社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2006－ＳＬＤＭ－126（13）
2006／10／26

C言語による音声認識システムの設計とその最適化

才辻 誠† 神戸 尚志‡

†近畿大学大学院総合理工学研究科 〒577-8502 大阪府東大阪市小若江 3-4-1

‡近畿大学理工学部電気電子工学科 〒577-8502 大阪府東大阪市小若江 3-4-1

E-mail: †0633340413y@kindai.ac.jp, ‡tkambe@ele.kindai.ac.jp

あらまし 大語彙連続音声認識において、出力確率計算処理ならびに Viterbi 探索に高い処理能力が必要となる。携帯端末へ搭載するため、低速な CPU と専用ハードウエアによるリアルタイム認識を目指すために第一パス処理に注目する。本稿では、動作合成を可能とする Bach システムを用いて第一パス処理のハードウェア設計を行い、特に出力確率計算処理ならびに Viterbi 探索の最適化手法を提案しその性能を評価する。

キーワード 音声認識，リアルタイム，最適化，Bach

C-based Design and its Optimization for a Speech Recognition System

Makoto SAITSUJI† and Takashi KAMBE‡

† Graduate School of Science and Engineering, Kinki University

3-4-1 Kowakae, Higashi-Osaka, Osaka, 577-8502 Japan

‡ School of Science and Engineering Department of Electric and Electronic Engineering, Kinki University

3-4-1 Kowakae, Higashi-Osaka, Osaka, 577-8502 Japan

E-mail: †0633340413y@kindai.ac.jp, ‡tkambe@ele.kindai.ac.jp

Abstract It is important to speed up the output probability calculation and the Viterbi search in the first pass process of the large vocabulary speech recognition for real time operation of hand held system with low speed processor. In this paper, we design and optimize the circuits of the first-pass process using Bach system, and evaluate its performance.

Keyword Speech Recognition, Real-time, Optimization, Bach

1. はじめに

音声認識技術は、キーボードやマウスより便利で自然な形での操作が可能な入力インターフェイスとして期待されている[1,2]。近年、半導体技術の進歩により音声認識技術が実用化され、カーナビゲーションのボイスサーチ・コマンドの自動応答受付などに音声認識が使われている。しかし、これらは、主に決められた単語に対して認識可能であり、文章による入力には対応していない。文章による音声認識を大語彙連続音声認識、または連続音声認識と呼ばれ、現在では、パソコン上で動作するソフトウェアとして実現されている。大語彙連続音声認識では、音響モデルとして隠れマルコフモデル（HMM）が用いられ、言語モデルとして N-gram を用いている。音声認識の根幹技術の一つである HMM を用いた認識は、認識性能が高い反面、出力確率を求める計算量や Viterbi 探索回数などが多くなる。そのため、ソフトウェアでリアルタイム音声認識を行うには高い処理能力を持つ CPU が必要となる。

本研究では、大語彙連続音声認識エンジン「Julius」[3]をもとに、ハードウェアとソフトウェアで構成するシステムを設計し、携帯端末への搭載を目指し、出力確率を求める計算及び Viterbi 探索を含む第一パス処理をハードウェア化することでシステム全体の大幅な効率化を提案する。HDL によるハードウェア設計より抽象度の高い設計が行なう Bach システム[4]を用いて設計した。Bach システムでは、ハードウェアの並列動作やパイプライン動作などを記述出来るように ANSI-C を拡張した動作合成システムである。第一パス処理のハードウェア設計にあたり、Get Backtrellis Thread、Outprob Thread の最適化を行い、回路の性能を評価する。

2. 音声認識技術

2.1. 音響分析

音声は、時間的に特徴量の変化が激しく、そのため、図 2.1.1 に示すように一定の時間幅(フレーム長)で切り出し、定常確率過程に従うと仮定してスペクトル解析し音響特徴量を算出する。その際に、切り出された

フレームの両端において不連続が起こりやすいため、フレーム区間の半分程度の長さで重ね合わせながらフレームをシフトして解析する。Julius は、フレーム長 25ms、フレーム間隔 10ms としてフレーム化処理を行われる。

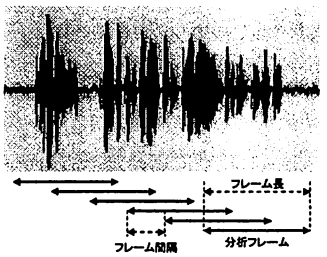

図 2.1.1 フレーム分析

2.2. 大語彙連続音声認識

大語彙連続音声認識技術とは、数万語の語彙を対照とした文章認識である。そのため、探索空間は膨大なため、音響モデル・言語モデルを組み合わせた認識システムが用いる。

大語彙連続音声認識エンジンである Julius は、その最大の特徴として可搬性を持たせて作られており、組み込むモデルファイルを変更することで幅広い用途に対応している。探索アルゴリズムでは、図 2.2.1 に示すように、第一パス処理と第二パス処理のマルチパス探索で行われている。第一パス処理では、音響モデルとなる音素 HMM から状態遷移・出力確率値を求め、また、言語モデルとなるバイグラムからは言語出現確率値を求めることで、中間結果（文の候補）が残される。第二パス処理では、第一パス処理で得た文の候補を、更にバイグラムより精度の高いトライグラムを用いて再探索・再評価され最終的な文の候補が出力される。本研究では、95%の認識精度を誇る高精度版 Julius を用いての設計・検討を行う。

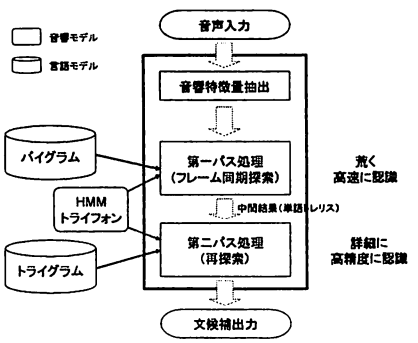

図 2.2.1 処理フロー（マルチパス検索）

2.3. HMM(隠れマルコフモデル)

音声中の音素は、連続する複数のフレームに対応する。そのため、音素の継続時間とは、環境に応じて変化する。このように、変化する時系列に対して確率的に生成することが出来る信号モデル（音響モデル）として HMM(隠れマルコフモデル)が用いられる。

HMM では、信号系列に対して定常と見なす「状態」を定義し、状態から状態へ遷移する確率（状態遷移確率）と状態ごとに信号が観測される確率（出力確率）が与えられる。特に、音声認識では、フレーム単位の音響特徴量は、p 次元ベクトルで表現され、音響特徴量の確率分布は複数のガウス分布を組み合わせた混合ガウス分布で表現される。そのため、混合ガウス分布による出力確率は以下の計算で求める。ここで、p 次元、t 番目フレームの音響特徴量ベクトルを o_{tp} とし、i 状態 m 混合目のガウス分布計算 b_{im} は、以下の式で与えられる。

$$\ln b_{im}(o_t) = \omega_i - \frac{1}{2}\sum_{p=1}^{P}\frac{1}{\sigma^2_{imp}}\left(o_{tp} - \mu_{imp}\right)^2 \qquad (1)$$

ここで、ω_i を混合重み値, μ_{imp} を平均ベクトル, σ^2_{imp} を分散ベクトルとする。これらは、HMM の学習時に計算により求められ、M 混合ガウス分布の出力確率計算値 b_i は、以下の式で与えられる。

$$\log b_i(o_t) = \log\sum_{m=1}^{M} b_{im}(o_t) \qquad (2)$$

2.4. N-gram

音声認識は、入力された音声 x に最もよく合う言語表現 w を推定するもので、次式のように最適な結果 w が推定される。

$$\begin{aligned}\hat{w} &= \arg\max_{w} P(w \mid x) \\ &= \arg\max_{w} P(x \mid w)\cdot P(w)\end{aligned} \qquad (3)$$

ただし、音響モデルより $P(x|w)$ が生成される。また、単語列 $w=\{w_1 w_2 \ldots w_n\}$ の出現する確率 $P(w)$ は、次式で近似される。

$$P(w_1 w_2 \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_{i-N+1} \ldots w_{i-1}) \qquad (4)$$

それぞれ、($N=1$) ユニグラム、($N=2$) バイグラム、トライグラム ($N=3$) と呼ばれ言語出現確率が求められる。これらより、与えられた信号系列を最も高い確率で生成する状態遷移系列を求めるため、Viterbi 探索が行わる。

3. 音声認識システム回路の設計

3.1. Bach-C を用いた設計工程

ハードウェアの設計手法は、ゲートレベルの設計や、レジスタ転送レベルのハードウェア設計言語の VHDL、Verilog-HDL を用いた設計が挙げられる。半導体技術

の進歩により、1つのLSIに数千万ゲートの回路が搭載可能となったため、現在の主流は、HDL記述による設計である。近年、HDLよりもさらに抽象度の高い記述での回路設計が行われている。これらは、動作合成システムと言われ、C言語やそれに近いアルゴリズム記述からのハードウェアの設計を行う。本研究で使用したBachもこの動作合成を実現したシステムの1つである。

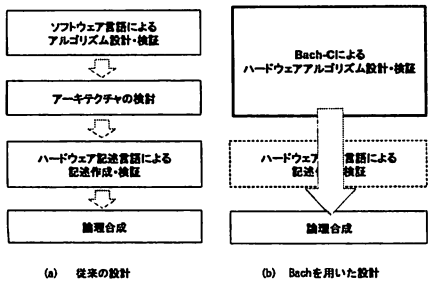

図3.1.1 Bachの設計工程

図3.1.1に示すように、従来のHDLによるハードウェア設計は、C言語によるアルゴリズム設計を行い、その後ソフトウェアを使った検証、HDLでの回路設計、RTL論理合成、シミュレーションといった検証手順で行われる。そのため、HDLで回路設計の不具合が見つかった場合、修正に大きな手間が必要となる。Bachを用いた設計では、アルゴリズムの設計段階から抽象度の高いBach-C言語を用い、機能設計・検証を行うことで、HDLによる機能検証が削減されるだけでなく、Bach-C記述からRTLのVHDLが自動生成される。また、各処理の計算時間や回路規模見積り、シミュレーションによるボトルネックの分析、機能の分割や並列化、中間メモリの自動生成、スループット指定によるパイプライン回路の自動生成などにより、各種アーキテクチャを短期間に探索できる。

3.2. 音声認識システムの分析

一般に各種システムにおいてハードウェア化は、高速化、低消費電力化のために行われ、ソフトウェアは、柔軟性を持たせて処理を行なう時に効果的である。ソフトウェアをハードウェア化するあたり、構造によって適切な部分が異なる。本文では、ソフトウェア処理の中で、処理が独立している部分をハードウェア化し全体の処理を大きく変更することなくソフト-ハード協調システムを実現する。まず、計算量が多い部分を分析し、Juliusのハードウェア化する箇所を決定する。

図3.2.1は、高精度版Juliusの構造解析と処理時間の分析結果である。第一パス処理が、全体の処理時間の67%を占めていることがわかる。その内訳として、出力確率計算(63%)、N-gram探索(6%)、尤度ソート(5%)、Viterbi探索etc(26%)が挙げられる。

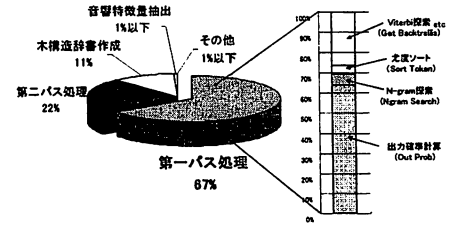

図3.2.1 プロファイルリング結果

3.3. 第一パス処理のアーキテクチャ

第一パス処理は、音声認識において独立した処理のひとつである。また、プロファイリング結果より第一パス処理が、大語彙連続音声認識を行う上でのボトルネックとなると考えられる。

図3.3.1に、提案する第一パス処理のアーキテクチャを示す。それぞれ、「出力確率パラメータ」(=出力確率計算を行うための平均・分散・重みを格納されたもの)、「状態遷移確率など」(=状態遷移に必要となる遷移確率・遷移先情報などを格納したもの)、「言語出現確率」(=N-gram探索に必要となるユニグラム・バイグラムを格納したもの)を外部RAMとして配列宣言する。また、予め複数回呼ばれている関数に対してそれぞれスレッド分割(OutProb Thread, Get Backtrellis Thread, Ngram Search Thread, Sort Token Thread)することにより演算の共有化を図り回路規模を削減する。本回路の動作は、以下の通りである。1フレーム分の音声特徴量をレジスタに格納し同期信号(in1)で第一パス処理全体の開始を促す。Get Backtrellis Threadは、同期信号(in2, in3)を用い、OutProb Threadによる出力確率計算、Ngram Search ThreadによるN-gram探索を開始させる。同期信号(out2, out3)より得られた確率からスコアが算出され、最終的にスコアの高いものを候補として残すためSort Token Threadより尤度ソートが行われる。

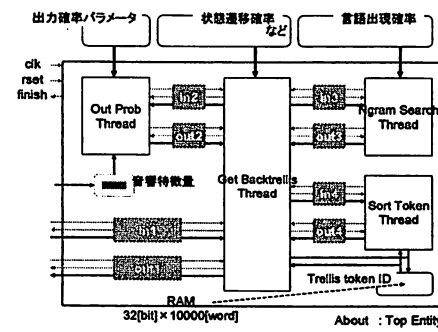

図3.3.1 第一パス処理のアーキテクチャ

3.4. 固定小数点演算ビット数の決定

第一パス処理における確率計算では、ソフトウェアで浮動小数点演算となっているが、ハードウェアでは浮動小数点演算を実装すると回路規模が大きくなるため、固定小数点演算での実装を検討する。固定小数点演算とする場合、その整数ビットと小数ビットで認識率が保証できる最小のビット数を決定する必要がある。そこで、実験的に様々なビット数で音声を認識させてソフトウェアの認識率と比較をするため、Bach システムのC言語ベースシミュレーションを用いてその検証を行った。表 3.4.1 に、その結果を示す。

表 3.4.1 認識率とビット数

ビット数			認識率
浮動小数点演算			95.28[%]
小数点演算	整数部	小数部	
固定	13[bit]	19[bit]	28.18[%]
	14[bit]	18[bit]	60.50[%]
	15[bit]	17[bit]	93.06[%]
	16[bit]	16[bit]	94.44[%]
	17[bit]	15[bit]	95.28[%]
	18[bit]	14[bit]	18.51[%]

これらより、更に整数部を減らすとオーバーフローとなり、小数部を減らすとアンダーフローの原因となり認識率が低下する。よって、整数 14bit、小数 18bit の固定小数点演算にすることでソフトウェアでの浮動小数点演算を行ったときと同様の認識率を維持できることが確認できる。

3.5. 第一パス処理のコード量比較

設計を行う上で、設計期間は重要な指標と言える。ただし、設計期間は、コーディングを始めてから回路動作の正しいことが確認されるまでの日数とする。今回設計した第一パス処理の設計期間の見積もりを算出するためコード量による比較を行った。表 3.5.1 は、第一パス処理の Bach-C 記述のコード量とそれより自動生成された RTL 記述のコード量を示したものである。

表 3.5.1 第一パス処理のコード量比較

	Bach-C 記述 [行]	RTL 記述 [行]
コード量	2,440	85,007

Bach-C 記述での設計に約 3 ヶ月を必要とした。そのため、仮に RTL 記述での設計を行った場合、約 12 ヶ月以上を要すると考えられる。

4. 第一パス処理のハードウエア化

リアルタイムを目指すためには、第一パス処理をハードウエア化することで必要がある。特に、処理時間が長い Viterbi 探索スレッドと出力確率計算スレッドの最適化を述べる。

4.1. Viterbi 探索スレッド

最も高い確率の状態遷移系列を求めるために、その遷移は数多く存在する。探索の高速化のための最適化手法を以下に示す。

4.1.1. 音素コードの変換

日本語の音素の表記は、ローマ字表記の母音と子音を分類され、その種類数は約 40 個となる。最適化前の回路において、音素表記をアスキーコードによる表現方法であった。そこで、約 40 種類の音素に対してそれぞれ番号を割り当て表現することで、コード量の削減と高速化を図る。図 4.1.1.1 は、アスキーコードによる表現方法と新たなコードによる表現方法の例を示したものである。

トライフォン	アスキーコード	新たなコード
a-k	61 96 6B 00 a - k ¥0	01 11 00 a k *
a+k	61 2B 6B 00 a + k ¥0	00 01 11 * a k
:	:	:
sp-a+k	73 70 69 61 2B 6B 00 s p - a + k ¥0	29 01 11 sp a k

図 4.1.1.1 新たなコードによるトライフォン表現

トライフォンとは、前後の音素を考慮した音素のことで、新コード表現においては、左音素、中央音素、右音素に割り当てられた番号で表現する。

4.1.2. メモリアクセスの高速化

最適化前の回路において、Viterbi 探索が行われる際外部 RAM に頻繁にアクセスが行われる。また、外部 RAM へのアクセスには 2 サイクル必要とするためレジスタファイル（トレリスレジスタ）を用意することでそのアクセス回数の削減を図る。

また、図 4.1.2.1 には、以上 2 つの最適化を行った場合における Viterbi 探索スレッドの流れを示す。

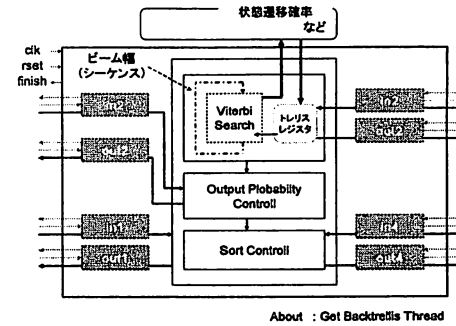

図 4.1.2.1 Viterbi 探索スレッドの流れ

4.2. 出力確率計算スレッド

HMM における出力確率計算は、演算回数が多く、また、演算時間に長い時間が必要となる。そこで、以下の最適化を行うことで効率化を図る。

4.2.1. ガウス分布計算のパイプライン化

ガウス分布計算において、多くの除算が行われており、その演算時間は長く、回路全体の処理時間に大きく影響する。除算回路を複数のステージに分割することでガウス分布計算のパイプライン実装を行った。それぞれ、外部 RAM「出力確率パラメータ」より必要となる値の読み出しに 4 サイクルとなるため、1 ステージ 4 サイクルとしてパイプライン設計を行う。また、1 bit の商を生成する除算回路を 6 つ用意することで必要精度の認識が得られた。除算回路においては、高速演算可能とされる冗長 2 進を用いた演算方法を用いて最適化を行った。(図 4.2.1.1 参照)

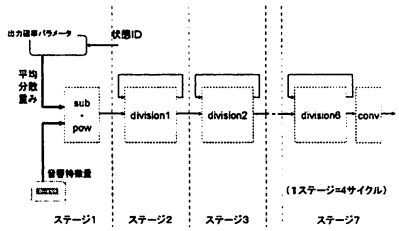

図 4.2.1.1 ガウス分布計算のパイプライン化

4.2.2. FSA 法による指数・対数計算回路の設計

出力確率計算には、式 (1) のように指数・対数の計算が多く行われる。まず、テーラー展開を用いて実装していたが、これを高速化するため Faster Shift and Add algorithms[3]「FSA 法」(図 4.2.2.1 参照) を用いて指数・対数計算用組み込み回路を設計した。

FSA 法は、テーラー展開アルゴリズムに比べ収束が早く処理時間の短縮が期待される。

$$L_{n+1} = L_n - \ln(1 + d \cdot 2^{-n})$$

$$E_{n+1} = E_n(1 + d \cdot 2^{-n}) = E_n + d \cdot E_n \cdot 2^{-n}$$

$$d_n = \begin{cases} 1 \text{ if } L_n \geq \ln(1 + 2^{-n}) \\ 0 \text{ otherwise} \end{cases}$$

入力 $L_0 = x$
出力 $E_n = \exp(x)$
初期値 $E_0 = 1,\ n\ (n = 0, 1, 2, 3 \cdots)$

$$T_{n+1} = T_n + \ln(1 + d \cdot 2^{-n})$$

$$E_{n+1} = E_n(1 + d \cdot 2^{-n}) = E_n + d \cdot E_n \cdot 2^{-n}$$

$$d_n = \begin{cases} 1 \text{ if } E_n(1 + 2^{-n}) \leq E_0 \\ 0 \text{ otherwise} \end{cases}$$

入力 $E_0 = x$
出力 $T_n = \ln(x)$
初期値 $T_0 = 0,\ n\ (n = 0, 1, 2, 3 \cdots)$

図 4.2.2.1 指数(exp)・対数(log)の FSA 法

4.2.3. 出力確率計算の効率化

音声認識では、音素を HMM でモデル化されるためその状態数 (1972 状態) が音素の種類に対応する。また、音素の種類として母音のように頻繁に出現するものや子音のようにあまり出現しないものに分類される。そのため、出力確率の計算回数においても違いが生じる。図 4.2.3.1 は、5 フレーム目と 10 フレーム目における出力確率計算回数を示したものである。そこで、一度算出された確率に対して内部 RAM (OutProb Cach) に格納し、再度計算されることを防ぐことでの演算回数の最適化を図る。

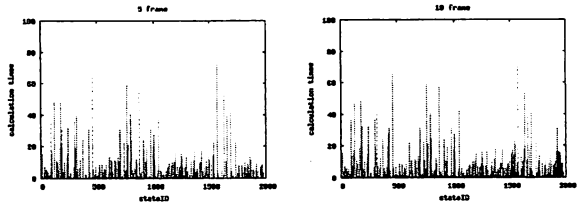

図 4.2.3.1 出力確率計算回数の分布

以上の 3 つの最適化を行った出力確率計算スレッドの流れを図 4.2.3.2 に示す。また、ガウス分布計算と指数・対数計算は、それぞれシーケンシャルに行われる。

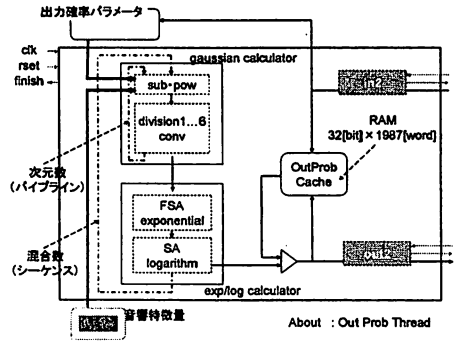

図 4.2.3.2 出力確率計算スレッドの流れ

5. 結果

Viterbi探索スレッドと出力確率計算スレッドに対して最適化を行った結果について表 5.1、表 5.2 に示す。ただし、高精度版 (ビーム幅、1500 個) とし動作周波数を 100MHz としたものである。

表 5.1 第一パス処理の回路規模

	回路構成	回路規模 [ゲート数]
①	最適前	194,270
②	① + 音素コード変換	188,066
③	② + メモリアクセスの高速化	187,287
④	③ + ガウス分布計算のパイプライン実装	197,777
⑤	④ + FSA 法による指数・対数の実装	132,209
⑥	⑤ + 出力確率キャッシュの実装	129,981

※ ゲート数は、NAND2 換算とする。

※ RAM は、ゲート数に含めないものとする。

表 5.2 第一パス処理の処理時間

フレーム番号	最適前 ① [ns]	最適化② [ns]	最適化 ③ [ns]	最適化 ④ [ns]	最適化 ⑤ [ns]	最適化 ⑥ [ns]
0	4,186,510	4,113,330	4,113,330	4,053,970	4,024,990	4,044,810
1	373,410	227,070	226,880	88,630	49,400	70,060
2	553,700	334,190	333,810	151,390	74,900	94,800
3	2,616,562,450	128,515,910	128,332,950	49,450,860	31,949,810	3,038,130
4	5,235,869,920	259,996,350	258,463,680	102,809,560	66,033,860	10,503,400
5	5,629,493,680	275,571,980	273,807,310	107,332,410	70,871,920	14,195,610
6	5,629,493,680	412,301,110	407,314,890	175,952,300	141,301,410	24,461,660

回路規模は、②から⑥の最適化を行うことで、最適前に比べ、33%削減した。また、処理時間は、6 フレーム目のデータにおいて約 230.1 倍の高速化ができた。回路規模縮小の理由としては、ビット幅の削減（最適化②）やアドレスデコーダの削減（最適化③）などが挙げられる。処理時間において大幅な最適結果が得られたが、音声認識においてリアルタイムの動作を目指すためには第一パス処理を 10ms 以下にする必要があるため、更なる最適化が必要とされる。

6. まとめと今後の課題

本研究では、音声認識システムの計算時間を多く要している部分（第一パス処理）をハードウェア化することにより、システム全体の効率化を提案した。その設計において、音声認識での必要なビット幅を検証しそのコード量の比較を行った。また、第一パス処理の中で Get Backtrellis Thread、Outprob Thread を最適化することで第一パス処理の高速化を行った。これより、リアルタイムへと一歩近づくことが出来た。

今後の課題としては、リアルタイム認識には 1 フレーム分を 10ms 以内にする必要があるため第一パス処理を構成する Ngram Thread、Sort Thread の最適化や、ソフトウェア・ハードウェア協調検証によるシステム全体の性能評価を行う必要がある。

7. 謝辞

Bach を用いたハードウェア設計を実現するに当たり、多大なる御指導を頂いたシャープ株式会社 IC 事業本部要素技術開発センター山田晃久様をはじめ、BACH 開発グループの皆様に心から御礼申し上げます。

音声認識技術を使用した研究を行うにあたり、オープンソースとして大語彙音声認識エンジン「Julius」を公開し、ソフトウェアの解析にあたってご指導いただいた名古屋工業大学大学院情報工学専攻 李晃伸助教授に深くお礼申し上げます。

本研究は、東京大学大規模集積システム設計教育研究センターを通し、シノプシス株式会社の協力で行われたものである。

文　献

[1] 平成 15 年特許庁総務部技術調査課：“音声認識技術に関する特許出願技術動向調査報告” http://www.jpo.go.jp/shiryou/pdf/gidou-houkoku/voice_recognition.pdf.

[2] 中村哲：“音声ヒューマンインターフェース”、専門講習会講演論文集ヒューマンインターフェース技術の最新動向、26～40(2003).

[3] 鹿野清宏、伊藤克亘、河原達也、武田一哉、山本幹雄：“音声認識システム”、オーム社、東京、（2002）.

[4] “Bach システムマニュアル”、シャープ株式会社提供(2004).

[5] 吉沢真吾、宮永喜一、吉田則信：“一括並列処理による HMM 高速化手法及びその VLSI 設計”、電子情報通信学会論文誌、Vol.J85-A、1440～1450（2002).

[6] Sergiu Nedevsschi, Rabin K. Patra, Eric A. Brewer: “Hardware Speech Recognition for User Interfaces in Low Cost,Low Power Devices”,proc. of 42th DAC,(2005).

[7] K.Okada, A. Yamada, T. Kambe: "Hardware Algorithm Optimization Using Bach C", IEICE Trans. Fundamentals vol.E85-A, No.4, (pp835-841), 2002.

[8] Muller,J.M.: “Elementary Functions”, Birkhauser, Boston,(1997).

[9] 奥村晴彦:“C 言語による最新アルゴリズム事典”、技術評論社、東京、(1991).

[10] 才辻誠, 松野裕之, 奥田真如, 山田晃久, 神戸尚志：“音声認識システムのハードウェア”、19 回回路とシステム軽井沢ワークショップ論文集,p205-210,(2006).