数理解析研究所講究録
423巻 1981年 323-343

サーチエンジンとソートエンジンの記号処理への応用

北海道大学・工学部
田中 譲

1. 序論

データベース処理，特に2つ以上の関係に跨がる関係演算の高速化を目指し，著者等は2種類の機能モジュールを開発した。[1][2] データベース処理においては，大量のデータの転送が隘路となる。この点を考慮して，2種類の機能モジュールを設計するにあたって，これらのモジュールにおけるデータ処理が，モジュールへのデータの転送，およびモジュールからのデータの転送と重畳され，同時実行されるように工夫した。2種類の機能モジュールは，サーチとソートというデータベース処理における2つの基本的な処理を行い，それぞれ，サーチエンジン，ソートエンジンと名付けられている。

サーチエンジンとソートエンジンは，各々のプロトタイプがTTLで組まれて完成している。プロトタイプのサーチエンジンは，16 bitの長さのキィーを4095個，ソートされた順番でテーブルに格納しておき，これに流れ込んでくる任意個の探索キィーの各々をテーブル内で探索し，見つかった場合

にはそのテーブルアドレスを出力し，見つからない場合には，そのキィーよりが大きい最小のキィーが入っているテーブルアドレスを出力する．n個の探索キィーをテーブルアドレスに変換するのにnμsecしかかからない．プロトタイプのソートエンジンは16bitのキィー4095個が1個づつ転送され，最後の1個の転送が終了するやいなや，これらをソートした順序で1個づつ他のモジュールに転送し始めることができる．プロトタイプは，4095個のソートを100msecで終了する．300nsec程度のメモリーを用いて2つのモジュールを構成した場合，両者とも1つのデータあたり1μsec以下の時間で各々の処理を行うことができる．

各々のモジュールは，ビットスライスのアーキテクチュアになっていて，n個の同種のモジュールを横方向に結合することにより，n倍のワード長のデータのサーチやソートを行う回路を構成することができる．両モジュールとも，VLSI化した際のピン数は数10ピン以内に制限することができ，上述の拡張性と共に，これらのモジュールがVLSIモジュールとして通していることを示している．

サーチエンジンとソートエンジンがVLSI化されるとすると，これらの機能であるサーチとソートが基本演算と見做し得るようになり，種々のデータ処理に新しいアルゴリズム

を提供するものと期待する。それは、このような機能モジュールにより、サーチとソートの時間複雑性が変化し、これにより、従来はサーチやソートを用いて処理することなど考えてもみられなかった問題を、2種類のモジュールを用いて処理することにより、今までのアルゴリズムより速く、スマートな処理方法を開発することができるかも知れないからである。

本論文では、2つのモジュールの記号処理への応用を目指し、この分野での基本的な問題の1つであるストリングマッチングを、上述のような問題の1つとして取り上げ、サーチエンジンやソートエンジンがこの種の問題にどのように応用され得るかを示す。本論文で示すストリングマッチングモジュールを核として、PROLOG等の論理型言語の実行を行うマシンを構成することが可能であると思われるが、これについては別の機会に報告する。このストリングマッチングのアルゴリズムでは、パターンをソートエンジンで前処理したものをサーチエンジンに格納し、複数のサーチエンジンで構成されるパイプに、サブジェクトストリングを先頭から順にデータの流れとして流し込むことによって処理する。これに関連して、本論文では可変長データのソートに関しても新しいアルゴリズムを与える。

以下，第2章ではサーチエンジン（以下ではSEEと略記する）とソートエンジン（以下ではSOEと略記する）の機能の概説を行い，第3章では可変長データのソートアルゴリズムを，第4章ではストリングマッチングアルゴリズムを述べる。

2. サーチエンジン（SEE）とソートエンジン(SOE)の機能

SEEは値の順にソートされたn個の被探索キィーからなるテーブル中で，別に与えられるm個の任意の順序に並んだ探索キィーを順々に探索し，値の等しい被探索キィーが見つかった時は，その旨を示すフラグと，値の等しい被探索キィーの内でそのテーブル内アドレスの最小のものアドレスを出力する。値の等しい被探索キィーが見つからない時は，探索キィーより大きい最小の被探索キィーのテーブルアドレスの内，最小のものを出力する（Fig. 2.1）。実際のSEEによる処理は，探索キィーを1個づつ逐次にテーブルアドレスに変換するのでなければ，すべての探索キィーの入力が終了してからテーブルアドレスの出力を開始するのでない。m個の探索キィーは長さmのデータの流れとしてSEEに流れ込み，SEEはこの流れを通す有限長のパイプとして働く。データの流れは，パイプから出るときにはフラグとテーブルアド

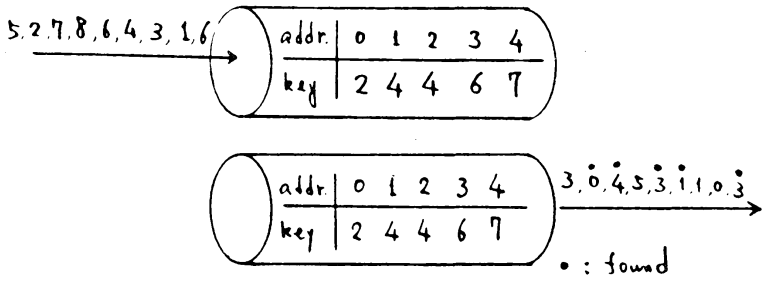


Fig. 2.1. SEEのサーチ機能

レスの流れへと変換されている（Fig. 2.2）。SEEのパイプの長さをSEEのレベル数といい，レベル数LのSEEは 2^L-1 個の被探索キィーからなるテーブルを格納することができる。

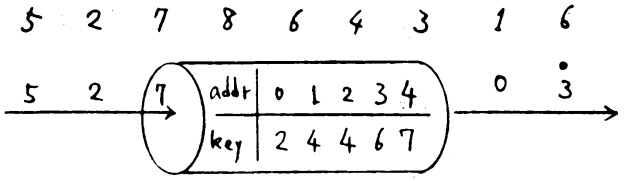


Fig. 2.2. SEEによるデータストリーム処理

フラグがONでテーブルアドレスが i のとき，これを i^* と表わすことにする。$\mathbb{Z}$ を非負の整数の集合とし，$\mathbb{Z}^*$ を

$$\mathbb{Z}^* = \{ i, i^* \mid i \in \mathbb{Z} \},$$

$$\forall i \in \mathbb{Z} \qquad i < i^* < i+1$$

と定義し，本論文ではこの数体系 $\mathbb{Z}^*$ を用いる。$\mathbb{Z}^*$ から $\mathbb{Z}$ への変換は存在し，例えば

$$\alpha : \mathbb{Z}^* \rightarrow \mathbb{Z}$$

$$\alpha(i) = 2i$$

$$\alpha(i^*) = 2i+1$$

とすればよい。以下ではフラグ出力とテーブルアドレス出力をまとめ $\mathbb{Z}^*$ の要素の出力と見做したものを単にテーブルアドレス出力と言うことにする。SEE はテーブルアドレスの流れだけでなく，データとデータの探索キィーの流れが同期して出力するものとし SEE を Fig. 2.3 のように記号化する。テーブ

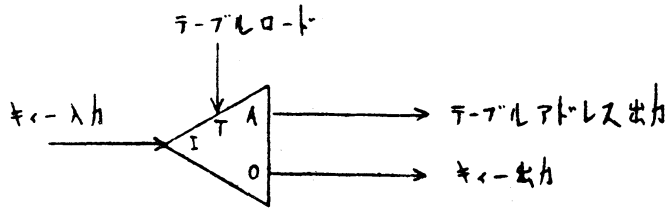


Fig. 2.3 SEE の記号

ル入力は，探索に先立って被探索キィーを入力するための入口である。この入力が，値の順にソートされたデータの流れが流入するのに同期して行われる。

SOEは勝手な順序で並んだn個のキィーからなるデータの流れがSOEに流入され終わると同時に，このn個のキィーを値の順序にソートしたデータの流れを流出し始める（Fig. 2.4）。SOEはすべてのキィーのソートを行うので，キィーのすべてがSOEに流入し終わる前にSOEからの流出が始まることはない。

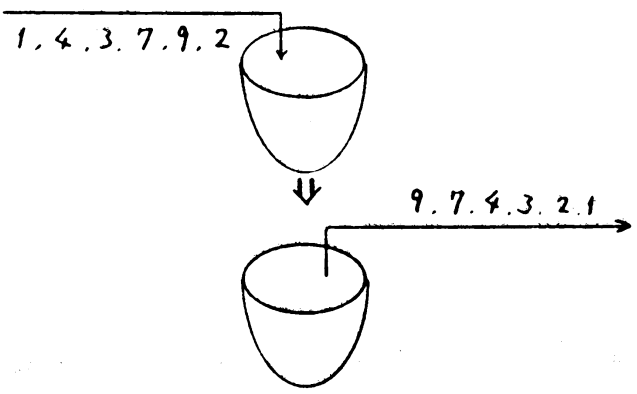


Fig. 2.4 SOEによるソート

SOEがLレベル数が定義されており，レベル数がLのSOEは，単独では最大$2^L - 1$個のデータのソートを行う。SOEはFig. 2.5のように記号化する。

SEEがSOEがビットスライスのアーキテクチュアになっており，多倍長語のデータを処理するには各々を倍数だけ結合すればよい。(1)

サーチにおいて，テーブルが大きくて1つのSEEには格納

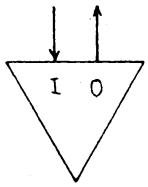


Fig. 2.5　SOE の記号化

できなくなったときは，被探索キィーを順に 2^L-1 個づつ，レベル数 L の別々の SEE に格納し，探索キィーはすべての SEE に流し込めばよい。データ数が多く，1個の SOE ではソートできない場合には，複数個の SOE をトーテムポール接続と名付けた形に接続することにより，使用した SOE の個数倍のデータのソートが可能になる。[1] したがって，1個の SEE，SOE のレベル数を L，語長を w とすると，等価的には，k，k を任意の正整数として，語長が kw で，扱い得るデータ数が単体の場合の 2^L-1 に対して $k(2^L-1)$ 個であるような SEE，SOE を構成することができる。

2^L-1 語のメモリーの各番地 i に，SEE に格納したテーブルの i 番地のキィーに対応するレコードを格納しておき，SEE の出力 j^* に対してこのメモリーの j 番地を読みだすことにすると，このシステムは全体として連想読み出しメモリーとなる（Fig. 2.6）。後置メモリーに，レコード部を上述のような順序でロードするためには，任意個の（キィー，レコ

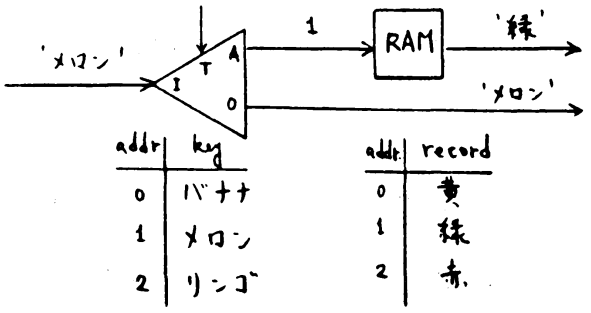
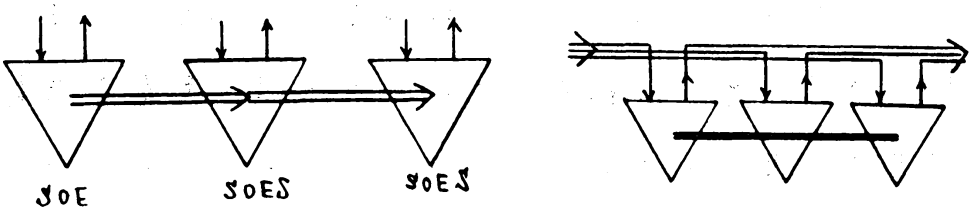


Fig. 2.6 SEEを用いた連想メモリー

ード）の対を，キィーの値の順序に並べ換えなければならない。このような処理は，SOEと，このSOEにレベル数＋1本の信号線で結合されるSOEシミュレータ（SOES）を用い，（キィー，レコード）対の流れのキィー部の流れとレコード部の流れを同期させて別々に各々SOEとSOESに流し込み，全部を流し込んだ後に，SOEからの出力とSOESからの出力を1個づつ対にして出力すればよい。SOE，SOESの結合をFig. 2.7 (a)のように記号で表わす。先述したSOE，SEEの多倍長化のための結合はFig. 2.7 (b)のように表わす。



(a) SOESの記号表現　　(b) SOEの多倍長化

Fig. 2.7 SEE，SOE，SOESの結合

3. 可変長データのソート

Σ を有限順序集合とする。$\perp$ を Σ の最小要素とし，$\Sigma-\{\perp\}$ の要素の有限系列の集合を $(\Sigma-\{\perp\})^*$ とする。$(\Sigma-\{\perp\})^*$ の有限部分集合 S の要素を辞書配列に従ってソートすることを考える。この処理は S の各要素に $\perp$ を適当な個数だけ連接して各系列の長さを一定にして処理を行う(Fig. 3.1)。S を固

H	H ⊥ ⊥ ⊥
A J K L	A J K L
E F	E F ⊥ ⊥
H I J K	H I J K
A B C	A B C ⊥
A B	A B ⊥ ⊥
B	B ⊥ ⊥ ⊥

Fig. 3.1 可変長系列の集合 S と S の要素の固定長化

定長化したものを S_0 で表わすことにする。S_0 のソート法として3種類の方法が考えられる。1つの方法は語長を多倍長化した SOE を用いることである。SOE はビットスライスのアーキテクチュアになっているので充分な個数の SOE を結合することにより，このような多倍長語の SOE を構成できる。他の

2種類の方法では，無制限に多倍長化の機能を用いることなく，多倍長データのソートを行う。

アルゴリズムAはSOEとSOESを用いて，この処理をラディックスソートの考えに基づいて実行する（Fig.3.2）。まずHLLL，AJKL，EFLL，HIJK，ABCL，ABLL，BLLLが最後尾の文字に関してソートされ，例えばEFLL，ABCL，ABLL，HLLL，BLLL，HIJK，AJKLとなる。今度はこれを後から2番目の文字でソートする。この文字が等しい系列a，bは，このときの系列の並びでaがbの前に並んでいればaをbの前に出力するようにする。これをSOEで行うために，もとの系列に順番を0，1，2，…と付け，後から2番目の文字とこの順番の連接をキーとしてソートする。この結果はEFLL，ABLL，HLLL，BLLL，ABCL，HIJK，AJKLの順になる。以下同様のことを後から3番目の文字，4番目の文字，…と繰り返すことによりABLL，ABCL，AJKL，BLLL，EFLL，HLLL，HIJKの順に出力され，ソートが完了する。

アルゴリズムBはSEEとSOEをFig.3.3のように組み合わせてこの処理を実行する。このアルゴリズムでは，テーブルアドレスi，i*を1文字と見做す。図中の1/2は1文字から2文字への変換器で，1文字のデータが2個到着するのを

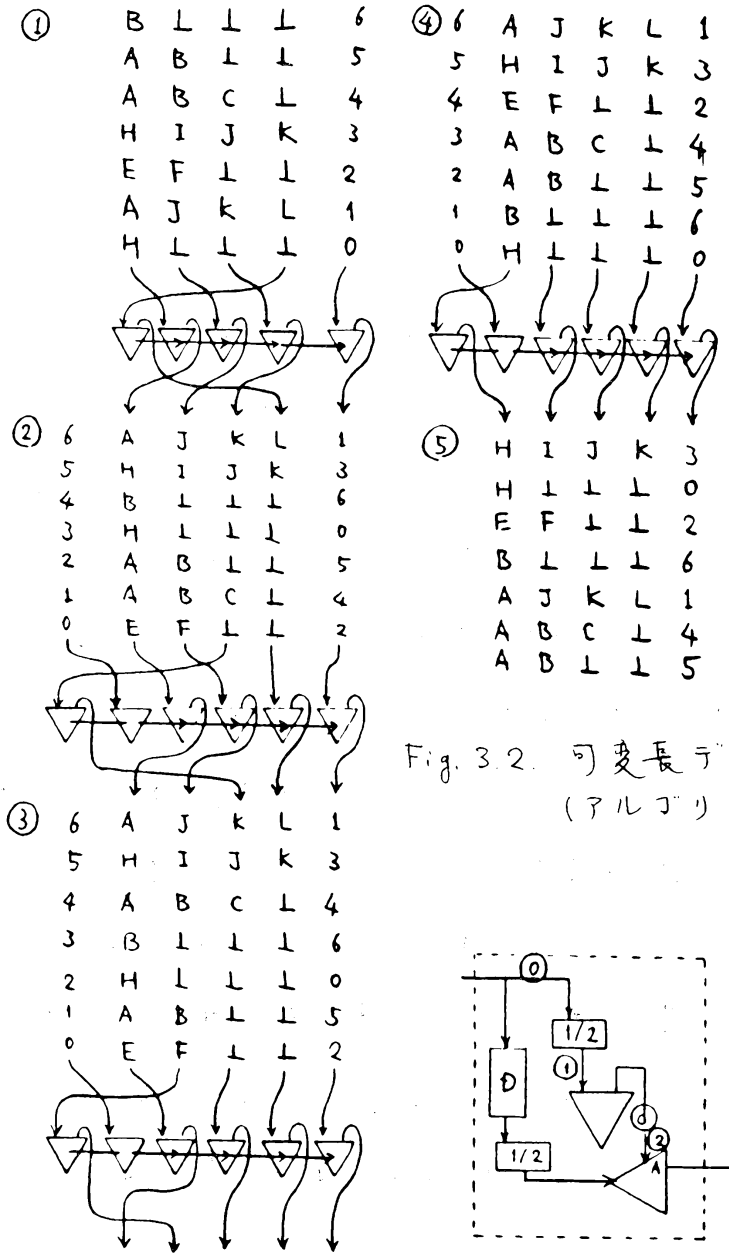


Fig. 3.2 可変長データのソート
(アルゴリズムA)

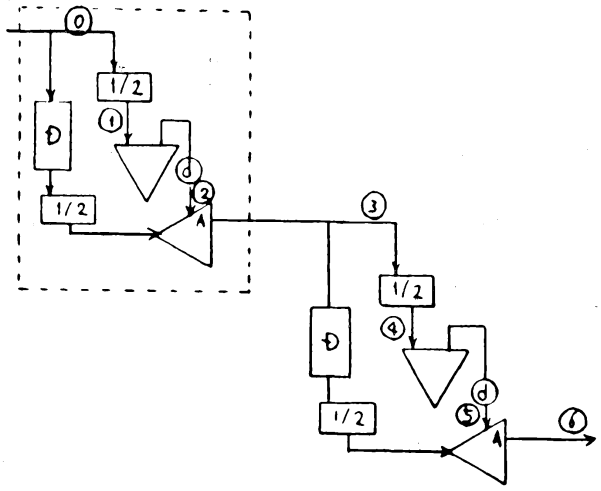


Fig. 3.3 可変長データのソート
(アルゴリズムB)

待って，これらを2文字長のデータ1個に構成して出力する。図中の ⓓ は，重複データの除去器である。図中に ⓓ の置かれている場所では，データはソートされており，従って重複データは連続する。よって重複データの除去は容易である。S_0のデータは各々2^l文字であるとする。そうでないときは，各データに⊥を適当な個数だけ連接すればよい。このとき，S_0のデータのソートを行うには，図中の破線で囲んだ部分をl段用いればよい。物理的に破線で囲んだ部分をl個用意する必要はなく，1個を繰り返し用いることが可能である。

例題のS_0に対し，図中の ⓞ には HL⊥⊥AJKLEF⊥⊥HIJK ABC⊥AB⊥⊥BL⊥⊥ をデータの流れとして入力する。①では これが2文字づつまとめられ

(HL)(⊥⊥)(AJ)(KL)(EF)(⊥⊥)(HI)(JK)(AB)(C⊥)(AB)(⊥⊥)
(B⊥)(⊥⊥)

となる。これをソートし，重複の除去をしたものが②で，これはテーブルとしてSEEに格納される。②とテーブルアドレスは

0	1	2	3	4	5	6	7	8	9
(⊥⊥)	(AB)	(AJ)	(B⊥)	(C⊥)	(EF)	(H⊥)	(HI)	(JK)	(KL)

となる。図中の印は，SEEへのテーブルの格納が終了すると同時にSEEへの探索キィーの入力が開始されるように時間遅れを調整するためのものである．SEEへの入力は①と同じものであり，従ってテーブルアドレス出力である③は，

6* 0* 2* 9* 5* 0* 7* 8* 1* 4* 1* 0* 3* 0*

となる．以下は同様で，④は，

(6* 0*)(2* 9*)(5* 0*)(7* 8*)(1* 4*)(1* 0*)(3* 0*)

となり，⑤とテーブルアドレスは

0 1 2 3 4 5 6

(1* 0*)(1* 4*)(2* 9*)(3* 0*)(5* 0*)(6* 0*)(7* 8*)

となり，⑥は，

5 2 4 6 1 0 3

となる。この並びは，S。をソートすると，HLLLは5+1番目に，次のAJKLは2+1番目に，…，ABLLは0+1番目に，そしてBLLLは3+1番目になることを示している。

4. ストリングマッチング

Σを有限順序集合とし，Σ^*をΣの有限系列の集合とする。Σ^*の有限部分集合Sをパターンの集合，$t \in \Sigma^*$をサブジェクトストリングとするストリングマッチングとは，各整数$i>0$に対してtのi文字目から始まる部分ストリングでSに含まれ，長さのβっとβ長いβのを見つける処理をいう（Fig. 4.1）。

$\Sigma = \{\perp, A, B, C, \cdots, Z\}$

$S = \{C, CB, CBC, CJKL, FD\}$

P1 P2 P3 P4 P5

t ： AFCBCJKLCAD

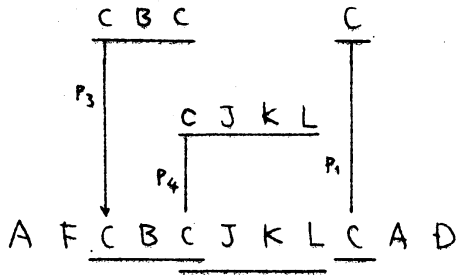


Fig. 4 1　ストリングマッチングの例.

mはSの最長要素長以上とし，Sの各要素αに対して，Σ^*の要素の内で長さがm以下のβのを辞書配列に並べたとき，αの前に並ぶ長さmの系列でβっとβ後に並ぶβのをα^-とし，

先頭部分が a であるような系列の内最後に並んでいるものを a^+ と定義する。a^+ が長さ m の系列となる。$\perp$ を Σ の最小要素とするとき、$a = \perp \cdots \perp$ のように $\perp$ が任意個並んだ系列に対しては、a^- は存在しない。Fig. 4.1 で $m = 4$ とすると、$C^- = BZZZ$, $C^+ = CZZZ$, $CB^- = CAZZ$, $CB^+ = CBZZ$, $CBC^- = CBBZ$, $CBC^+ = CBCZ$, $CJKL^- = CJKK$, $CJKL^+ = CJKL$, $FD^- = FCZZ$, $FD^+ = FDZZ$ となる。S_1 を

$$S_1 = \{ a^-, a^+ \mid a \in S \}$$

とし、S_1 をソートしたものを S_2 とする。Fig. 4.1 の例では、順序集合 S_2 は

$$S_2 = \{ BZZZ, CAZZ, CBBZ, CBCZ, CBZZ, CJKK, CJKL, CZZZ, FCZZ, FDZZ \}$$

となる。

Σ^* の要素の内、長さが m のものを Σ^m で表わす。Σ^m は辞書配列に従う順序集合である。各 $a \in S$ は、a^-, a^+ によって Σ^m を最大3つの区間に分ける。

$u \in \Sigma^m$ に対して、

if $u \leq a^-$ then　u の先頭部分は a と一致しない。

if $a^- < u \leq a^+$ then uの先頭部分がaに一致する.

if $a^+ < u$ then uの先頭部分はaと一致しない.

のような関係がある。S_2の要素は順序集合Σ^mをFig. 4.2のように分割する。

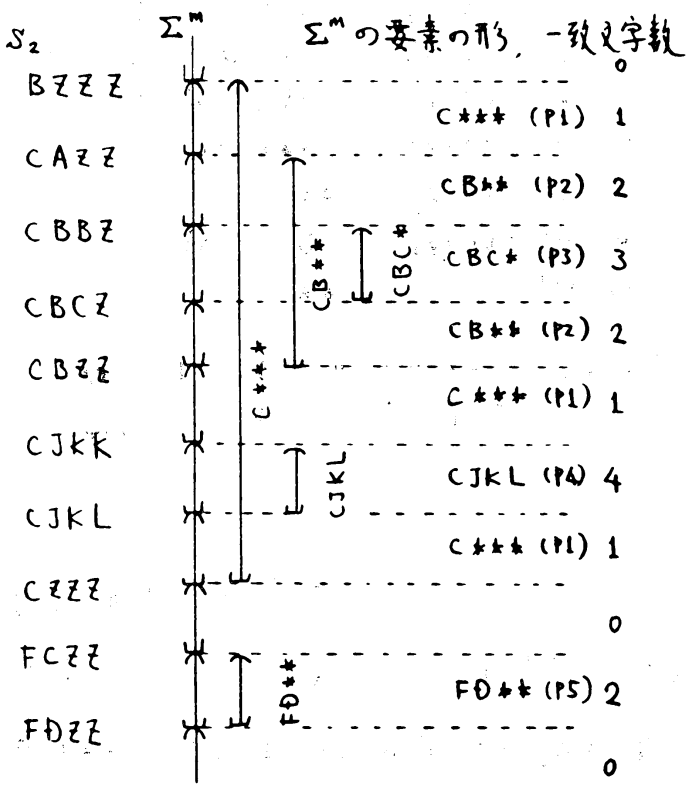


Fig. 4.2 S_2によるΣ^mの分割

後に，4文字表の語表を持つSEEがあり，例のS_2がこのSEEに格納されていて，$u \in \Sigma^*$が入力されたとする。このとき，uと出力されるテーブルアドレスにはFig. 4.3のような関

係がある。

テーブルアドレス	u	一致文字数
0, 0* (Bzzz)	****	0
1, 1* (CAzz)	C*** (P1)	1
2, 2* (CBBz)	CB** (P2)	2
3, 3* (CBCz)	CBC* (P3)	3
4, 4* (CBzz)	CB** (P2)	2
5, 5* (CJkk)	C*** (P1)	1
6, 6* (CJkL)	CJkL (P4)	4
7, 7* (Czzz)	C*** (P1)	1
8, 8* (FCzz)	****	0
9, 9* (FDzz)	FD** (P5)	2
10, 10*	****	0

Fig.4.3 S_2をリサーチした際の出力アドレスとuの関係

m文字表（例ではm=4）のSEEを用いるかわりに，前章のアルゴリズムBと同様に，文字ャテーブルアドレスを2つづつ組にして処理する。例えばBzzzzはBzとzzに分割される。S_2の各要素をこのように分割したときできる2文字データの集合をソートし，重複を除去すると，例題のS_2に対しては，{Bz, CA, CB, CJ, Cz, FC, FD, kk, kL, zz}が得られる。このi番目の2文字データを$(i-1)^*$の値にコード化する。その結果S_2の各要素はFig.4.4 (a)のようにコード化される。この

コードをさらに，Z^*の要素を2個づつに区切り，同様なコード化を行う。このとき，$m=2^l$のストリングマッチングシステムはFig. 4.4のような構成で，SEEをl段連結したものとなる。処理の仕組は図より明らかであろう。

このストリングマッチングシステムは，SEEのレベル数をLとするとき，長さが，

$$\log m \times L + (m-1)$$

のパイプと見做すことができ，サブジェクトストリングは文字の流れとして1文字づつこのパイプに流れ込み，各文字が流出するときにはFig. 4.4の出力部に示すようなストリングマッチングの情報が付加されて流れる。サブジェクトストリングが，このパイプ中を流れるだけで，1度に多くのパターンとの照合がとれる点が，本システムの特長である。(Fig. 4.5)

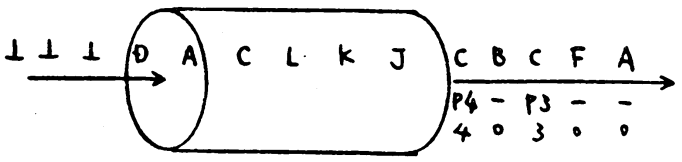


Fig. 4.5 ストリングマッチング
モジュールのストリーム処理

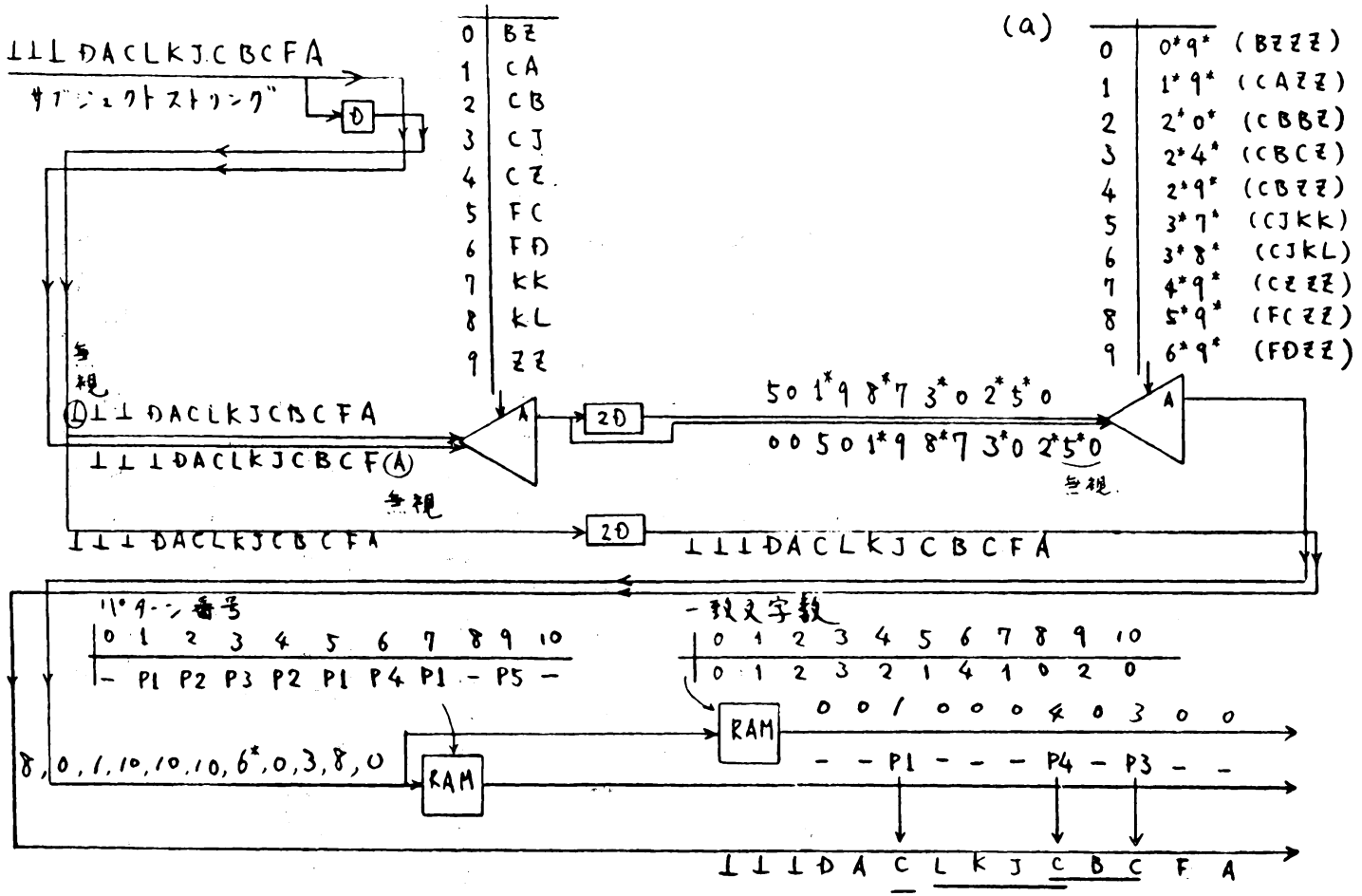


Fig. 4.4 SEEを用いて構成したストリングマッチングモジュール

5. 結論

データベースの高速処理を目指して開発したSEE, SOEが拡張性と実現性の両方において, VLSI化に向いたアーキテクチャであることを述べ, これらがVLSI化されることにより, サーチやソートを基本演算と見做し得るようになった場合に, どのような効果が期待され得るかを, ストリングマッチングを例にとって示した。さらに高度な応用として, 論理型言語の推論機構部への応用等を検討中である.

本論文で提案したストリングマッチングシステムは, Fig. 4.5のような1つの機能モジュールと見做すことができ, これを組み合わせることにより, ワードプロセッシングやドキュメントプロセッシングへの応用が期待される.

参考文献

(1) Y. Tanaka et al., "Pipeline Searching and Sorting Modules as Components of a Data Flow Database Computer," Proc. IFIP Congress 80, 1980 (Tokyo), pp. 427-432.

(2) 田中譲, "データストリーム処理方式のデータベースコンピュータ," 情報処理研資, 記号処理 12-14, 1980, pp 97-103.