データベース・システム 57－5
（１９８７．１．１９）

関係演算処理のための専用ハードウェアの構成とその評価

武田　英昭、中村　敏夫、北村　正
ＮＴＴ電気通信研究所

本報告は、大規模なテーブルに対する関係演算（主として結合演算）を高速に処理する専用ハードウェアの構成とその評価について述べたものである。
本ハードウェアでは、演算のO(n)化、処理の並列化、パイプライン化することを基本思想とし、結合演算をふるい落とし、整列、突合せの３つのフェーズに分けて処理する３フェーズ・ジョイン方式を採用している。
ふるい落とし専用回路を設計する際の要点として、ふるい落とし用ハッシング法の解析を行い、統合法を基本とした回転重ね合わせ法が有効であること、及び、回転重ね合わせ法で問題となる回転数の決定法を示している。また、整列回路では２ウェイマージ方式を基本とし、処理の中断を防ぐメモリ複写機能をもった整列回路の基本構成法及びバスの共通化を図った場合のハード量について考察している。

A STUDY OF DESIGN OF SPECIAL HARDWARE FOR RELATIONAL OPERATIONS

Hideaki TAKEDA, Toshio NAKAMURA and Tadashi KITAMURA
NTT Electrical Communicaiton Labotatories, NTT
1-2356, Take, Yokosuka-Shi, Kanagawa, 238-03 Japan

The point of design of database machine for large-scale information management using filter and sorter is described. For high-speed processing of relational operations, mainly join operation, the machine of this paper uses the method of deviding join processing into three phases (filtering, sorting and comparing) and processes records in parallel and pipeline. The filtering circuit is discussed with the choice of hashing function to remove unnecessary records and we propose a rotate-superimposing method based on a folding method. For the sorting circuit, we propose a sorter with the mechanism of memory copy based on 2-way merge sorter and discuss with making it into LSI.

1　はじめに

近年の情報化社会の発展に伴い、データベースシステムは、利用形態の多様化、大規模化への対応に向けた高機能化、高性能化が切実な要求となってきている。

また、ＬＳＩ技術の進歩に伴うハードウェア価格の低下を背景にデータベース処理専用のアーキテクチャを有する計算機（データベースマシン）の研究も活発に進められている[1]、[2]、[3]。

データベースマシンの研究に際して考慮されるデータモデルの多くは，関係データベースモデルである。今後予想されるデータベースの高水準化動向を考えると、今後とも、関係データベース処理の高速化がデータベースマシン研究の重要なテーマといえる。

関係データベース処理を大きく分類すると選択、準射影のように単一の関係に対する演算と、結合のように複数の関係に対する演算に分けることができる。本稿では、（データベース量、トラフィック等に関して）大規模なデータベースに対し、後者の演算の高速化を図ることを目的とした、データベースマシンの構成について考える。

データベースマシンでは上述の演算を高速化する手段として、ソータがよく利用される。しかし、処理の高速化のためにはソータの容量を関係の大きさの最大値に近いものに設定する必要があるため、大規模データベース処理という観点から見た場合、ハードウェアの実現性、利用効率の面から問題があった。

著者らは、関係の大きさが数100ＭＢ（10^6タプル程度）、データベース量が数10ＧＢ程度の大規模データベースを対象としたデータベースマシンを提案している[4]。本稿では、このデータベースマシンの構成要素である結合等の演算の高速化を図る専用ハードウェアにおける主要な実現方式とその評価について述べる。

以下、第２章では専用ハードウェアの概要、第３章ではふるい落としを行う専用ハードウェアのフィルタにおけるハッシング方法について、第４章では整列専用ハードウェアであるソータの構成について述べる。

2　専用ハードウェアの概要

2．1　演算方式の考え方

本装置の演算方式に関する基本的な考え方は以下の通りである。

（1）　演算のO(n)化

結合演算は単純なアルゴリズムで実現した場合、O(n^2)の処理時間が必要となり、結合対象タプル数(n)が大きくなるとこの時間は極めて大きくなる。本装置では、O(n)の処理時間で整列可能なソータを利用することにより、結合演算のO(n)化を実現する。

（2）　処理の並列化

結合演算を整列と突合せに分けて考えると、前者は２つの関係で独立な処理である。本装置では整列を関係単位で並列処理し、突合せは１ヶ所で行う。

（3）　処理のパイプライン化

演算を幾つかの処理に分割し、それらをパイプライン的に動作させることにより、処理の高速化とハードウェアの利用率の向上を図る。パイプライン動作の単位としては、バイト単位、タプル単位、適当な大きさのバッファ単位を適宜使用する。

以上の考え方に立ち、基本的な演算方式とし、結合演算処理を以下の３つのフェーズに分割し、パイプライン的に動作させる方式（３フェーズ・ジョイン方式[5]）と採った。

① 結合可能性のないキーをふるい落とす処理
② ふるい落とされたキーを整列する処理
③ 整列済みのキーの突合せとキー値の一致するタプルを連結する処理

2．2　専用ハードウェアの構成と機能

（1）　構成

処理負荷が大きく、並列／パイプライン処理による性能向上の効果が大きいと思われるふるい落とし処理と整列処理を専用ハードウェア化する。図２に本装置の構成概要を示す。図中、フィルタ、ソータは各々ふるい落とし処理、整列処理を行う専用ハードウェアであり、制御ＣＰＵではキーの突合せ・連結等を行う。

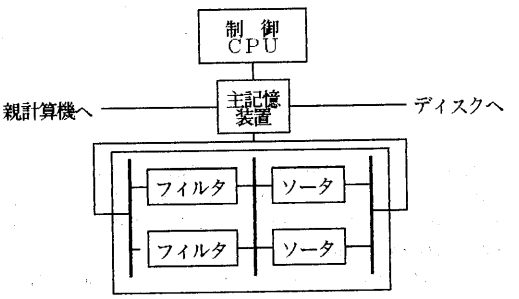


図2　装置の構成概要

以下にフィルタ、ソータの機能について説明する。

（2）　フィルタの機能

フィルタは、以下のような機能を有する。

① ふるい落とし機能

ハッシュ化ビットアレイ[6]を用いて、結合可能性のないキーをふるい落とす。結合する２つの関係はクロスハッシュ方式[7]を用いて、並列に処理する。

② キー抽出機能
タプルの中から整列対象となるキーを抽出する。
③ 可変長キーの固定長化機能
可変長のキーをソータに入力可能な固定長キーに変換する。

(3) ソータの機能
ソータは、以下のような機能を有する。
① 整列機能
キーをその値によって昇順に並べ替える。
② データ形式変換機能
文字データ、固定小数点データを整列可能な絶対値数に変換するとともに、降順整列データを昇順で整列可能な形式に変換する。
③ データ形式逆変換機能
データ形式を変換されたキーを元の形式に逆変換する。

キーから表の大きさに適合するように一定の桁を抽出する方法。
⑦ 乱数関数利用法
キーを乱数の初期値と見なし、乱数発生ルーチンを利用して乱数を求め、それを表の大きさに従って正規化する方法。
各ハッシング方法の定性的な比較を表3.1に示す。

表3.1 ハッシング・アルゴリズムの比

ハッシュ・アルゴリズム	処理速度(ソフト処理)	処理速度(ハード処理)	ハード量	データ分析*1	データ属性への対応		衝突の発生度合
					文字	整数	
除算法	○	△	△	不要	△	○	○
乗算法	○	△	△	不要	△	○	○
統合法	△	○	○	不要	○	△	△
基数変換法	×	×	×	不要	△	○	○
代数符号化法	○	△	×	不要	△	○	○
桁解析法	○	○	○	要	○	△	△
乱数関数利用法	×	−	−	不要	△	○	○

*1 データ分析：ハッシングのためにあらかじめデータの特性等を分析しておく必要性

3 フィルタにおけるふるい落とし用ハッシング・アルゴリズム

フィルタでは、ふるい落とし用ハッシング・アルゴリズムがその性能を決めるもっとも重要な要因となる。以下では、標準モデルによる解析と実データによるシミュレーションにより、ふるい落としに適したハッシング・アルゴリズムを見いだす。

3.1 従来のハッシング・アルゴリズム

データベース管理システムでは、従来、データ格納用などに以下のようなハッシング手法[8]、[9]が利用されている。

① 除算法
キーを表の大きさに従ったある定数で除し、その剰余をとる方法。
② 乗算法
キーをそれ自身，あるいはある定数と乗じ、その結果のビット列の中央付近から表の大きさに従って所要のビット数だけを抽出する方法。
③ 統合法
キーを幾つかの切片に区分し、各切片を加算、排他的論理和などで重ね合わせる行う方法。
④ 基数変換法
キーをｐ進数の数字列と見なし、それをｑ進数に変換した数字列から必要なビット数取り出す方法。
⑤ 代数符号化法
除算法において、キーを２進表現し、それを多項式の係数と見なし、表の大きさに従った多項式で剰余計算する方法。
⑥ 桁解析法

3.2 ふるい落とし用ハッシングの特徴

データベースの格納制御などに用いられるハッシングに比べ、ふるい落とし処理で用いるハッシングでは、以下のような特殊性がある。
ここで、ふるい落とし処理の概略は次の通りである；はじめに一方の関係をハッシングし、ハッシング結果に対応するビットアレイのビット位置に"1"を立てる。次に、他方の関係をハッシングし、そのハッシング結果に対応するビットアレイのビット位置に"1"が立っているもののみを有効とする。２つの関係の内、前者を"設定"関係、後者を"参照"関係と呼ぶこととする。

① ハッシュ表（ビットアレイ）の１バケットの大きさは１ビット。
② ビット列のため、以下に定義するロードファクタが大きくとれる。
③ 衝突に対する処理は不要。

$$\text{ロードファクタ}=\frac{\text{ビットアレイのバケット数}}{\text{参照データ数}}$$

④ ハッシュ表の空間効率よりも処理速度が重視される。
⑤ 可変長文字列への対処が必要である。

さらに、フィルタにおけるハッシングでは、以下のような条件がある。

⑥ 専用ハードウェアかを図るため、ハードウェア化が容易。
⑦ 語単位（例えば、１バイト）で入力し、入力に遅れることなく処理可能。
⑧ データ属性はデータベース定義情報で得られる範囲で既知である。

3.3 フィルタにおけるハッシング・アルゴリズム

(1) 基本アルゴリズム

フィルタでは、3.2述べたような特徴があるため、表3.1の中で、特にハード化を重視し、統合法を基本アルゴリズムとして採用した。

しかし、統合法による単純な重ね合わせでは以下のような問題点がある。

① データ集合のデータ値の偏りのハッシング結果への影響が大きいため、データ値に偏りがある場合はハッシング効果が悪い。

② 入力データ幅とハッシング結果の幅は一般に異なるため、ハッシング結果の幅によってはその間の変換回路が複雑になる。

そこで、入力したデータをずらしながら（ハッシングの中間結果を回転させながら）、重ね合わせる方式（回転重ね合わせ法）を採用することとした。

回転重ね合わせ法を実現する回路構成を図3.1に示す。

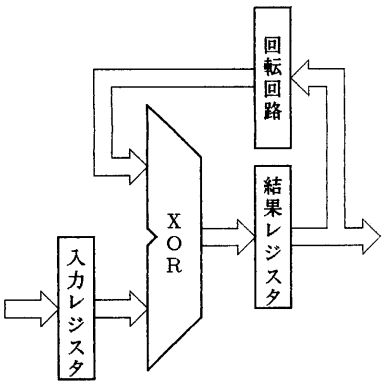


図3.1 回転重ね合わせ回路の構成

(2) 回転重ね合わせ法のハッシング効果

回転重ね合わせ法では、回転数の選択がハッシング効果を決める1つの要素である。本節では回転数とハッシング効果の関係について文字列データの場合について考察する。

(a) 前提条件

評価は文字データ（数字、英字、カタカナ）を対象とし、ふるい落とし以前に得られる情報はデータの属性のみとする（各ビットの出現確率等の情報は得られない）。解析モデルの前提条件は以下の通りである。

① 文字データで各文字種の出現する確率は等しいとする。

② ビットの出現する確率は他のビットの影響を受けないこととする。

③ データ長は6バイト、ハッシング結果の長さは16ビットとする。

④ 処理は1バイトごとに行う。

(b) 解析方法

各文字種の1バイトの各ビットに"1"の出現する確率で構成されるデータ（表3.2）を回転重ね合わせ法によりハッシングする。ハッシング結果の各ビットで"1"の出現する確率p_iに対し、次の評価関数B_sによって重ね合わせの回転数とハッシングの効果を評価する。

$$B_s = \frac{\sum_{i=0}^{l-1} | p_i - 0.5 |}{\sqrt{(l-1)}}$$

ただし、lはハッシング結果の長さ[ビット]

表3.2 1バイト中の各ビットで"1"の出現する確率

ビット	7	6	5	4	3	2	1	0
数字	0/10	0/10	10/10	10/10	2/10	4/10	4/10	5/10
英字	0/26	26/26	13/26	11/26	11/26	12/26	13/26	13/26
カタカナ	58/58	32/58	26/58	32/58	32/58	30/58	30/58	29/58

評価関数Bsが0に近いほどハッシング結果が散り、良いふるい落とし効果が得られる。

表3.2に示すモデルを重ね合わせ場合の回転数と評価関数B_sとの関係を図3.2に示す。

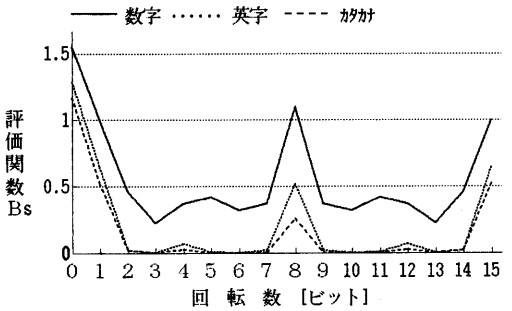


図3.2 文字種と評価値

(c) 考察

回転数0の時が1バイトずつの、回転数8の時が2バイトずつの単純な重ね合わせの場合に対応する。単純な重ね合わせに比べるとハッシング効果は上がっているものの、回転重ね合わせ法でも回転数によりハッシング効果にばらつきができる。

ハッシング前に得られる情報としてはデータベース定義時のデータ属性（文字、固定小数点、10進データ等）のみであり、ハッシング回路を実現するに当ってはデータ属性ごとに回転数を一意に決める必要がある。図のように回転数によりハッシング効果にばらつきが合ったのでは一意に回転数を決めるのは困難である。

ばらつきの原因は次の2つが考えられる。

① ハッシング結果に重ね合わさらない部分ができる。回転数1、15ビットの時がこの場合

である。この時の重ね合わせの様子を図３．３に示す。

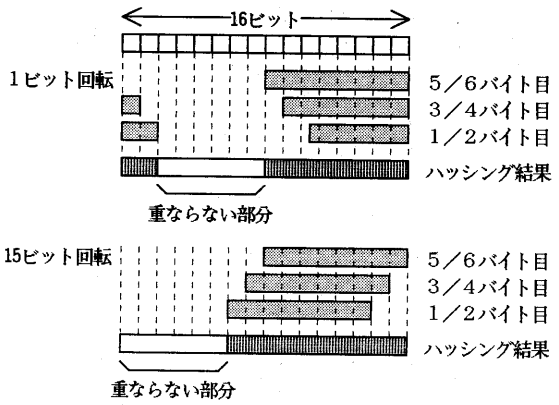


図3.3　回転数1、14ビットのときの重ね合わせの様子

②　”１”の出現確率が０または１に近いビットのみ重なり合う。数字の４～７ビット目、英字の６、７ビット目、カタカナの７ビット目が最も出現確率に偏りのあるビットである。回転数８ビットのときが、これらのビットのみ重なり合う極端な例であるが、他の回転数でもこの原因によりばらつきができると考える。

①の原因はデータ長とハッシング結果の長さが分かれば避けられる。しかし、前提条件によりデータ長の長さはハッシング前には未知のため、重ね合わせの回転数を１あるいはハッシング結果の長さに近いものは避けることである程度解決できる。

②の原因に対し、”１”の出現確率に偏りのあるビット、特に数字データでハッシング効果が悪いので、数字データにおいて確率に極端な偏りのある上位４ビットを削除して、２バイトを１バイトにパックしてからハッシングしてみる。

図３．４にパック化してから回転重ね合わせ法を行った結果を示す。

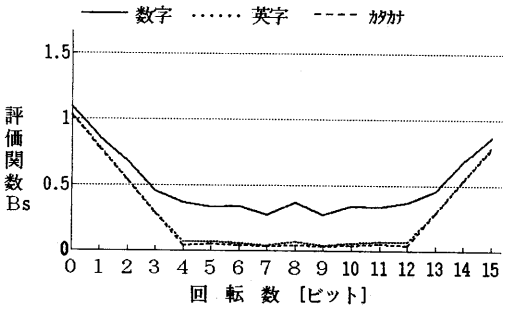


図3.4　パック化した文字種と評価値

図３．４の結果を見ると、回転数が４～12ビットの範囲ではハッシング効果が平準化されていることがわかる。１～３ビット及び13～15ビットでハッシング効果が悪いのは先の①の原因によるものと考える。パック化されているため、３バイトデータとなっており、重ならない回転数の範囲は広がっている。グラフの中央付近が平準化されており、この辺りのどの回転数をとってもほぼハッシング効果が良い。

（３）　ふるい落としシミュレーションとの比較

（２）ではハッシングそのものの評価を行ったが、ここでは実データを用いたふるい落とし処理の評価を行う。

（ａ）　評価関数

解析の際はハッシング結果の各ビットに”１”の出現する確率の偏差を用いたが、ふるい落としの評価では、全データで各ビット位置に”１”の出現する確率を求めるのは困難なため、より簡易で実際的な以下のような衝突率を評価関数として用いることとした。

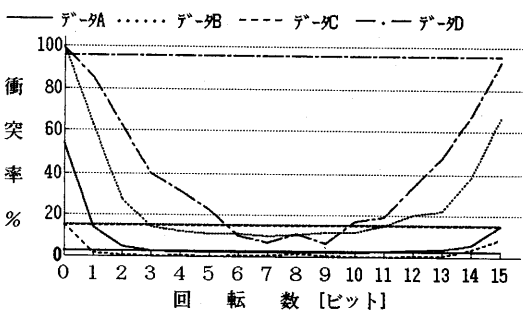
$$衝突率=\frac{残データ数-一致データ数}{参照データ数-一致データ数}$$

（ｂ）　ふるい落とし評価結果

表３．３にシミュレーションに使用したデータの特性を、図３．４にシミュレーション結果を示す。図３．４においてビットアレイの大きさは、

表3.3　ふるい落としシミュレーションのデータ

データ名		属　性	長ｻ[B]	件数	一致数
A	設定	文字(ｶﾀｶﾅ)	2～12	184	184
	参照	[実ﾃﾞｰﾀ]		3386	
B	設定	文字(ｶﾀｶﾅ)	3～12	1564	984
	参照	[実ﾃﾞｰﾀ]		2585	
C	設定	文字(数字)	10	24	6
	参照	[実ﾃﾞｰﾀ]		6672	
D	設定	文字(数字)	6	1729	1697
	参照	[実ﾃﾞｰﾀ]		25318	



＊水平な線は単純重ね合わせ

図3.5　ふるい落としシミュレーション結果

ハッシングの評価時と同じ2^{15}ビットである。図３．５には参考までに２バイト単位の単純重ね合わせの結果も示している。

図３．５から次のことが言える。

① 単純重ね合わせと比べると、回転数の中央付近ではふるい落とし効果が向上している。

② 回転数の中央付近のふるい落とし効果は、図３．４と同様にかなり平準化されており、この付近に回転数を設定することにより、高いふるい落とし効果が望める。

4 メモリ複写機能を持つ２ウェイマージソート法

ソータの機能は前述のとおりであるが、本章では整列機能を実現する整列回路の機能について述べる。

4.1 整列回路の構成に関する基本的な考え方

ハードウェアによる高速な整列用にlogｎ台の比較器を直列に接続して動作させる２ウェイマージソータが各種提案されている。これらは、

① メモリ量の整列データ量に対する割合

② 独立にアクセス可能なバンク数

③ メモリへのアクセス法

などにより構成方式が異なる。

現在のメモリ価格の急速な低下傾向を考えると、整列回路の実現にあたってはメモリ量よりも処理時間、制御の容易性に力点が置かれると思われる。この観点から、２つの独立にアクセス可能なメモリバンクを持った図４．１に示す構成を基本的な構成とする。

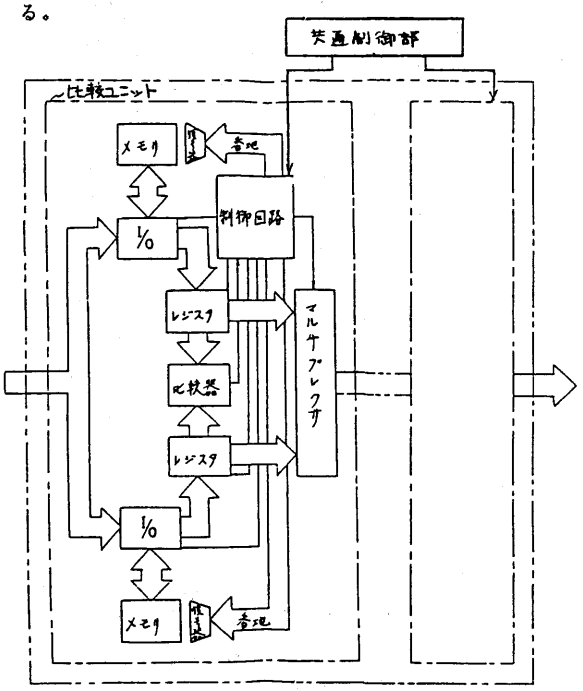


図4.1 整列回路の構成

4.2 メモリ複写機能[10]の動作概要

上記の構成の比較器を複数接続してパイプライン的に動作させる場合に、入力するデータの大小関係によって各段の比較器の空きとなるメモリバンクが変動するので、この空きが判断できるまでメモリへの書き込みを待つ必要が生じる。このため、パイプライン動作が中断するという問題があり、対策が必要となる。

整列回路を構成する格段の比較ユニットは、入力される２つの整列済み部分列を併合して１つの整列済み部分列を作成する処理を繰り返す。比較ユニットに入力される最初の部分列を第１部分列と呼ぶ。この部分列は無条件に比較ユニット内の一方のメモリに格納する。次の入力部分列データ（第２部分列と呼ぶ）は、入力と平行して、メモリに格納済みの第１部分列データと併合処理を行う。この併合処理の途中で次の部分列（第３部分列と呼ぶ）が入力されてくる。この部分列は、次に入力される部分列との併合用のデータなのでメモリに格納しておく必要があるが、第３部分列の入力開始時点ではいずれのメモリが空くか判断できず、判断可能になるまで処理を待つ必要がある。

このように判断できるまで処理を待つのは、処理時間上も無駄である。このため、第３部分列の入力データを２バンクのメモリに同時に書き込むという複写の処理を、どちらか一方のメモリの併合中の部分列が全て出力されるまで、繰り返す。一方の部分列が全て出力されたところで、複写を中止し、まだ出力すべきデータの残っている方のメモリへ書き込まれたデータを破棄する。その後、第３部分列の一方のメモリへのデータ書き込みを継続する。

このような制御を行うことにより、第３部分列のデータを遅れなく２バンクのメモリのうち空きとなるバンク側に書き込むことが可能となる（処理の中断を避けることができる）。複写機能の動作概念を図４．２に、この機能を実現する整列回路の状態遷移図を図４．３に示す。図４．３の各ノードは以下の状態を表す。

［A］ 第１部分列の入力状態
［B］ 第２部分列の第１語の入力状態
［C］ 第２部分列の第２語以後の語の入力状態
［D］ 第３部分列の複写モードでの語の入力状態
［E］ 第３部分列の複写モード以外での入力状態

状態Ｄが複写の機能を実現するモードである。

4.3 処理速度

上記の制御を行った場合の処理速度について、比較に必要なクロック数（メモリへの書き込み／読み出しからなる単位動作を１クロックとした場合）で算出した結果を図４．４に示す。データ複写を行った場合は制御を行わなかった場合に比較して15～30％の処理速度向上が図れることがわかる。

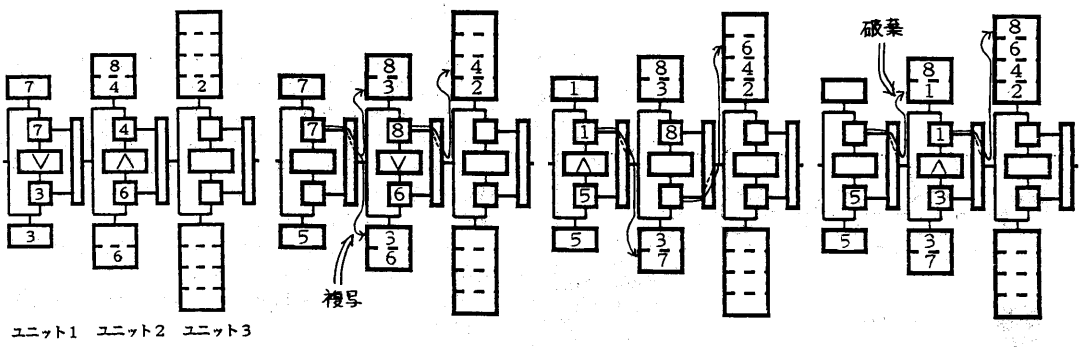


図4.2　データ複写の様子

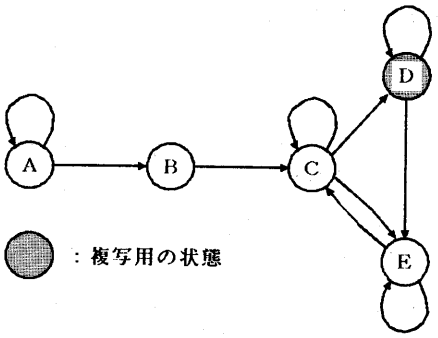


図4.3　状態遷移図

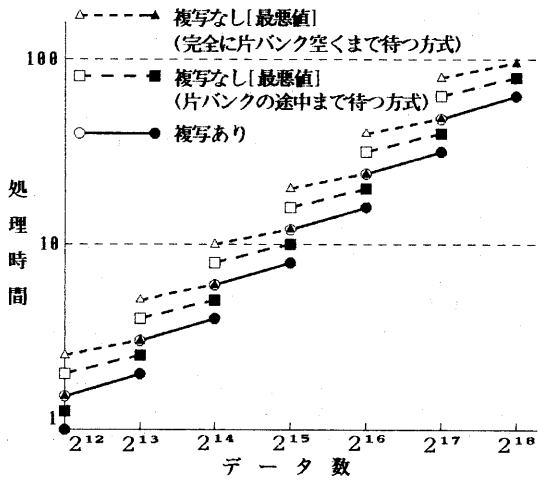


図4.4　整列回路の処理時間

4.4　メモリバスの共通化

（1）　共通化の背景

これまでは、独立アクセス可能な2バンクのメモリを持つ構成を前提に複写機能を持つ整列回路の構成を議論した。前述の動作概要からわかるように、メモリの書き込み動作は、同一のデータを2バンクのメモリの同一の番地へ同時に書き込む（複写する）か、あるいは、どちらかの一方のバンクへデータを書き込むかの2種類である。すなわち、同時に2つのバンクのメモリの異なったアドレス部へアクセスすることは無い。また、読み出し時は比較回路側に1データ分のバッファを用意しておくことにより、メモリへのアクセスを一方のバンクのみに限定することが可能である。

（2）　共通化時の制御法

このような考え方に基づき、メモリバスの共通化を図った場合の整列回路の比較ユニットの構成を図4.5に、状態遷移図を図4.6に示す。また、1語2バイト，最大整列キー長512バイト，整列件数2^{12}～2^{18}件程度の整列が可能な整列回路を実現すると想定した場合のゲート数及びピン数の比を各々図4.7、図4.8に示す。

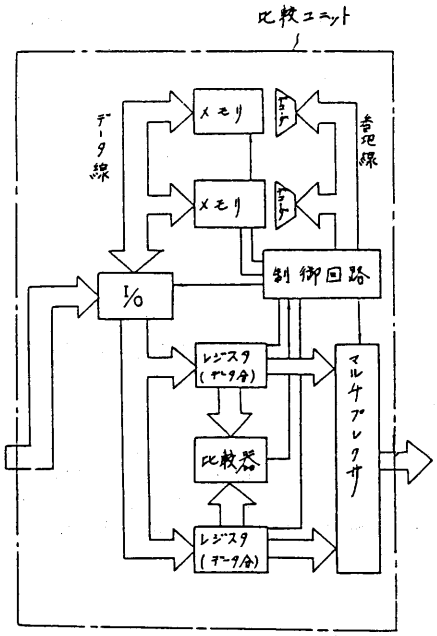


図4.5　バス共通化時の比較ユニットの構成

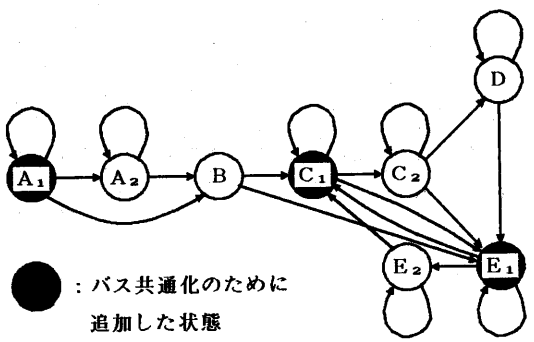


図４．６　バス共通化時の状態遷移図

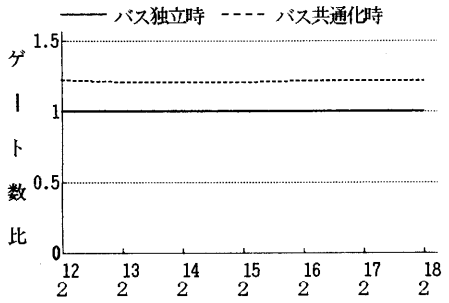


図4.7　整列可能データ数とゲート数

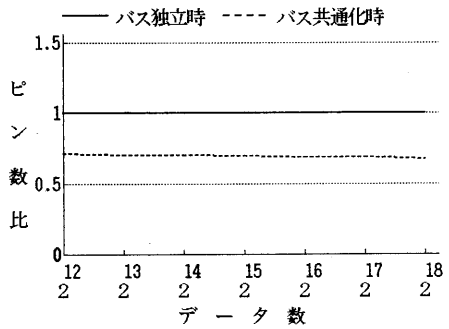


図4.8　整列可能データ数とピン数

図４．７、図４．８からわかるように、ゲート数は図４．３のＡ、Ｃ、Ｅの３つの状態がメモリバスの共通化のために増加するため、約20％程度増加するものの、ピン数は約30％削減可能となる。また、変動分はデータ件数にあまり依存しないことがわかる。

5　おわりに

本稿では３フェーズ・ジョイン方式によって関係演算を実現するために必要なフィルタ、ソータの機能、構成の概要を報告するとともに、これらの専用ハードウェア設計上の要点となる回転重ねあわせ法によるふるい落とし用ハッシング法及びメモリ複写機能を持つパイプライン型２ウェイマージソータの制御法について報告した。

今後は、ハッシング法の実データによる有効性の検証を継続するとともに、制御用ＣＰＵを含むフィルタ、ソータとの連動動作時の動作分析、機能分担の検討をしていく予定である。

最後に、本検討を進めるにあたり討論及び協力を頂いた関係各位に深謝します。

{参考文献}

[1]　角田健男、宮崎収兄他「関係代数演算専用エンジンを備えた関係データベース・マシンＤｅｌｔａ」、日経エレクトロニクス、1985.9.23、pp235～280

[2]　喜連川優、鈴木重信他「ＨａｓｈとＳｏｒｔによる関係代数マシン」、信学技報、ＥＣ81－35、1981

[3]　Yuzuru Tanaka, "MPDC:MASSIVE PARALLEL ARCHITECTURE FOR VERY LARGE DATABASE", Int. Conf. on FGCS,1984.11

[4]　井上潮、北村正他「情報提供サービスに適用可能な超大規模リレーショナル・データベースマシン」、情処データベースシステム研究会資料、47－５、1985

[5]　佐藤清実、北村正他「大規模ＲＤＢ向きデータベース・マシン・アーキテクチャに関する一考察」、情処データベースシステム研究会資料、35－３、1983

[6]　C.D.McGregor,R.G.Thomson et al."HIGH PERFORMANCE HARDWARE FOR DATABESE SYSTEMS",Systems for Large Data Base,North Holland Publishing Co., 1976

[7]　武田英昭、井上潮、中村敏夫「並列ふるい落としアルゴリズムに関する一考察」、情処26回全国大会、1983

[8]　Lum,V.Y.,et al,"Key-to-address transform techniques : a fundamental performance study on large existing formatted files",CACM,Vol.21,No.4, 1971

[9]　Knott,G.D.,"Hashing functions",The Compt.J., Vol.18,No.3,1975

[10]　武田英昭、中村敏夫、北村正「メモリ複写機能を有する２ウェイ・マージ・ソータに関する一考察」、情処32回全国大会、1986