 (2000).

[115] 湯浅寛子, 小島啓二 :“情報のブロードキャッチシステム,” 情報処理学会情報メディア研究会, No. 013-006, pp. 37-44 (1993).

[116] 小島啓二 : “ビジュアルインタフェースの研究動向と応用, ” Chapter 2. 9, ビジュアルインタフェース - ポスト GUI を目指して, bit 別冊, 共立出版, pp. 168-175 (1996).