計算機アーキテクチャ　90－8
（１９９１．１０．７）

ワークステーション内蔵型フルテキストデータベースプロセッサＳＤＰ

菅野 祐司，　安藤 敦史，　伊藤 正雄，　田村 登，　鶴林 健，　早川 佳宏

松下電器産業㈱　情報通信東京研究所

文字コードで蓄積された大量のテキスト情報の全文検索を高速に実行するフルテキストデータベースの検索プロセッサの構成、扱うデータ形式、機能、処理方式とワークステーションでの利用法を延べる。　本プロセッサはワークステーションの拡張バスに内蔵でき、データフィルタとして動作する。　入力としてホストコンピュータ上の生のテキストデータを受け入れ、テキストの持つ構造を認識して検索・照合の単位を設定、この単位毎に正規表現による文字列照合を行って、条件に合致する単位からなるテキストを出力する。

A WORKSTATION-ATTACHABLE TEXT-RETRIEVAL-PROCESSOR "SDP"

Yuji KANNO, Atsushi ANDO, Masao ITO, Noboru TAMURA,
Ken TSURUBAYASHI, Yoshihiro HAYAKAWA

Tokyo Information and Communications Research Laboratory,
Matsushita Electric Industrial Co., Ltd.

3-10-1, Higashimita, Tama-ku, Kawasaki 214, Japan.

We show our processor "SDP" for document retrieving. SDP searchs whole text after specified character-strings. It can be attached to many workstations which have VME bus and works as filter for text data.

It accepts ordinary text data on computer and recognizes the structure of the text and then divides the text into "record". A record can also be divided into "field". String-matching operation is performed for each field and sequence of matched-records is obtained in the form of ordinary text. We can easily build high performance and easy-to-use document retrieval systems utilizing SDP.

1．はじめに

近年、報道・出版・特許などの産業分野やオフィスでの文書の電子化に伴い、文書のテキスト全文を文字コードで蓄積した「フルテキストデータベース」が普及し[1)]、その構築と利用の技術も開発されつつある[2)]。

筆者らも、「検蔵君」と名付けたフルテキストデータベースの検索システムの開発を行ってきた[3)]。

今回、「検蔵君」の検索専用ハードウエアであるＳＤＰ（Stream Data Processor）の機能を拡張し、かつ中心部をＬＳＩ化して小型化を図り、ワークステーションの拡張スロットに内蔵可能な全文検索プロセッサを開発した。本稿ではこのプロセッサの構成・機能・処理方式及びワークステーションとの結合法について報告する。

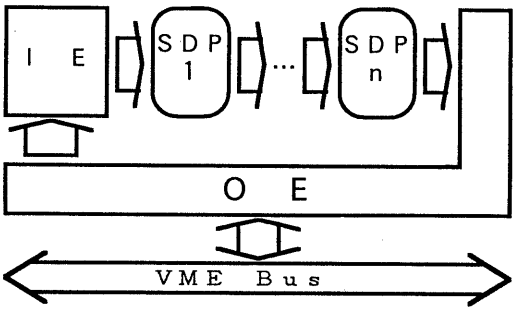
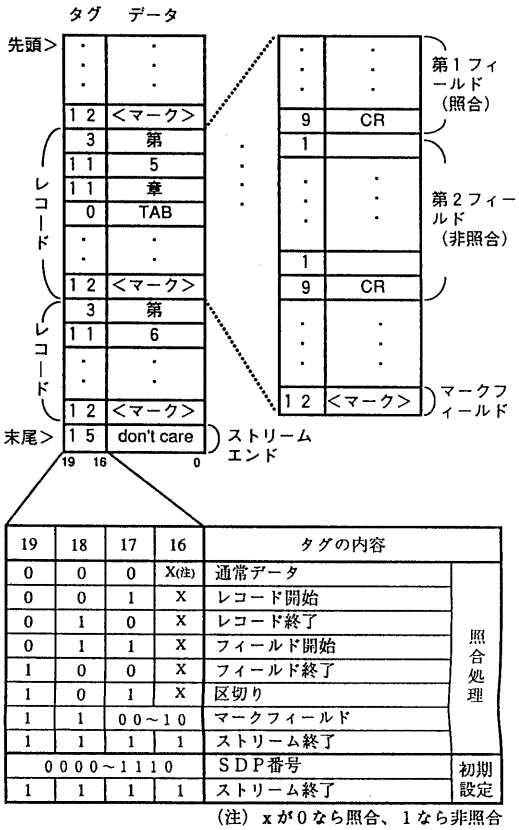


図１．全体構成



19	18	17	16	タグの内容	
0	0	0	X(注)	通常データ	照合処理
0	0	1	X	レコード開始	
0	1	0	X	レコード終了	
0	1	1	X	フィールド開始	
1	0	0	X	フィールド終了	
1	0	1	X	区切り	
1	1	00～10		マークフィールド	
1	1	1	1	ストリーム終了	
0000～1110				ＳＤＰ番号	初期設定
1	1	1	1	ストリーム終了	

（注）ｘが０なら照合、１なら非照合

図２．レコードストリーム形式

2．プロセッサの構成と機能

2．1．全体構成と特徴

プロセッサは図１に示すようにＩＥ，ＳＤＰ，ＯＥの３種のユニットからなる。各ユニットはダブルハイトのＶＭＥモジュールとして実装している。検索時、ＯＥがＶＭＥバスから読み込んだ生のテキストデータはＩＥに送られ、文書構造が調べられて各文字にタグの付いた「レコードストリーム」に変換され、ＳＤＰに送られる。ＳＤＰでは与えられた照合条件に従って文字列照合を行い、条件に合致するレコードを出力する。ＳＤＰを最高１５ユニットまでパイプライン接続して、複雑な照合条件を分割処理することができる。

ＳＤＰからの出力はＯＥに戻され、もとのテキスト形式に変換されてＶＭＥバスへ送出される。プロセッサには全テキストを格納するメモリは持たず、全体としてはテキストに対するデータフィルタとして動作する。検索の前にＳＤＰに与える照合条件も、「初期設定テキスト」を上記と同様に転送することで設定する。

今回の改良による主な特徴は次の３点である。

１）ワークステーションと密結合し、半角・全角文字が混在する生のテキストデータを直接検索できる。

２）被検索テキストの構造（区切り）をハードウエアで認識し、レコード単位の検索、フィールド単位の照合を実現した。

３）検索部のＬＳＩ化で小型化を図り、照合条件が複雑な場合の照合速度の低下を抑える工夫をした。

2．2．レコードストリーム形式と検索処理

ＳＤＰユニットの入出力データ形式である「レコードストリーム」形式は、図２に示すように１６bit の文字データに４bit のタグが付加された２０bit 幅のストリームである。半角（１バイト）文字の場合は上位バイトに格納し、下位バイトは０とする。原テキストは検索処理の単位である「レコード」に分割される。各レコードの末尾には「マークデータ」が１語付加され、検索時の中間結果やエラー情報を格納される。レコードストリーム全体の末尾には「ストリームエンド」が付加され、ＳＤＰやＯＥに処理終了を指示する。ＳＤＰは１レコードを読み込み、条件に合致したレコードのみを出力するという動作をストリームエンドが来るまで繰り返す。１つのレコードの内部は照合処理の単位である「フィールド」に分割され、個々のフィールドとの文字列照合が行われる。マークデータもマークフィールドという特別のフィールドとして照合対象にできる。レコードやフィールドの開始・終了はテキスト中の文字列（句切りパターン）に図のような一連のタグを付与することで、ＳＤＰが認識する。また、「非照合」のタグを与えると、照合処理時に、その文字をないものとみなすため、制御文字等で分離された文字列も照合することができる。ＳＤＰへの初期設定テキストの場合には、タグの意味が図のように変化し、個々のＳＤＰユニットが持つＳＤＰ番号を指定することで、パイプライン接続された複数のＳＤＰに１回のストリームで照合条件を設定できる。検索、初期設定のいずれの場合でも、タグ・マーク・ストリームエンドの付与、半角文字のパッキングはＩＥがハードウエア的に行うため、ホスト側は原テキストを転送するだけでよい。

なお、プロセッサは、マークデータを付けたままのテキストデータ（マーク付きテキスト）の入出力も可能であり、検索結果をマークに累積することができる。

2．3．検索条件

検索条件として、次の３つを指定する。

（１）動作指定

（２）句切り指定

（３）照合条件

動　作	選　択　肢
漢字コードの解釈	1）ＥＵＣコード
	2）シフトＪＩＳコード
エラーレコードの処理	1）照合処理を行わず、一切出力しない
	2）照合処理を行わず、無条件で出力する
アヤシイレコードの処理	1）エラーレコードと解釈する
	2）通常のレコードと解釈する
照合成功時の処理	1）レコードを出力しない
	2）レコードをそのまま出力する
	3）マークデータに演算をしてレコードを出力
照合失敗時の処理	1）レコードを出力しない
	2）レコードをそのまま出力する
	3）マークデータに演算をしてレコードを出力
入力データの種類	1）マークなしの被検索テキストを入力
	2）マーク付きの被検索テキストを入力
	3）ＳＤＰの初期設定テキストを入力
出力データの種類	1）マークなしのテキストを出力
	2）マーク付きのテキストを出力
	3）先頭からのバイトオフセット値を出力
区切りエラーの処理	1）エラーレコードにする
	2）アヤシイレコードにする
カウンタ値のバイト並び	1）LITTLE ENDIAN と解釈する
	2）BIG ENDIAN と解釈する
エラー／アヤシイ検出時の処理	マークデータに指定されたマスク演算を施した結果で、ホストへの通知を制御
入力テキストデータサイズの指定	入力テキストのバイト長を設定する（ストリームエンドパタンと併用可）

図３．主な動作指定

（１）の動作指定は、検索プロセッサ全体の動作モードを指定する。その主なものを図３に示す。図中の「主力データの種類」として「先頭からのバイトオフセット」を選択すると、特定のマークデータを持つレコードについて、本来出力されるべきテキスト中での位置を、４バイトのバイトオフセット値で出力する。この場合には「照合成功・失敗時の処理」は１）あるいは２）を選択する。

（２）の区切り指定は、入力テキストをレコードストリーム形式へ変換する方法を指定する。区切り指定の形式を図４に示す。図中の「区切り指定子」が正規表現の区切りパタンと区切りの種類を指定し、例えば、

ＲＳ／第［０－９］＋章／

と指定することで、図２のように「第５章」という入力テキスト中の文字列に、レコード開始を示す一連のタグがＩＥによって付与される。「クラス」がＲＳ，ＲＳＭ，ＦＳ，ＦＳＳ，ＳＫの場合にはその区切りから始まるレコード／フィールド／読み飛ばしのバイト長を１６bit の「カウント」で指定できる。カウント

【BNFによる定義】　(…は1個以上の繰り返し)

```
〈区切り指定文字列〉 ⇒
  〈区切り指定子列〉…〈区切り指定子列〉

〈区切り指定子列〉 ⇒
  〈空白〉〈区切り指定子〉〈空白〉
 |〈空白〉〈区切り指定子〉〈繰り返し〉〈空白〉                    /* 繰り返し   */
 |〈空白〉|〈区切り指定子列〉…〈区切り指定子列〉|〈空白〉          /* 選言       */
 |〈空白〉|〈区切り指定子列〉…〈区切り指定子列〉|〈繰り返し〉〈空白〉/*選言の繰り返し*/
 |〈空白〉{〈区切り指定子列〉…〈区切り指定子列〉}〈空白〉          /* 並び       */
 |〈空白〉{〈区切り指定子列〉…〈区切り指定子列〉}〈繰り返し〉〈空白〉/*並びの繰り返し*/

〈区切り指定子〉 ⇒ 〈クラス〉〈パターン〉
                 |〈クラス〉〈パターン〉〈カウント〉

〈クラス〉       ⇒
                  RS     /* レコードの開始パターンで、照合対象にしない  */
                 |RSM    /* レコードの開始パターンで、照合対象にする    */
                 |RE     /* レコードの終了パターンで、照合対象にしない  */
                 |REM    /* レコードの終了パターンで、照合対象にする    */
                 |FS     /* フィールドの開始パターンで、照合対象にしない*/
                 |FSM    /* フィールドの開始パターンで、照合対象にする  */
                 |FSS    /* 非照合フィールドの開始パターン              */
                 |FE     /* フィールドの終了パターンで、照合対象にしない*/
                 |FEM    /* フィールドの終了パターンで、照合対象にする  */
                 |FES    /* 非照合フィールドの終了パターン              */
                 |SK     /* 読み飛ばし部分の開始パターン                */
                 |SE     /* ストリームエンドパターン                    */

〈パターン〉     ⇒ /〈正規表現〉/
〈カウント〉     ⇒ C*|C〈数値〉
〈繰り返し〉     ⇒ R*|R〈数値〉
〈数値〉         ⇒ |1以上の数値を表すASCII文字列|
〈空白〉         ⇒ |0個以上の半角空白文字かタブ文字の並び|
```

【区切りパターンによるレコード／フィールドの切出し方】

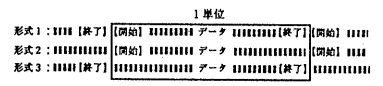


図4．区切り指定の形式

が「C数値」の時は即値データが、「C*」の時は入力テキスト中から読みだした値が、それぞれ使われる。

「区切り指定子列」は「区切り指定子」の繰り返し，並び，並びの繰り返し，選言，選言の繰り返しの5種の組合せで指定する。「繰り返し」は「区切り指定子」を指定回数並べた並びと解釈される。ただし、「R*」は1回以上の繰り返しを表す。「並び」は左から右へ並んだ要素で順に入力テキストを区切ることを表す。「選言」は{}中の要素のいずれかで入力テキストを区切ることを表す。複雑な場合の例を1つ掲げる。

```
RS/a[b-e]*f/C900 {FS/g/ FE/h/R3 [FS/i/ FE/j/]}R*
```

なお、ストリームの終了条件として、SE，SEM（ストリームエンドパタン）と、動作指定の「入力データサイズの指定」を選択・併用することができる。

（3）の照合条件は、レコードストリーム中の各レコードに対する文字列照合の条件を指定するもので、照合結果（成功あるいは失敗）と、（1）の動作指定（照合成功・失敗時の処理）とでSDPユニットの動作が定まる。照合条件の形式を図5に示す。従来と同様[4]、各フィールドに関する条件を「関係式」で表し、その論理式で照合条件を与える。マークフィールドについてはマークデータの特定ビット群の値を、それ以外のフィールドについては正規表現との完全一致，前方・後方・任意一致を、それぞれ調べることができる。また、レコード全体を1つのフィールドと見なす「レコード全体指定」や各フィールドのいずれかと照合すればよいことを表す「全フィールド指定」もできる。

【BNFによる定義】　(…は1個以上の繰り返し)

```
〈照合条件〉 ⇒
    〈照合条件項〉
  |〈照合条件〉OR〈照合条件項〉        /* 選 言   */
  |〈照合条件〉or〈照合条件項〉
  |〈照合条件〉||  〈照合条件項〉

〈照合条件項〉 ⇒
    〈照合条件因子〉
  |〈照合条件項〉AND〈照合条件因子〉 /* 連 言   */
  |〈照合条件項〉and〈照合条件因子〉
  |〈照合条件項〉&&  〈照合条件因子〉

〈照合条件因子〉 ⇒
    〈関係式〉
  |(〈照合条件〉)
  |NOT(〈照合条件〉)                 /* 否 定   */
  |not(〈照合条件〉)
  |  !(〈照合条件〉)

〈関係式〉 ⇒
    $〈フィールド番号〉〈演算子〉/〈照合パターン〉/
  |$A  〈演算子〉/〈照合パターン〉/    /*全フィールド*/
  |$a  〈演算子〉/〈照合パターン〉/
  |$0  〈演算子〉/〈照合パターン〉/    /*レコード全体*/
  |$M  &〈マークマスク〉  ==  〈ビットパターン〉
  |$M  &〈調査ビットパターン〉
  |$M  |〈調査ビットパターン〉

〈演算子〉 ⇒
    ~              /* 包含する     */
  |!~              /* 包含しない   */
  |==              /* 完全一致する */

〈照合パターン〉 ⇒
      〈正規表現〉     /* 任意一致   */
  |^〈正規表現〉       /* 前方一致   */
  |  〈正規表現〉$     /* 後方一致   */
  |^〈正規表現〉$      /* 完全一致   */

〈正規表現〉 ⇒
    〈文字〉
  |.                  /* 任意文字           */
  |〈正規表現〉*       /* 0回以上の繰り返し  */
  |〈正規表現〉+       /* 1回以上の繰り返し  */
  |〈正規表現〉?       /* 省略可能部分       */
  |〈正規表現〉〈正規表現〉   /* 結合         */
  |〈正規表現〉|〈正規表現〉  /* 選択         */
  |[〈文字クラス〉]          /* 文字集合     */
  |(〈正規表現〉)
```

図5．照合条件の形式

例えば、条件

```
(   (($M & 0xff00)==0xa300)
 OR ($0 ~ /１９[５-９][０-９]年/))
AND ($3 !~ /ＳＤＰ|ｓｄｐ/)
```

は、『マークフィールドの上位バイトが 0xa3 であるか、またはレコード全体の中に“１９５０年”～“１９９９年”のいずれかの文字列を含むレコードのうち、第３フィールドに“ＳＤＰ”も“ｓｄｐ”も含まないものと照合せよ』と解釈される。

３．各ユニットの機能と処理方式

３．１．ＩＥ

ＩＥの内部構成を図６に示す。ＩＥは入力テキスト格納用に２組の入力バッファを持ち、これをリングバッファとして使用する。その動作は、図７のように出力していない最初の文字 C(Q) を指すレジスタＱと、現在着目している文字 C(P) を指すレジスタＰの２本のポインタで挟まれた文字列有限状態オートマトンによってＰを１文字ずつ進めながら調べ（探査処理）、文字列がどのパターンかが識別できると、Ｑを進めな

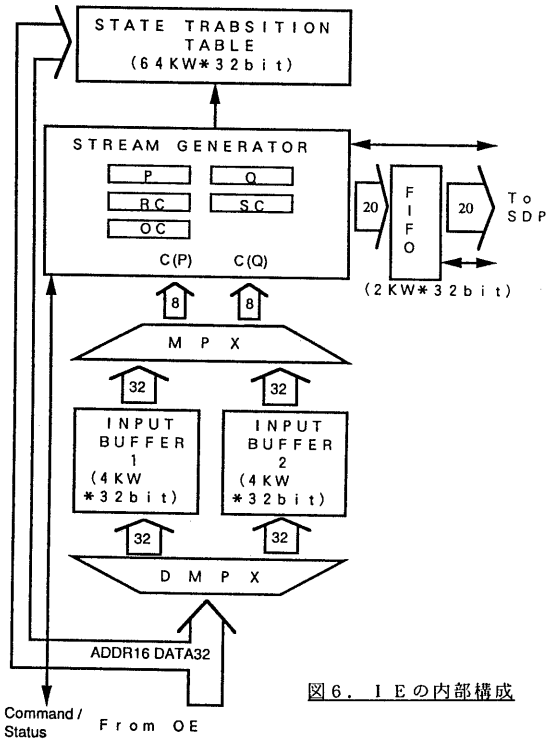


図６．ＩＥの内部構成

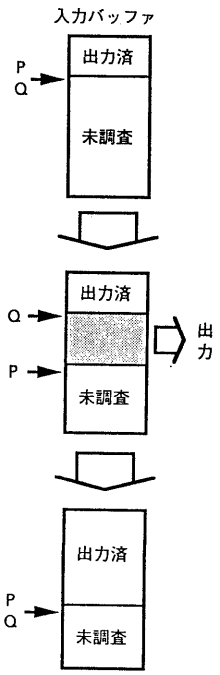
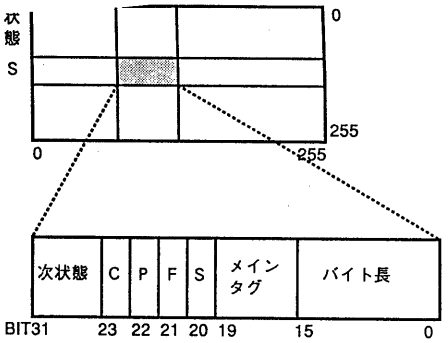


図７．ＩＥの動作原理



【各項目の解釈】

＊メインタグ

0：照合対象データ	7：フィールド開始区切り（非照合）
1：読み飛ばし文字列	8：フィールド終了区切り（照合対象）
2：レコード開始区切り（照合対象）	9：フィールド終了区切り（非照合）
3：レコード開始区切り（非照合）	１１：非照合フィールド開始区切り
4：レコード終了区切り（照合対象）	１３：非照合たフィールド終了区切り
5：レコード終了区切り（非照合）	１５：ストリーム終了区切り
6：フィールド開始区切り（照合対象）	

＊フラグ（ＣＰＦＳ）

Ｃ＝０、Ｐ＝０	：探査続行
Ｃ＝０、Ｐ＝１、Ｆ＝０	：パターンで区切って出力
Ｃ＝０、Ｐ＝１、Ｆ＝１	：パターンで区切って出力、Ｐポインタを動かさない
Ｃ＝１、Ｐ＝０、Ｆ＝０	：入力テキストから得たバイト長で区切って出力
Ｃ＝１、Ｐ＝０、Ｆ＝１	：状態遷移表から得たバイト長で区切って出力
Ｃ＝１、Ｐ＝１、Ｆ＝０	：パターン＋テキスト中のサイズで区切って出力
Ｃ＝１、Ｐ＝１、Ｆ＝１	：パターン＋状態遷移表中のサイズで区切って出力

図８．ＩＥの状態遷移表

がら一連のタグを付けてＳＤＰに出力する（変換処理）、という処理を基本とする。なお、ここでいう「文字」はバイト文字であり、ＩＥは入力テキストをバイト単位で処理する。上記の基本動作に、

・漢字（ＥＵＣ／ＳＪＩＳ），ＡＳＣＩＩ文字，バイナリデータの識別とパッキング
・レコード長カウンタ（ＲＣ），スキップカウンタ（ＳＣ）を用いた、レコード／フィールド／読み飛ばしの各部分の終了の検出
・マーク，ストリームエンドの付与
・出力カウンタ（ＯＣ）を用いた出力レコード長のオーバーフローの検出と例外処理

などが加わって実際の処理が実行される。動作の制御は状態遷移表によりおこなう。その形式と解釈を図８に示す。この３２bit データが状態ｓで文字ｃ(P) を読んだ際の動作を定める。図の「メインタグ」、「バイト長」は、２．３で述べた「クラス」、「カウントの数値」に、それぞれ対応する。また、ＳＤＰの「初期設定テキスト」を変換する場合には、状態遷移表のメインタグがＳＤＰ番号の意味に変化する。

以上の処理の高速化のため、２つのポインタレジスタＰ，Ｑによる文字のアクセスのパイプライン化や、探査処理と変換処理のオーバーラップ化などの工夫をしている。また、図７の原理では、一部の正規表現の場合に区切り方があいまいになったり、実際より長めのパターンを区切りとする事があるため、そのような区切りを含むレコードにはアヤシイマークを付加する。

ＩＥユニットの現在の仕様は以下の通りである。

・入力テキストの変換速度：約１３Ｍバイト／ｓｅｃ
・最大区切りパターン長　：３２Ｋバイト
・状態遷移表の状態数　　：最大２５６状態
・入力バッファの大きさ　：１６Ｋバイト＊２組

３．２．ＳＤＰ

ＳＤＰユニットの内部構成を図９に示す。基本構成は、従来筆者らが開発してきたもの[4]を継承しており、ストリーム入力部（ＳＩＥ），照合処理部（ＰＭＥ），ストリーム出力部（ＳＯＥ）の３ブロックにより構成され、３組のレコードバッファと各要素がクロスバスイッチで接続された、３段のパイプラインをなす。ＳＤＰの動作・機能の詳細は文献５）に譲るが、今回の主な改良点は以下の通りである。

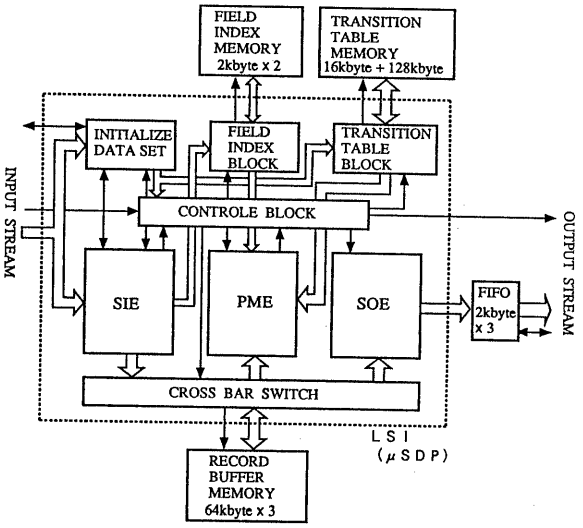


図９．ＳＤＰの内部構成

・中心部（図９の破線部）をＬＳＩ（ゲートアレイ）化して小型化を図った。ピン数の制約のため、入出力ストリームの幅を２０bit とし、レコードバッファも単一のメモリ空間を時分割して使用している。
・状態遷移の高速化のため、文字コードを添字とする遷移先の表を１セット、文字コードを添字とする１bit の「有効ビット表」を４セット搭載した。
・レコードストリームのタグをＩＥが生のテキストから生成しやすいものに変え、タグの不整合のチェック機能を付けた。
・照合処理の機能を充実させ、照合以外の機能の多くは削除した。

中心となるＬＳＩ（μＳＤＰと呼ぶ）に状態遷移表，レコードバッファ，フィールドテーブル，出力ＦＩＦＯの各メモリを接続するだけで、２．２で述べたレコードストリームに対し、２．３の照合条件で検索を行う機能を持つ。現在のＳＤＰユニットの性能を掲げる。

・最大照合速度　　：１６．７Ｍバイト／ｓｅｃ
・最大レコード長　：３２７６６語 ＊ ２０ｂｉｔ
・最大フィールド数：２５６フィールド
・状態数　　：最大２０４８状態

３．３．ＯＥ

ＯＥの内部構成を図１０に示す。ＯＥでは、レコードストリームからタグ，マーク，ストリームエンドを排除し、半角文字のアンパックを行って通常のテキストに戻す。（マーク付き出力の場合はマークも出力）また、タグを監視して出力レコード数をカウントし、エラー・アヤシイレコードの場合には動作指定に従ってホストコンピュータへの通知を行う。出力データ指定が「オフセット出力」の場合（２．３参照）には、マークデータにマスクをかけた値が指定した値のときのみ、そのレコードの先頭文字のバイトオフセットを出力する。

ＯＥはまた、ＶＭＥバスとのＤＭＡによるデータ転送や割り込み制御、ＩＥ，ＳＤＰの動作モードの制御も受け持つ。ＶＭＥバスから見えるモジュールはＯＥのみで、検索プロセッサ全体としては、転送アドレスレジスタ，コマンドステータスレジスタなど８個の３２bit レジスタを持つＤＭＡ入出力デバイスとしてホストコンピュータと結合する。

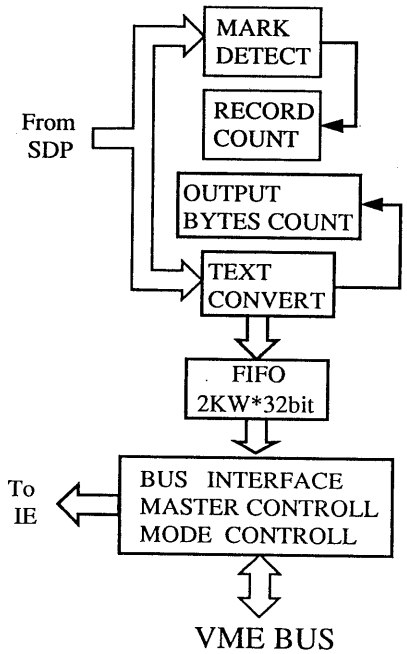


図１０．ＯＥの内部構成

４．ワークステーションでの利用

ＶＭＥバスをＩ／Ｏ拡張バスに持つＳＵＮ，ＳＥＱＵＥＮＴなどのＵＮＩＸワークステーションにこの検索プロセッサを内蔵することにより、ワークステーション上のテキストデータを直接検索するシステムが容易に構築できる。その場合、ワークステーションの持つＣＰＵ、主記憶、補助記憶などの資源が利用でき、システムコストの低減を図ることができる。

筆者らは現在、図１１のようにこの検索プロセッサをＳｏｌｂｏｕｒｎｅワークステーションに内蔵し、新聞記事データを対象にした全文検索システムを構築して評価実験を行っている。このような構成の場合、アプリケーションプログラムからは検索プロセッサがキャラクタ型あるいはストリーム型のデバイスとして扱えるよう、デバイスドライバを書いて、オペレーティングシステムに追加する必要がある。このことにより、多くのＵＮＩＸワークステーションで検索プロセッサを用いた同一のアプリケーションを修正なしで動かすことができる。

また、メモリマップ，ネットワークファイルシステムなど、ＵＮＩＸの持つ先進的なメモリ管理・ファイル管理の機構を利用して、図１２のように、ネットワーク上のテキストファイルの検索を効率的かつ容易に

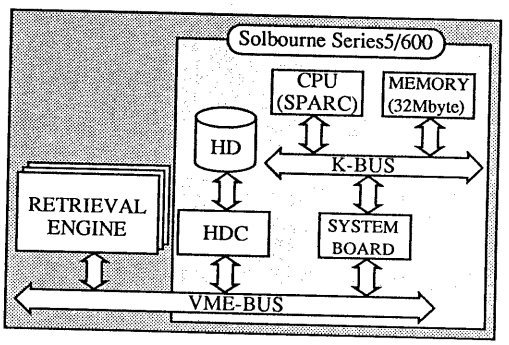


図１１．評価実験システムの構成

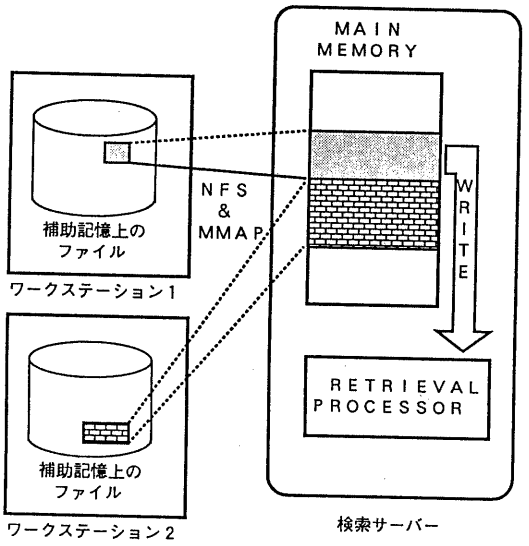


図１２．ネットワーク上のファイルの検索例

行うことができる。

一方このようなホストコンピュータの資源を利用する方法は、検索用の専用メモリを持つ方法と比べて、検索時のデータ転送速度など、効率の悪さが問題になるが、図１１の実験システムの場合には、被検索テキストを格納するのに十分な主記憶がホスト側にあれば、検索プロセッサの性能を十分に引き出せることがわかった。図１１のシステムでの新聞記事データに対する実効データ転送速度は ８～１０ＭＢ／ｓｅｃ であり、この値は他のアプリケーションなどによるホストコンピュータの負荷にはほとんど影響を受けない。

さらに、検索プロセッサのための「検索サーバー」ソフトウエアを作成すれば、マルチユーザ・マルチプロセスでの利用や、大量のデータを複数の「検索サーバー」に分割し、検索サーバー「クラスター」で並列処理を行うことによって、さらなる高速化も期待できる。

５．おわりに

ワークステーション内蔵型のフルテキストデータベースプロセッサについて、その構成，データ形式，機能，ワークステーションでの利用について報告した。

今後は、性能の改善・機能の強化とともに、検索プロセッサの持つ機能・性能を十分に引き出すためのシステムソフトウエアを構築して全文検索システム「検蔵君」の核として活用し、種々の実証規模のテキストデータに対する検索アプリケーションの開発を通して、実用的な全文検索システムの実現を目指していきたい。

【参考文献】

[1]"１９８９年度版データベース白書",(1989)

[2]菅野他："フルテキストデータベースの技術動向"，信学技報，DE90-34，(1991)

[3]伊藤他："フルテキストデータベース検索システム「検蔵君」(1-3)"，第４１回情処全大，(1990)

[4]早川,伊藤他："ストリームデータプロセッサＳＤＰ(1,2)"，第３７回情処全大，(1988)

[5]伊藤他："文書検索システム「検蔵君」(1-2)"，第４３回情処全大，(1991)