形態素抽出ハードウェアアルゴリズムとその実現†

福島俊一††

本稿では，形態素抽出の新しいハードウェアアルゴリズムを提案する．形態素抽出処理は，単語辞書を検索することによって，入力されたテキストに出現したと思われる単語（形態素）をすべて抽出する処理である．自然言語解析に不可欠であり，かつ，時間のかかる処理であることから，高速化が強く望まれていた．提案するハードウェアアルゴリズムは，基本的な1回の照合サイクルをできる限り短縮する方針で設計し，(1)シフトレジスタによるテキストの順送り，(2)先頭文字による辞書範囲の絞り込み制御，(3)単語を構成する全文字の同時照合など，ワイヤードロジックとして容易に実現できる手法を組み合わせたものである．本アルゴリズムを実行する専用ハードウェア（形態素抽出マシン MEX-I）を試作し，8万語の単語辞書を用いた形態素抽出処理を，テキスト1万文字当たり約1秒で実行できるアルゴリズムの高速性を確認した．この処理速度は，パーソナルコンピュータ（CPU：80386，クロック：16 MHz）上のソフトウェアと比較して，100倍～1000倍高速である．さらに，テキストの各文字が M 個の候補をもつ場合の処理時間が，従来のソフトウェアアルゴリズムでは M の累乗オーダになるのに対して，本アルゴリズムは M の線形オーダに抑えられる優位性をもつ．

1. はじめに

近年，自然言語解析技術は，日本語ワードプロセッサや機械翻訳システムなどにおいて実用化されてきている．この自然言語解析技術をより広い応用へ結び付けてゆくためには，より深い解析の追究による高精度化と並行して，処理の高速化を図ってゆく必要がある．

より深い解析のためには，より大規模な辞書・知識が必要になるが，処理速度の低下をまねく．自動通訳・音声認識における言語処理，日本語ワードプロセッサのかな漢字変換，データベース検索の自然言語インタフェースなど，ユーザの入力に対する実時間の応答が要求される応用では，処理の高速性が不可欠である．

一方，大量のテキストを一括処理する応用がある．例えば，一括型の機械翻訳システム，校正支援（ドキュメント検査）システム，文字認識における言語処理（大量ドキュメント入力），テキストのデータベース化・CD-ROM 化のための自動キーワード付けなどである．このような応用では，解析処理が高速になるほど人手と比較したときの所要時間が短縮され，メリットが大きくなる．

自然言語解析処理の高速化は，これまで主に，逐次型の汎用計算機の発展と，その上でのソフトウェア/アルゴリズムの改良に支えられてきた．しかし，高速化の要求はそれ以上に強く，近年は汎用並列計算機が期待されている．そして，並列論理型プログラミング言語で記述された構文解析プログラム[1]や，P-RAM (Parallel Random Access Machine) を想定した並列構文解析アルゴリズム[2)~4)]，マルチプロセッサ型の形態素抽出システム[5]などが考案されている．しかし，このアプローチで十分な性能が得られるようになるには，今後，共有メモリに対するアクセス競合や，プロセス間通信のボトルネック問題などが克服されてゆく必要がある．

これに対して，筆者らは，ハードウェアアプローチを提案し，専用ハードウェア（文章解析アクセラレータ：Natural Language Parsing Accelerators）の開発を進めている[6)~8),22)]．

他分野では既にハードウェア化の試みがあるが，自然言語解析（特に日本語解析）では，次のような特徴から新しいハードウェアアーキテクチャが必要になる．

(1) 数値計算に比較して文字列処理の占める比率が圧倒的に大きい．しかも日本語では，文字の種類が極めて多い（数千種類）．

(2) 語彙や規則を収めた辞書へのアクセス頻度が高い．しかも，辞書が大規模である（数万件から数十万件）．

文字列処理ハードウェア[9),10)]，シストリックアレイを用いて CYK 法や Earley 法を並列化した文脈自由言語認識方式[11),12)]，音節の種類だけのアドレス空間をもつ RAM を用いた音声認識用の辞書照合方式[13)]などは，自然言語解析へ一部利用できそうなハードウェアアーキテクチャをもつが，(1)，(2)が性能の妨

† A Morpheme Extraction Hardware Algorithm and Its Implementation by TOSHIKAZU FUKUSHIMA (Terminal Systems Research Laboratory, C & C Systems Research Laboratories, NEC Corporation).

†† 日本電気(株) C & C システム研究所ターミナルシステム研究部

げとなっている．

筆者は，(1)，(2)の特徴を考慮した自然言語解析向き専用ハードウェアの開発という新しい試みにおいて，まず形態素抽出処理のハードウェア化を行い，形態素抽出マシン MEX (Morpheme EXtraction machine) を試作した[6),7)]．

本稿では，まず第2章で，形態素抽出処理とそのハードウェア化の必要性を述べ，第3章で，ハードウェア向きの新しい形態素抽出アルゴリズムを提案する．第4章では，アルゴリズムの時間計算量などの分析を行い，第5章では，アルゴリズムの高速性を検証するために試作した形態素抽出マシン MEX-I の概要と性能評価結果を示す．さらに第6章では，関連研究との比較により，アルゴリズムの特長を明確にする．

2. 形態素抽出処理のハードウェア化

日本語文では，入力された文章を単語列（形態素列）に分割し各単語の品詞を認定する形態素解析処理は，通常，次のような3過程に分けることができる[14)]（**図1**参照）．

(1) 形態素抽出：単語辞書を検索し，入力テキスト中に出現したと思われる単語（形態素）をすべて抽出する．

(2) 接続検定：接続表（二単語間の文法的な接続可否条件を記述したもの）を参照して，(1)で抽出された単語のうち接続可能なものを連結する．

(3) 候補選択：構文・意味などの情報やヒューリスティックスを用いて，(2)で連結された単語の組み合わせのなかから，最も確からしいものを選択する．

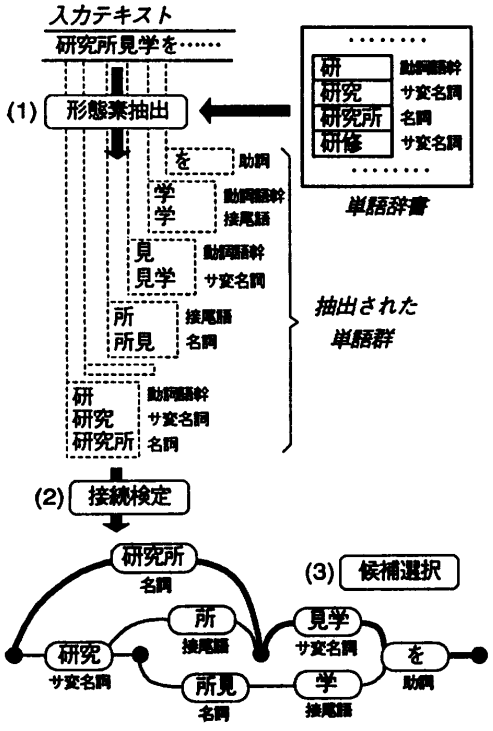


図1 形態素解析処理の流れ
Fig. 1 Morphological analysis process for Japanese text.

自然言語解析において，(1)のような辞書アクセス処理は最も基本であり，かつ，時間のかかる部分であることから，高速化が強く望まれている．また，近年，並列解析方式の研究が盛んになってきたが[1)~5)]，前述のように，実際にインプリメントした場合には，共有メモリに置かれる辞書のアクセス時間が性能のボトルネックになることが予想される．この問題を軽減するためにも，形態素抽出処理のような辞書アクセス処理の高速化は重要である．

さらに，処理の大半が単純な照合処理の繰り返しであり，ソフトウェアとしてはアルゴリズムがほぼ確立されている点が，ハードウェア化に適している．

以上の点から，高速化が望まれ，かつ，ハードウェア化に適した処理単位として，まず形態素抽出処理を取り上げ，そのハードウェア化を行った．

3. ハードウェアアルゴリズム

従来の形態素抽出処理（ソフトウェア）では，単語辞書から該当する単語を検索するために，二分探索法・ハッシュ法・桁探索法などの探索アルゴリズム[15),16)]を用いている．

これに対して，本稿では，以下に示す新しいハードウェアアルゴリズムを提案する．

3.1 基本的な構成・動作

形態素抽出ハードウェアアルゴリズムを実行するための基本的な構成は，**図2**のとおりである．おもな構成要素は，単語辞書，シフトレジスタ，インデックス，アドレス生成器，比較器である．ここで N は，単語辞書内の単語はいずれも長さが N 文字以下となるように定める．

単語辞書は，N 文字幅の領域に1単語ずつを割り当てて格納する．さらに，各単語に対応する N 文字が同時に読み出せるように，1文字目辞書～N 文字目辞書の N 個に分割しておく（**図3**参照）．なお，長さが N より短い単語については，N に満たない部分

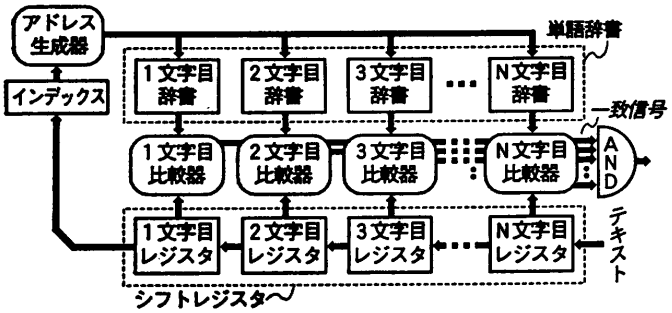


図 2 基本アルゴリズムの構成
Fig. 2 Fundamental architecture of hardware algorithm.

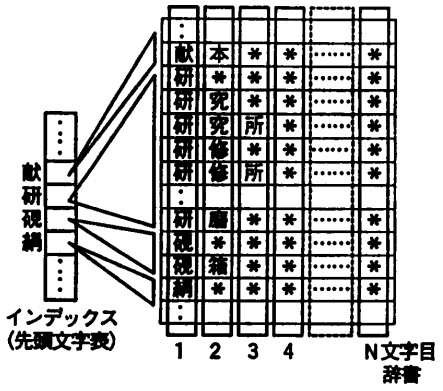


図 3 単語辞書の構成とインデックスとの関係
Fig. 3 Relation between word dictionary memories and index memory.

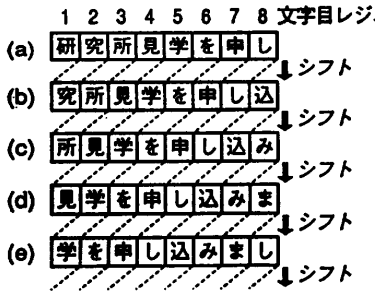


図 4 シフトレジスタの動作 ($N=8$ のときの例)
Fig. 4 Movement in shift registers.

に，あらかじめ定めた残余記号を格納しておく（図3では「*」が残余記号を表している）．

シフトレジスタは，1文字目レジスタ～N 文字目レジスタから成り，テキストの N 文字分を格納し，順送りする（図4参照）．

インデックス（先頭文字表）は，ある文字に対して，その文字で始まる単語群の，単語辞書における存在範囲を示す（図3参照）．

なお，単語辞書では，先頭文字が同一の単語群は，連続するアドレス範囲に格納しておく．

アドレス生成器は，1文字目辞書～N 文字目辞書に対して同一のアドレスを与え，かつ，そのアドレスを同時に変化させる．変化させるアドレスの範囲は，シフトレジスタの1文字目と同じ文字で始まる単語群の存在範囲であり，インデックスから得る．

比較器は，単語辞書とシフトレジスタの各文字位置を対応付けるように，全部で N 個設ける．n 文字目 ($1 \leqq n \leqq N$) に対応する比較器は，n 文字目辞書から読み出された文字が，シフトレジスタの n 文字目か残余記号のどちらかに一致したときに，一致信号を出力する（残余記号はワイルドカードの働きをする）．単語辞書内の1単語に対応する N 文字，および，シフトレジスタ内の N 文字は同時に読み出され，上記の照合処理は N 個の比較器で同時に実行する．全比較器から一致信号が出力されたときに，単語（形態素）が1個検出されたことになる（図5参照）．

さて，上記のような構成にもとづく形態素抽出のハードウェアアルゴリズム（基本アルゴリズム）は，以下のとおりである．

□基本アルゴリズム

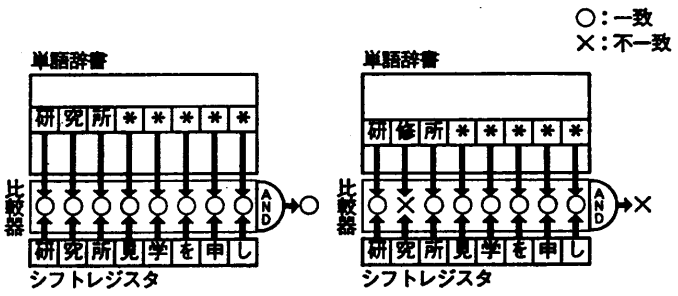
■メインプロシジャ：

【ステップ1】 シフトレジスタに入力テキストの先頭の N 文字をロードする．

【ステップ2】 1文字目レジスタにテキスト末尾マークが到達するまで，プロシジャ1を繰り返す．

■プロシジャ1：

【ステップ1】 1文字目レジスタと同じ文字で始



(a)「研究所」を検出 (b)「研修所」は検出せず
(a) "研究所" is detected. (b) "研修所" is NOT detected.

図 5 同時照合による単語の検出 ($N=8$ のときの例)
Fig. 5 Simultaneous comparison in fundamental architecture.

まる単語群の単語辞書内アドレス範囲を，インデックスより得て，その先頭アドレスを単語辞書のカレントアドレスとする．

【ステップ2】 カレントアドレスが上記のアドレス範囲内にある間は，プロシジャ2を繰り返す．

【ステップ3】 シフトレジスタ内の文字列を，1文字分順送りする．

■プロシジャ2：

【ステップ1】 カレントアドレスにおいて単語辞書から読み出された N 文字と，シフトレジスタ内の N 文字とを，N 個の比較器で同時に照合し，単語の検出を判定する．

【ステップ2】 カレントアドレスをインクリメントする．

(基本アルゴリズム終わり)□

以下では，上記の基本アルゴリズムと図を対応付けて，動作例を簡単に説明する．

「研究所見学を申し込みました……」という入力テキストに対して，メインプロシジャのステップ1の直後に，シフトレジスタの内容は図4の(a)のようになる．プロシジャ1のステップ3ごとに，シフトレジスタ内の文字列は，(a)→(b)，(b)→(c)，(c)→(d)，(d)→(e)，……と移動する．プロシジャ1は，図4の(a)(b)(c)(d)(e)……の各々に対して実行される．

図4の(a)の状態では，シフトレジスタの1文字目が「研」である．そこで，プロシジャ1のステップ1では，単語辞書における「研」で始まる単語群のアドレス範囲をインデックスより得る．図3の単語辞書では，「研」「研究」「研究所」「研修」「研修所」……「研磨」が，そのアドレス範囲に該当する．そして，そのアドレス範囲内の各単語について，プロシジャ2のステップ1の照合処理が順次実行される．

3.2 複数候補付きテキストへの拡張

本節では，前節に示した基本アルゴリズムを，テキストの各文字が複数個（$=M$ 個とする）の候補をもつ場合に拡張する（$M=1$ の場合が基本アルゴリズムに相当する）．複数候補付きテキストは，文字認識や音声認識の結果などにおいて発生する[13),17)]（図6に，複数候補付きテキストと，それに対する形態素抽出結果の例を示す）．

図7は拡張アルゴリズムを実行するための構成である．この構成（図7）は，基本的な構成（図2）に対して，次の3点の修正を加えたものである．

第一は，シフトレジスタを M 個設けた点である．M 個のシフトレジスタは同期して，テキストの同一位置の M 個の候補を同時に順送りする．

第二は，n 文字目に対応する比較器が，n 文字目辞書から読み出された文字と，M 個のシフトレジスタの M 種類の n 文字目とを照合する点である．その比較器は，n 文字目辞書から読み出された文字が，残余記号，または，M 個のシフトレジスタのいずれかの n 文字目に一致したときに，一致信号を出力する（図8参照）．このとき M 個の文字の処理は同時に実行され

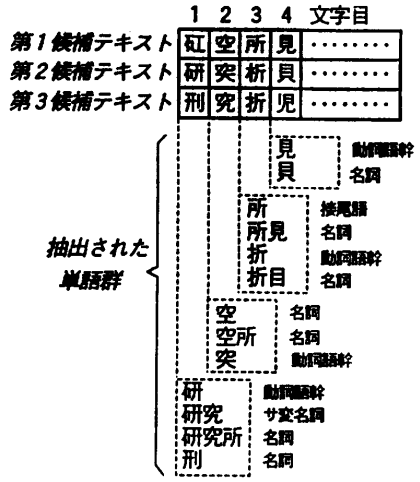


図6 複数候補付きテキストと形態素抽出結果（$M=3$ のときの例）

Fig. 6 Text with multiple candidates and morpheme extraction results.

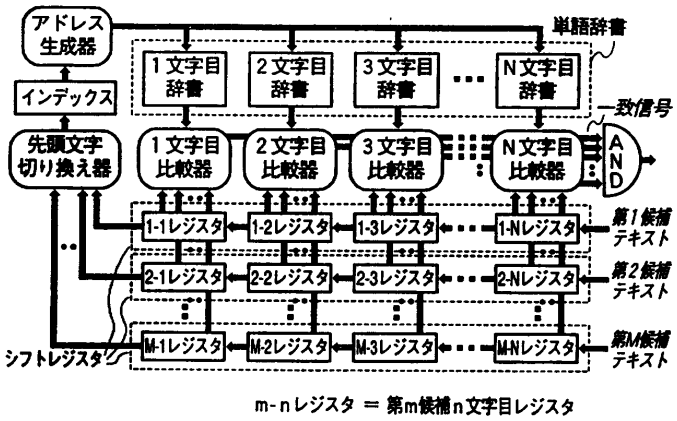


図7 拡張アルゴリズムの構成

Fig. 7 Extended architecture of hardware algorithm.

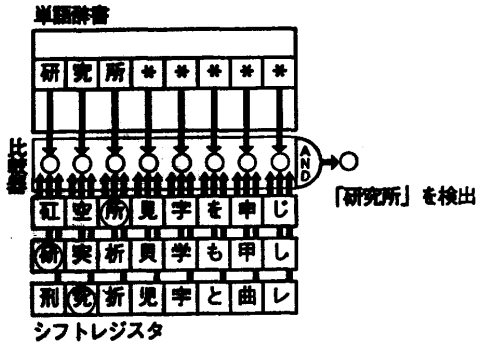


図 8 複数候補付きテキストでの同時照合
($N=8$, $M=3$ のときの例)
Fig. 8 Simultaneous comparison in extended architecture.

るので，比較器の処理時間は候補数 M に依らない．

第三は，先頭文字切り換え器を設けた点である．先頭文字切り換え器は，M 個のシフトレジスタの1文字目を，順番に切り換えてインデックスへ渡す．

このような構成を用いた拡張アルゴリズムは，前節に示した基本アルゴリズムに対して，プロシジャ1を次のようなプロシジャに置き換えればよい．

□拡張アルゴリズムへの置換部分

■プロシジャ1：

【ステップ1】 先頭文字切り換え器の指すカレントレベルを第1候補とする．

【ステップ2】 カレントレベルが M 以下である間，プロシジャ 1.5 を繰り返す．

【ステップ3】 シフトレジスタ内の文字列を，1文字分順送りする．

■プロシジャ 1.5:

【ステップ1】 カレントレベルのシフトレジスタの1文字目と同じ文字で始まる単語群の単語辞書内アドレス範囲を，インデックスより得て，その先頭アドレスを単語辞書のカレントアドレスとする．

【ステップ2】 カレントアドレスが上記のアドレス範囲内にある間は，プロシジャ2を繰り返す．

【ステップ3】 カレントレベルをインクリメントする．

(置換部分終わり)□

3.3 冗長性とユーザ登録機能

第3.1節および第3.2節で述べたアルゴリズム（基本アルゴリズムおよび拡張アルゴリズム）では，1文字目辞書と1文字目比較器は冗長であり，省略可能である．インデックスによる辞書範囲の絞り込みによって，1文字目比較器で一致が発生するのは明らかなためである．

一方，ユーザ登録を想定して，単語の追加登録領域を設けることが可能である．単語があらかじめ登録されインデックスで管理されたアドレス範囲の後に，追加登録のためのアドレス範囲を設けて，追加登録単語を順次書き込むようにする．そして，形態素抽出の際には，シフトレジスタの先頭文字をもとにインデックスから得られるアドレス範囲の単語群に加えて，追加登録アドレス範囲の単語群も照合対象として読み出す．すなわち，基本アルゴリズム（第3.1節）や拡張アルゴリズム（第3.2節）のプロシジャ1において，ステップ3の直前に，次のようなステップ2.5を加える．

□ユーザ登録のためのアルゴリズム追加部分

【ステップ 2.5】 カレントアドレスを追加登録アドレス範囲の先頭に位置付け，カレントアドレスが追加登録アドレス範囲にある間は，プロシジャ2を繰り返す．

(アルゴリズム追加部分終わり)□

この追加登録領域には，単語をソートして登録する必要はない（あらかじめ登録されインデックスで管理された単語の領域とは異なり，同一の先頭文字の単語群をまとめておく必要もない）．その代わり，追加登録領域の単語については，1文字目比較器で常に一致が得られるという保証はなくなる．この場合には，1文字目辞書と1文字目比較器は省略できない．

4. アルゴリズムの特性

前章で提案したハードウェアアルゴリズムでは，各文字が M 個の候補をもつ長さ L のテキストに対する，語数 D の単語辞書を用いた形態素抽出処理の時間計算量が $O(L \cdot M \cdot D)$ となる（その証明は付録参照）．また，第3.1節で定義した N に対して，単語辞書の容量は $N \cdot D$ である．

4.1 文字候補数と処理時間

従来ソフトウェアで実現されている形態素抽出処理では，テキストから切り出した部分文字列に一致する単語を，単語辞書から検索する処理を繰り返す．テキストの各文字が複数（$=M$ 個）の候補をもつ場合には，部分文字列の種類がその組み合わせの数（$=M$ の累乗個）に応じて増える．したがって，時間計算量は M の累乗オーダとなる．

これに対して，提案したハードウェアアルゴリズム

では，時間計算量が M の線形オーダに抑えられる優位性をもつ．

4.2 辞書語数と処理時間

従来の辞書探索アルゴリズムでは，辞書の語数 D に対して時間計算量が $O(\log D)$ となるものが多い[15),16)]☆．

前章のハードウェアアルゴリズムは，基本的な1回の照合サイクルをできる限り短縮する方針で設計した．その設計方針にしたがい，（1）シフトレジスタによるテキストの順送り，（2）先頭文字による辞書範囲の絞り込み制御，（3）単語を構成する全文字の同時照合など，ワイヤードロジックとして容易に実現できる手法を組み合わせている．

したがって，D に関するオーダで比較するとソフトウェアアルゴリズムが優っているが，本ハードウェアアルゴリズムは，ソフトウェアに比べて D に関する係数の極めて小さいハードウェアを実現できる優位性をもつ．すなわち，辞書の語数 D が現実的なサイズであれば，ソフトウェアアルゴリズムに比べて十分高速な実行速度が得られる．

この優位性については，第5章で検証を行う．

5. 形態素抽出マシン MEX-I

提案するハードウェアアルゴリズムに関し，第 4.2 節で指摘した高速性を検証するために，形態素抽出マシン MEX-I を試作した．MEX-I は，第 3.2 節の拡張アルゴリズムをワイヤードロジックによって実現した専用ハードウェアである．

5.1 MEX-I の概要

MEX-I は，NEC パーソナルコンピュータ PC-9800 シリーズのバックエンドマシンとして動作する．MEX-I は，ホストパーソナルコンピュータからテキストを受け取り，そのテキストに対する形態素抽出結果の単語群を返す．テキストは，16 ビット系の漢字かな混じり文字列であり，形態素抽出結果の単語群は，各単語のテキスト内位置（先頭位置と単語長）と辞書内アドレスの組のリストの形式とした．

☆ ハッシュ法では，ハッシュ表の占有率が十分小さければ，$O(1)$ の時間計算量が得られる．しかし，現実的な辞書容量の制約のもとでは，占有率は大きくなり，時間計算量は $O(D)$ に近づく．また，桁探索法の一種である trie 法でも $O(1)$ の時間計算量が得られる．しかし，本来の trie 法は，各ノードが常に全文字種分のサイズをもつので，日本語では膨大な辞書容量が必要になり，現実的でない．したがって，桁探索法では通常，trie の冗長性を除去した木構造辞書が採用されており，その時間計算量は $O(\log D)$ である．それ以外によく用いられる二分探索法，平衡木法なども $O(\log D)$ である．

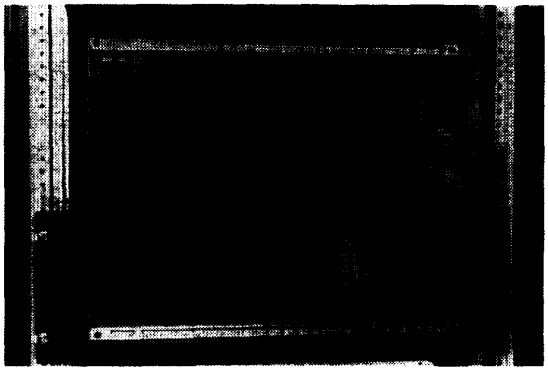


図 9 MEX-I の外観
Fig. 9 Overall view of MEX-I.

MEX-I は，図7の構成要素（単語辞書メモリ，シフトレジスタ，インデックスメモリ，先頭文字切り換え器，アドレス生成器，比較器）に加えて，ホストとの入出力のためのテキストメモリと抽出結果メモリを含んでいる．MEX-I は，約 80 個のメモリ IC（総容量：約 2 MB）と約500個のロジック IC を載せた，合計 12 枚のボードから成る．**図9**に，MEX-I の外観を示す．

アルゴリズムのパラメータは，$N=8$（ボードを増設することで 16 まで拡張可能），$M=1\sim3$ とした．以下の実験において MEX-I に登録した単語辞書の語数 D は約8万語である（自立語だけでなく付属語・接辞なども含む）．

5.2 処理速度

種々のテキスト（小説・エッセイ，新聞記事，技術論文が各9件，合計27件）に関して，形態素抽出所要時間 T を実際に測定した結果を，**図 10** に示す．単語辞書の語数 D は前述のとおり約8万語である．

MEX-I は 10 MHz のクロック（100 n 秒サイクル）

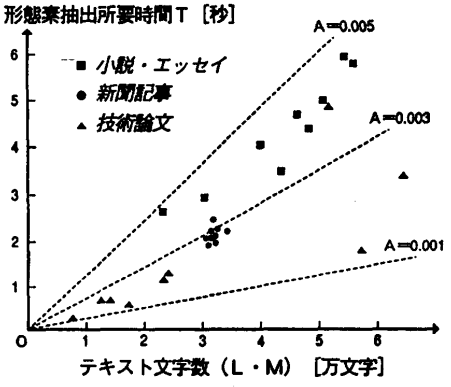


図 10 MEX-I の形態素抽出所要時間
Fig. 10 Implementation time measurement results.

表 1 形態素抽出所要時間の比較
Table 1 Implementation time comparison.

形態素抽出方式（辞書容量）		テキスト5000文字の処理時間					
		テキスト1	テキスト2	テキスト3	テキスト4	平均	(比)
ソフトウェア	二分探索法 (1.2MB)	564秒	642秒	615秒	673秒	624秒	(1248)
	先頭文字表利用二分探索法 (1.1MB)	133秒	153秒	147秒	155秒	147秒	(294)
	順序ハッシュ法 (2.3MB)	406秒	440秒	435秒	416秒	424秒	(848)
	木構造辞書の桁探索法 (1.1MB)	52秒	56秒	54秒	54秒	54秒	(108)
形態素抽出マシン MEX-I (1.2MB)		0.56秒	0.50秒	0.51秒	0.44秒	0.50秒	(1)

で動作している．1単語の照合処理（第3章のアルゴリズムのプロシジャ2）を3クロック（=300n 秒）で実行するように設計した．このとき，形態素抽出所要時間 T は，およそ次式のようになる．

$$T \approx A \cdot D \cdot L \cdot M \cdot 300 \ [\text{n秒}].$$

ここで，D, L, M は前述のとおりで，新たに導入した A は，インデックスによって絞り込まれて実際に照合される単語の D に対する比率である（以下では，A をインデクシング係数と呼ぶ）．対象テキストによってインデクシング係数 A にばらつきがあるが*，図 10 の測定結果では，$A=0.001$〜0.005 におさまっている．

この測定結果は，8万語の単語辞書を用いた形態素抽出処理を，テキスト1万文字当たり1秒程度で実行できる MEX-I の処理速度を示している．

5.3 ソフトウェアとの比較

表1には，形態素抽出所要時間 T を MEX-I と4通りのソフトウェアとで比較した結果を示した．ソフトウェアは，4通りのアルゴリズム（二分探索法[15]，先頭文字表利用二分探索法**，順序ハッシュ法[18),19)]***，木構造辞書の桁探索法[15),16)]）ともC言語でコーディングし，NEC パーソナルコンピュータ PC-98XL² (CPU：80386，クロック：16 MHz) 上で動作させた．その際，8万語の単語辞書は RAM (I・O バンク方式拡張メモリ) に置いた．この比較は，4種類のテキスト（いずれも，$L=5000$，$M=1$，小説・エッセイ）に対して行った．

この比較実験の結果によると，最も単純な二分探索法を用いたソフトウェアと比較して，MEX-I は約1000倍高速である．4通りのなかで最も高速な桁探索法を用いたソフトウェアと比較しても，MEX-I は約100倍高速であることが確認できた．

なお，この比較では，$M=1$（すなわち，テキストの各文字に複数の候補がない）としており，$M>1$ の場合には，第4.1節で述べたように，ソフトウェアに比較した MEX-I の高速性はさらに顕著になる．

* 小説など，かな文字の比較的多いテキストでは，インデクシング係数 A が大きくなる傾向が見られる．

** 図3と同様の先頭文字表による辞書範囲の絞り込みを行った二分探索法である．表1の辞書容量は，先頭文字表＋2文字目以降固定長としたものである．

*** ハッシュ表のサイズは 149999 とし，ハッシュ関数は文献 19) と同じものを用いた．

6. 関連研究

提案するハードウェアアルゴリズムの特長を明らかにするため，本章では，他の分野で開発されたいくつかのハードウェアアーキテクチャについて，形態素抽出への適用可能性を検討する．後述するように，これらの従来アーキテクチャでは，第1章で述べたような自然言語解析の特徴（文字種の多い文字列の処理，大規模辞書の高頻度アクセス）が性能の妨げとなる．

6.1 連想メモリによる複数単語同時照合

指定したアドレスの内容を読み出す通常のメモリに対して，指定した内容を格納したアドレスを検出する連想メモリ[20]がある．その一種で文字列照合機能を強化した ISSP (Intelligent String Search Processor)[10] も開発されている．連想メモリでは，内部に単語辞書を登録すれば，入力テキストに対して，登録した複数の単語を同時に照合でき，形態素抽出処理が実現できる．

しかし，現状の連想メモリ LSI は1個の容量が数K〜数十Kビットであるから，数万語〜数十万語の単語辞書を一度に登録するには，数千個を越える多数の連想メモリ LSI が必要になる．数個〜十数個という現実的な数の連想メモリ LSI を想定し，単語辞書を1回で登録できるサイズの複数個に分割して，その登録・検索を繰り返すようにすると，今度は，繰り返し登録に要する時間が障壁となり，極めて多量のテキストを一括処理するときでないと十分な高速性が得られなくなる[21]．

6.2 マルチプロセッサによる多重検索

複数のプロセッサで，テキストの1文字目・2文字目・3文字目・……から同時に単語辞書を検索する方

法[5]がある．この場合，単語辞書は容量が大きいため共有メモリに置き，複数のプロセッサからのアクセス競合の調整機構を設けることになる．プロセッサ数を増加すると，処理速度は向上するが，それにつれてアクセス競合の頻度が増すので，処理速度の向上は頭打ちになる．一般に，100個のプロセッサを用いても，1個の場合の100倍の処理速度を得ることはできない．

6.3 RAM へマッピングした候補との辞書照合

本稿で提案したアルゴリズムと同様に，1個の単語を構成する全文字位置で同時に一致を判定する手法を用いたものに，音声認識を対象とした浜口らのアルゴリズム[13]がある．

浜口らのアルゴリズムでは，文字（音節）の種類だけのアドレス空間をもつ RAM へ候補をマッピングしておき，照合時には，照合対象の単語辞書側の文字に対応するアドレスの値を RAM から読み出して一致を判定する．照合のために比較器は用いていない．したがって，照合方法に関しては，音声のように文字（音節）の種類が少なく候補数が多い対象には，浜口らのアルゴリズムの方が適しているが，漢字を含み字種の多い日本語テキストには，本稿のアルゴリズムの方が適している．

また，本稿のアルゴリズムでは，シフトレジスタによるテキストの順送り機構によって，テキストの各位置での辞書照合を高速に繰り返すことができる．これに対して，文字（音節）を RAM にマッピングする方法では，テキスト内位置をずらすのに全 RAM の内容の書き直しが必要となり，高速な順送り機構を実現することが難しい．

7. おわりに

基本的な1回の照合サイクルをできる限り短縮する方針で設計し，（1）シフトレジスタによるテキストの順送り，（2）先頭文字による辞書範囲の絞り込み制御，（3）単語を構成する全文字の同時照合など，ワイヤードロジックとして容易に実現できる手法を組み合わせた，新しい形態素抽出ハードウェアアルゴリズムを提案した．本アルゴリズムでは，語数 D の単語辞書を用いた長さ L のテキスト（各文字が M 個の候補をもつ）の形態素抽出所要時間 T が $O(L \cdot M \cdot D)$ となる．従来のソフトウェアアルゴリズムの処理時間が M の累乗オーダになるのに対して，本アルゴリズムは，M の線形オーダに抑えられる優位性をもつ．

さらに，提案するアルゴリズムの実行速度の高速性を検証するために，形態素抽出マシン MEX-I を試作した．その結果，MEX-I では，8万語の単語辞書を用いた形態素抽出処理が，テキスト1万文字当たり約1秒で実行できることを確認した．また，パーソナルコンピュータ（CPU：80386，クロック：16 MHz）上のソフトウェアとの比較では，100倍～1000倍高速という結果を得た．MEX-I では，メモリから読み出して比較する1回の照合サイクルを 300 n 秒で実行しており，より高速アクセス可能なメモリ IC を用いれば，さらに1桁近い高速化も可能である．

本稿で提案したアルゴリズムの高速性は，ワイヤードロジックとして容易に実現可能なシンプルな構成により得られている．このシンプルな構成は，VLSI 化に適しており，既に形態素抽出 LSI（MEXEN）の試作・評価も完了している[22]．

本アルゴリズムおよび形態素抽出マシンの開発は，冒頭で述べたように自然言語解析の高速化を目的とし，文章解析アクセラレータの開発というハードウェアアプローチにしたがったものである．今後，形態素抽出以降の処理（接続検定処理など）についてもハードウェア化を図り，かつ，応用システムも試作してゆくことで，アプローチの有効性をさらに裏付けてゆくことが今後の課題である．

謝辞 文章解析アクセラレータの開発というハードウェアアプローチは，日本電気(株)において，関西C＆C研究所宮井均課長，C＆Cシステム研究所ターミナルシステム研究部大山裕課長とともに検討・提案し，機能エレクトロニクス研究所福地弘道所長代理，ならびにC＆Cシステム研究所ターミナルシステム研究部西谷隆夫部長の叱咤激励のもとで進めてきた．ここで深謝する．

参考文献

1) 松本裕治：論理文法の並列構文解析，情報処理学会論文誌，Vol. 29, No. 4, pp. 335-341 (1988).
2) Rytter, W.: Parallel Time $O(\log n)$ Recognition of Unambiguous Context-free Languages, *Inf. Comput.*, No. 73, pp. 75-86 (1987).
3) 峯 恒憲，谷口倫一郎，雨宮真人：文脈自由文法の並列構文解析，情報処理学会自然言語処理研究会資料，NL-73-1 (1989).
4) 中村貞吾，日高 達：文脈自由文法の並列構文解析法，第39回情報処理学会全国大会論文集，pp. 620-621 (1989).

5) 中村 修，田中明通，菊池英夫：形態素抽出アルゴリズムの高速処理方式，第37回情報処理学会全国大会論文集，pp. 1002-1003 (1988).
6) 福島俊一，大山 裕，宮井 均：文章解析アクセラレータ(1)—形態素抽出マシンの試作—，情報処理学会自然言語処理研究会資料，NL-75-9 (1990).
7) Fukushima, T., Ohyama, Y. and Miyai, H.: A Hardware Algorithm for High Speed Morpheme Extraction and Its Implementation, *Proc. 28th ACL*, pp. 307-314 (1990).
8) 福島俊一：アレイプロセッサによる文脈自由言語の並列認識アルゴリズム，第40回情報処理学会全国大会論文集，pp. 462-463 (1990).
9) 山本昌弘，梅村 護，小長谷明彦，横田 実：文字列処理とアーキテクチャ，情報処理，Vol. 23, No. 8, pp. 719-729 (1982).
10) Takahashi, K., Yamada, H. and Hirata, M.: A String Search Processor LSI, *J. Inf. Process.*, Vol. 13, No. 2, pp. 183-189 (1990).
11) Chu, K. H. and Fu, K. S.: VLSI Architectures for High Speed Recognition of Context-free Languages and Finite-state Languages, *Proc. 9th Annu. Int. Sympo. Comput. Arch.*, pp. 43-49 (1982).
12) Chiang, Y. T. and Fu, K. S.: Parallel Parsing Algorithms and VLSI Implementation for Syntactic Pattern Recognition, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. PAMI-6, No. 3, pp. 302-314 (1984).
13) 浜口重建，鈴木義武：音声日本語入力システムにおける高速な言語処理のための辞書照合アルゴリズム，電子情報通信学会論文誌D，Vol. J70-D, No. 8, pp. 1589-1596 (1987).
14) 長尾 真 (監修)：日本語情報処理，電子通信学会 (1984).
15) Knuth, D. E.: *The Art of Computer Programming Vol. 3 (Sorting and Searching)*, Addison-Wesley (1973).
16) 弓場敏嗣，星 守：木構造を用いた見出し探索の技法，情報処理，Vol. 21, No. 1, pp. 28-49 (1980).
17) 福島俊一，大山 裕，大竹暁子，首藤友喜，首藤正道：盲人用読書器における文音声変換のための文章解析，情報処理学会日本語文書処理研究会資料，JDP-2-4 (1985).
18) Amble, O. and Knuth, D. E.: Ordered Hash Table, *Comput. J.*, Vol. 17, No. 2, pp. 135-142 (1974).
19) 横山晶一，元吉文男，井佐原均：二次記憶上の大規模語彙を用いる自然言語処理システム，情報処理学会論文誌，Vol. 29, No. 6, pp. 570-580 (1988).
20) 小倉 武，山田慎一郎：連想メモリ LSI の現状と今後，電子通信学会誌，Vol. 69, No. 7, pp. 745-751 (1986).
21) 福島俊一，菊地芳秀，大山 裕，宮井 均：多重照合型形態素抽出方式に関する検討，第39回情報処理学会全国大会論文集，pp. 591-592 (1989).
22) 福島俊一：形態素抽出マシン MEX-II，情報処理学会自然言語処理研究会資料，NL-81-1 (1991).

付 録

第3章に示した形態素抽出ハードウェアアルゴリズムの時間計算量が $O(L \cdot M \cdot D)$ であることを証明する．ここで，L はテキスト長，M はテキストの各文字当たりの候補数，D は単語辞書の語数である．

□証明：

アルゴリズム i のプロシジャ j のステップ k について，その所要時間を $T^i{}_{j,k}$，下位のプロシジャの繰り返し回数を $C^i{}_{j,k}$ とする．ただし，基本アルゴリズムは $i=0$，拡張アルゴリズムは $i=1$，メインプロシジャは $j=0$ で表す．

(1) このとき，基本アルゴリズム (第3.1節) による形態素抽出所要時間 T^0 は，次のように表せる．

$$T^0 = T^0{}_{0,1} + C^0{}_{0,2} \cdot (T^0{}_{1,1} + C^0{}_{1,2} \cdot (T^0{}_{2,1} + T^0{}_{2,2}) + T^0{}_{1,3})$$

ここで，$C^0{}_{0,2}$ は $O(L)$，$C^0{}_{1,2}$ は $O(D)$ であり，$T^0{}_{0,1}$, $T^0{}_{1,1}$, $T^0{}_{1,3}$, $T^0{}_{2,1}$, $T^0{}_{2,2}$ はいずれも L, D に依らず一定である．

したがって，T^0 は $O(L \cdot D)$ である．基本アルゴリズムでは，$M=1$ であるから，$O(L \cdot M \cdot D)$ を満たしている．

(2) 次に拡張アルゴリズム (第3.2節) では，形態素抽出所要時間 T^1 は次のようになる．

$$T^1 = T^1{}_{0,1} + C^1{}_{0,2} \cdot (T^1{}_{1,1} + C^1{}_{1,2} \cdot (T^1{}_{1.5,1} + C^1{}_{1.5,2} \cdot (T^1{}_{2,1} + T^1{}_{2,2}) + T^1{}_{1.5,3}) + T^1{}_{1,3})$$

$C^1{}_{0,2}$ は $O(L)$, $C^1{}_{1,2}$ は$O(M)$, $C^1{}_{1.5,2}$ は $O(D)$ である．$T^1{}_{0,1}$, $T^1{}_{1,1}$, $T^1{}_{1,3}$, $T^1{}_{1.5,1}$, $T^1{}_{1.5,3}$, $T^1{}_{2,1}$, $T^1{}_{2,2}$ はいずれも L, M, D に依らず一定である．したがって，T^1 は $O(L \cdot M \cdot D)$ である．

(3) ユーザ登録機能 (第3.3節) を設けた場合，$T^{0/1}{}_{1,2.5}$ が加わるが，$T^{0/1}{}_{1,2.5}$ は L, M, D に依存しないので，時間計算量 $O(L \cdot M \cdot D)$ は変わらない．

(証明終わり)□

(平成3年3月19日受付)
(平成3年6月13日採録)

福島 俊一（正会員）

1958年生．1982年東京大学理学部物理学科卒業．同年日本電気(株)入社．現在，C＆C システム研究所ターミナルシステム研究部主任．自然言語処理システムの研究開発に従事．人工知能学会，計量国語学会，ACL 各会員．