ハードウェアにおける高速なオーディオフィンガープリント生成システムの性能評価

荒木光一[†] 佐藤幸紀[††]
V.K. Jain[†††] 井口寧[††]

オーディオフィンガープリンティング技術は，インターネット上のアプリケーションで利用されている．インターネット速度の向上に伴い，オーディオフィンガープリント（FPID）の生成時間は重要になる．本稿では，楽曲から高速にFPIDを生成するために，ハードウェアでFPID生成を行うHiFPを提案し，HiFPのハードウェア化について評価する．既存システムはFPID生成に高速フーリエ変換を用いるが，HiFPではハードウェア実装に適し，高速なFPID生成のために離散ウェーブレット変換のサブバンド分解を用いる．FPGAによる評価の結果，HiFPは既存システムと比較してスループットを向上させ，使用リソース数も削減できた．そして，HiFPは70.3Gbpsのネットワーク上で適応できることが分かった．ロバスト性と識別の信頼性も既存システムと比較して向上した．

Performance Evaluation of High-speed Audio Fingerprint Generation System on Hardware

Koichi Araki,[†] Yukinori Sato,[††] V.K. Jain[†††]
and Yasushi Inoguchi[††]

Audio fingerprinting techniques are uesd by internet applications. As the internet connection speed increases, the speed of the audio fingerprint (FPID) generation becomes more important. In this paper, we propose a HiFP to realize the high speed FPID generation and evaluate the implementability in hardware. In order to generate an FPID from an audio file, the HiFP uses subband decomposition of Discrete Wavelet Transform which is suitable for hardware implementation. We compare the HiFP with a conventional FFT based system. The HiFP implemented on an FPGA can increase throughput and reduce required resources. The HiFP can be used on the 70.3 Gbps network.The HiFP can enhance robustness and reliability of audio identification.

1. はじめに

近年，任意の楽曲を識別する技術としてオーディオフィンガープリンティング技術（AFT）が注目されている[1)~4)]．AFTは楽曲の知覚的特徴を用いて生成されるオーディオフィンガープリント（FPID）によって楽曲の識別を行うので，電子透かしと比べて音質の劣化に対するロバスト性に優れている．このことから，AFTはブロードキャストモニタリング[5)]やP2Pファイルシェアリング上の著作権侵害防止[6)]などのようにインターネット上で利用されている．

インターネット速度の向上により，インターネット上で転送されている楽曲ファイルからFPIDを生成する時間は重要となる．しかし，これまでのAFTの研究はロバスト性と識別の信頼性の向上を焦点としているため，FPID生成の高速化に関する評価が行われていない[2)~5),7)]．例えば，10Gbpsのネットワーク上で約3分のMP3（3,351KB）を転送する場合，転送時間は2.68msである．FPID生成はPCMデータを利用するため，MP3をデコードする必要がある．従って，リアルタイムに処理するためには2.68ms内でデコードとFPID生成を行う必要がある．今後，インターネット速度の向上により転送時間が更に短くなるため，FPID生成の高速化は重要となる．

そこで，本稿では，高速にFPIDを生成するためにハードウェアを用いたHigh-speed Audio Fingerprint

† 北陸先端科学技術大学院大学 情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology
†† 北陸先端科学技術大学院大学 情報科学センター
Center for Information Science, Japan Advanced Institute of Science and Technology
††† 南フロリダ大学 電気工学科
Department of Electrical Engineering, University of South Florida

Generation System（HiFP）を提案する．楽曲ファイルからの知覚的特徴の取得に，従来の FPID 生成法は高速フーリエ変換（FFT）を用いる．しかし，従来の FPID 生成法をハードウェア化した場合，FFT がハードウェア処理全体においてボトルネックとなり，また，乗算器や大容量メモリが必要となるため，回路も大規模になることが報告されている[8]．そこで，HiFP の FPID 生成には，知覚的特徴の取得に離散ウェーブレット変換（DWT）のサブバンド分解を用いる．DWT のサブバンド分解は高い並列性を持ち，基底関数に Haar ウェーブレットを用いることで整数の加減算で構成できる．これにより，FFT ベースの従来の FPID 生成法のハードウェアと比較して，HiFP は高速で簡素なハードウェアを実現できる．本稿では，HiFP のロバスト性と識別の信頼性の評価だけでなく，FPGA でハードウェア化による FPID 生成時間と回路規模の評価も行う．

2. オーディオフィンガープリンティング技術

2.1 オーディオフィンガープリンティング技術について

AFT は，楽曲の知覚的特徴から生成された FPID により，未知の楽曲を識別する技術である．楽曲の識別は，未知の楽曲の FPID とデータベースに保存されている多量の楽曲の FPID を比較して行われる．

図 1 に，AFT のシステムモデルを示す．AFT のシステムは FPID 生成器，FPID 比較器とデータベースで構成される．データベースには事前に FPID 生成器で生成された多量の楽曲の FPID が保存されている．FPID 生成器は，楽曲内の知覚的特徴を用いて FPID を生成する．FPID 比較器は FPID 生成器から出力された未知の楽曲の FPID とデータベース内の FPID の比較し，未知の楽曲の FPID がデータベース内のある FPID と同一か否かを識別する．本稿の焦点は FPID 生成器であり，提案する HiFP は FPID 生成器に適応される．

2.2 関連研究

AFT の FPID 生成に関して多くの研究が行われてきた[2)~4),7)]．これらの先行研究は，知覚的特徴として楽曲の信号の周波数成分を用い，FPID を生成する．周波数成分の取得には FFT[2),4)] や連続ウェーブレット変換[3)] を用いている．周波数成分からの FPID 生成には，スペクトラムの差分[2)] やウェーブレット係数の量子化[7)] が用いられている．これにより，ロバスト性や識別の信頼性が優れている FPID を生成できる．しかし，これらの研究ではロバスト性や識別の信頼性の評価は行われているが，FPID 生成の高速化については議論のみで評価されていない．インターネット上で AFT を利用する場合，インターネット速度の向上により FPID 生成の高速化は今後無視できない．

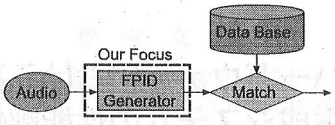

図 1 オーディオフィンガープリンティング技術のシステムモデル
Fig. 1 System model of audio fingerprinting technique

FPID 生成の高速化とハードウェア化に関する研究に磯永ら[8)] の研究がある．彼らは，Philips オーディオフィンガープリンティングシステム[2)] を FPGA でハードウェア化した．ハードウェア処理により，ソフトウェア処理と比較して FPID 生成時間を大幅に短縮した．しかし，彼らのシステムは，FFT がボトルネックとなり 3.1Gbps までのネットワークが限界であるため，現在の標準規格である 10Gbps のネットワークに適応することは困難である．FFT の処理時間は，全処理時間の 90% を占めている．

これらの先行研究に対し，本稿ではロバスト性と識別の信頼性の評価だけでなく FPID 生成時間に関しても評価を行う．磯永ら[8)] の知見を踏まえて，提案する HiFP では周波数成分の取得に並列性の高い DWT のサブバンド分解を用いる．これにより，ハードウェアで FPID 生成を更に高速化する．

3. HiFP

3.1 DWT のサブバンド分解アルゴリズムの並列性

HiFP で用いる DWT のサブバンド分解は高い並列性をもち，Haar ウェーブレットを基底関数にすることで整数の加減算で構成できる．DWT のサブバンド分解は，入力信号に対してハイパスフィルタ，ローパスフィルタとダウンサンプリングによって，ナイキスト周波数を半分にした高周波成分 Hi と低周波数成分 Lo に分割する．式 (1) と式 (2) に，ダウンサンプリングを含み，基底関数が Haar ウェーブレットのハイパスフィルタとローパスフィルタをそれぞれ示す．

$$Hi[i] = (X[2 \times i] - X[2 \times i + 1])/2 \quad (1)$$

$$Lo[i] = (X[2 \times i] + X[2 \times i + 1])/2 \quad (2)$$

ここで，X は入力信号であり，i はイテレータである．入力信号のサンプル数は 2 の二乗個である．次に，得られた低周波数成分 Lo に対してさらに高周波成分と低周波数成分に分割する．これを繰り返すことで任意の時間周波数成分を取得できる．

図 2 に，上記で説明した DWT のサブバンド分解のアルゴリズムを示す．入力として入力信号 wav，信号のサンプル数 n と出力する時間周波数成分のサンプル数 m を与える．出力は時間周波数成分である．このアルゴリズムは，2 つの繰り返し命令を持つ．6-9 行

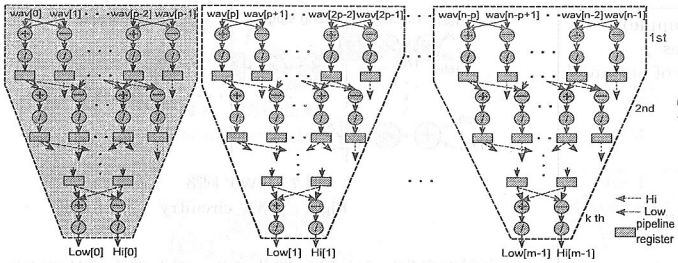

図 3 DWT のサブバンド分解のデータフローグラフ
Fig. 3 Data flow graph of subband decomposition of DWT

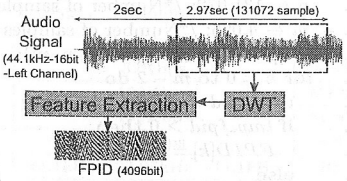

図 4 HiFP のフレームワーク
Fig. 4 Framework of HiFP

```
1. DWT(wav[] ← Input signal,
2.          n ← Number of samples,
3.          m ← Number of samples of output){
4.   while n ≥ m do
5.     n/ = 2;
6.     for i = 0 to n do
7.       Hi[i] = (wav[2 × i] − wav[2 × i + 1])/2;
8.       Lo[i] = (wav[2 × i] + wav[2 × i + 1])/2;
9.     end for
10.    wav[] ← Lo[];
11.  end while
12.  return (Hi, Lo);}
```

図 2 DWT のサブバンド分解のアルゴリズム
Fig. 2 Algorithm of subband decomposition of DWT

目の for 内の処理は入力信号である wav 配列に対して依存関係がないため，空間的並列性を利用して処理できる．4-11 行目の While 内の処理は，10 行目で低周波数成分 Lo を wav 配列へ代入した後に次の While 内の処理をするため，データ依存が存在する．

図 3 に，図 2 のデータフローグラフを示す．n サンプルの入力信号から m サンプルの時間周波数成分を取得するために，ステージ数 k ($=\log_2 n - \log_2 m$) のデータフローグラフが形成される．図 3 のデータフローグラフをハードウェア化する場合，for 内の処理の空間的並列性を利用して並列処理できる．また，各ステージ間にレジスタを挿入することで，動作周波数の向上もできる．しかし，入力信号の全サンプルを同時に入力するため，入力信号のサンプル数が多量であり，ステージ数 k が多い場合，ハードウェアは大規模になる．そこで，入力信号を p ($=2^k$) サンプルずつ分割して入力し，1 サンプルを出力するハードウェア構成（図 3 の点線で囲まれた薄いグレー部分）でパイプライン処理することで，ハードウェアの小規模化を図る．パイプライン化により，各ステージは入力信号の異なる部分に対して独立に処理するので，While 内の処理は時間的並列処理される．初めに，$wav[0] - wav[p-1]$ が入力され，1st ステージで処理される．$wav[0] - wav[p-1]$ の 2nd ステージでの処理開始と同時に，$wav[p] - wav[p+1]$ が入力されて 1st ステージで処理される．残りも同様に順次入力されて処理される．k クロックで 1 サンプルを処理するので，$k + (n/p)$ クロックで n サンプルの入力信号から m サンプルの時間周波数成分を取得する．

3.2 HiFP の FPID 生成アルゴリズム

HiFP は，DWT のサブバンド分解で楽曲ファイルから知覚的特徴を取得して，4,096bit の FPID を生成する．**図 4** に，HiFP のフレームワークを示す．HiFP の入力信号は，44.1kHz-16bit-Stereo にデコードされた楽曲ファイルの左チャンネルの先頭から 2 秒後の 2.97 秒間（131,072 サンプル）である．データベースに保存する FPID が音楽 CD から生成されることを考慮して，HiFP の入力信号は音楽 CD と同様の 44.1kHz-16bit-Stereo とした．また，楽曲ファイルの先頭から数秒は無音の可能性があるため，先頭から 2 秒後の信号を利用する．DWT のサブバンド分解によって，HiFP は文献 2) と同様に入力信号から可聴域（20Hz-20kHz）内で Human Auditory System（HAS）に大きく影響する時間周波数成分（687Hz-1,378Hz）を取得する．そして，特徴抽出で時間周波数成分から知覚的特徴を抽出し，4,096bit の FPID を生成する．

図 5 に HiFP の FPID 生成アルゴリズムを示す．入力は，2.97 秒間に相当する 131,072 サンプルの PCM データである（1 行目）．2 行目は入力のサンプル数であり，3 行目は DWT のサブバンド分解関数の出力のサンプル数である．4 行目は図 2 の DWT のサブバンド分解の処理であり，HAS に関係する時間周波数成分を取得する．楽曲ファイルは 44.1kHz にデコードされるので，ナイキスト周波数は 22.05kHz である．従って，5 回目のサブバンド分解の高周波数成分（687Hz-1,378Hz）が HAS に関係する周波数に対応するため，5 回サブバンド分解を行う．低周波数成分を取得する式 (2) を 4 回行った後，高周波数成分を取得する式 (1) を行うことで，687Hz-1,378Hz の時間周波数成分を取得する．そして，知覚的特徴を抽出するために，687Hz-1,378Hz の時間周波数成分に対して

```
1. HiFP(wav[] ← PCM data (131,072sample)){
2.   n ← 131,072; /*Number of samples*/
3.   m ← 4,096; /*Number of samples of output*/
4.   Hi[], Lo[] ← DWT(wav[], n, m);
5.   for k = 0 to m − 2 do
6.     tmp_fpid = Hi[k] − Hi[k + 1];
7.     if tmp_fpid > 0 then
8.       FPID[k] = 1;
9.     else
10.      FPID[k] = 0;
11.    end if
12.  end for
13.  FPID[m − 1] = 0;
14.  return FPID; }
```

図 5 HiFP の FPID 生成アルゴリズム
Fig. 5 Algorithm for generating FPID of HiFP

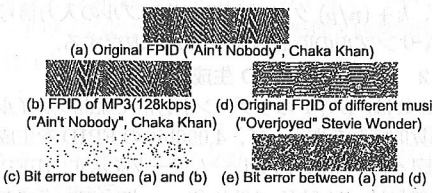

図 6 FPID と各 FPID 間の Bit Error
Fig. 6 FPIDs and bit error between FPIDs

隣接する時間の間で差分をとる（6 行目）．この差分と 0 を比較して，FPID の 1bit 分を決定する（7-11 行目）．5 回目のサブバンド分解の出力は 4,096 サンプル（=131,073 サンプル/2^5）であるため，5-12 行目の処理では 4,095bit 分の FPID しか算出されない．計算機の性質を考慮して FPID のサイズを 2 の二乗にするために，4,096bit 目は常に 0 とする（13 行目）．最終的に，4,096bit の FPID を出力する（14 行目）．

FPID 生成の重要なことは，比較する 2 楽曲が同じ楽曲ならば類似した FPID を生成し，異なる楽曲ならば異なる FPID を生成することである．**図 6** に，HiFP から生成された FPID（(a), (b), (d)）と，2 つの FPID 間の Bit Error（(c), (e)）を示す．(a) と (b) は同一楽曲である．(a) は WAVE ファイル中の PCM データから生成されたオリジナル FPID であり，(b) は 128Kbps の MP3 へエンコードした後にデコードして得た PCM データから生成された FPID である．(a) と (d) は相違楽曲である．(d) は (a) と同様に WAVE ファイル中の PCM データから生成されたオリジナル FPID である．(a), (b), (d) の黒の部分は FPID の 1bit が 0 であり，白の部分は 1 である．2 つの FPID の比較には，文献 2) と同様にハミング距離を採用した．(c), (e) の黒は 2 つの FPID 間でビット値が異なること示し，白はビット値が同じであることを示す．(c) のように，(a) と (b) はエンコードによる

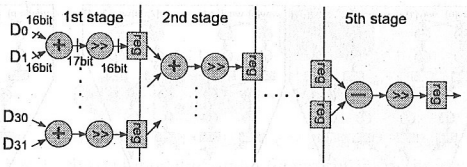

図 7 DWT 回路
Fig. 7 DWT circuitry

音質の劣化により同一楽曲であっても全く同じではない．従って，一度エンコードした楽曲を同一楽曲として識別するには，ある程度の Bit Error を考慮する必要がある．逆に，相違楽曲の FPID 間では，(e) のように Bit Error が多いことが分かる．これらのことから，2 つの FPID 間の Bit Error Rate が低ければ同一楽曲であり，高ければ相違楽曲と識別できる．

3.3 HiFP のハードウェア実装

HiFP は DWT 回路，特徴抽出回路と FPID 形成回路の 3 つで構成する．HiFP の入力は，131,072 サンプルの符号付き 16bit 整数の PCM データである．クロック毎に 32 サンプルが同時に HiFP へ転送される．入力開始から 39 クロックのレイテンシ後，FPID の 32bit 分が HiFP から 32 クロック毎出力される．最終的に，4,103 クロック（= 39 クロック + 32 クロック × 127 回）で 4,096bit の FPID が HiFP から出力される．

3.3.1 DWT 回路

図 7 に，DWT 回路の構成を示す．DWT 回路は，バイナリツリー形の 5 ステージ・パイプラインで構成されている．クロック毎に 32 サンプル（HiFP の入力）が同時に DWT 回路への転送され，パイプライン処理される．DWT 回路の出力は符号付き 16bit であり，5 クロックのレイテンシ後，クロック毎に出力される．この出力は，特徴抽出回路へ転送される．

HAS に関係する周波数成分を取得するために，HiFP の DWT のサブバンド分解は，1-4 回目のサブバンド分解で低周波数成分を取得し，5 回目で高周波数成分を取得する．従って，1-4 回目は式 (2) の計算を行い，5 回目は式 (1) の計算を実行する．1-4 回目のサブバンド分解に対応する 1-4 ステージ目は，符号付き 16bit 加算器と 2 による除算に相当する右 1bit シフトの処理で構成されている．各ステージにおいて空間的並列処理を行う．5 回目のサブバンド分解に対応する 5 ステージ目は，符号付き 16bit 減算器と右 1bit シフト処理が各 1 つで構成されている．また，各ステージ間には，16bit レジスタが配置されている．各ステージにおける加減算後のシフト処理は，符号付き 16bit 同士の加減算の結果である符号付き 17bit の上位 16bit を取る．従って，加減算器の結果に対して従来の 1bit

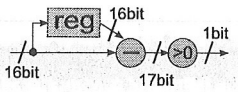

図 8　特徴抽出回路
Fig. 8　Feature extraction circuitry

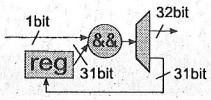

図 9　FPID 形成回路
Fig. 9　FPID formation circuitry

シフトを行っていない．

3.3.2　特徴抽出回路

特徴抽出回路は，FPID の 1bit 分を生成するために DWT 回路から転送されたデータの差分を取り，0 と比較を行う．**図 8** に，特徴抽出回路の構成を示す．特徴抽出回路は 16bit 減算器，比較器と 16bit レジスタで構成されている．知覚的特徴の抽出は DWT の結果を隣り合う時間の間で差を取る（図 5 の 6 行目）．従って，クロック毎に DWT 回路から転送されてくる符号付き 16bit のデータをレジスタに保存し，次のクロックで転送されてくるデータと減算器で差を取る．そして，図 5 の 7-11 行目のように FPID の 1bit 分を取得するために比較器で減算器からのデータと 0 の比較し，FPID の 1bit 分を FPID 形成回路へ転送する．

3.3.3　FPID 形成回路

特徴抽出回路は FPID の 1bit 分を出力するので，楽曲識別を早急に開始するには 1bit ずつ FPID を比較をすればよい．しかし，ソフトウェアで楽曲識別を行う場合，特徴抽出回路からソフトウェアへ FPID を転送する必要がある．計算機の性質を考慮すると 1bit 毎の転送より 32bit 毎の転送の方が効率が良い．このことから，FPID 形成回路は特徴抽出回路から転送されてくる FPID の 1bit を保存して，FPID の 32bit 分への形成を行う．ハードウェアで楽曲識別を行う場合は，FPID 形成回路は必要ない．

図 9 に，FPID 形成回路の構成を示す．FPID 形成回路は 1bit と 31bit を連結する機構，32bit レジスタと 2 出力デマルチプレクサで構成されている．特徴抽出回路から転送されてきた FPID の 1bit とこれまでに転送されてきた FPID を連結し，FPID の 32bit 分が完成したら出力を行う．従って，HiFP のレイテンシである 39 クロック後，32 クロック毎に FPID の 32bit 分を出力する．32bit 分を形成できてない場合は，31bit 以下分の FPID をレジスタに保存する．

4.　楽曲識別検証とハードウェア化の評価

本章では，HiFP の楽曲識別の検証と FPGA によるハードウェア化の評価を行う．

表 1　圧縮フォーマット
Table 1　Music format

Format	Description	Index
MP3	32Kbps	M32
	128Kbps	M128
	192Kbps	M192
Real	20Kbps	R20
Resampling	44.1kHz→22.05kHz→44.1kHz	RS
WMA	20Kbps	W20
	48Kbps	W48
WAVE	No	ORG

表 2　エンコーダとデコーダ
Table 2　Encoder and decoder

Format	Type	Software
MP3	Encoder	lame 3.98.2[9]
	Dncoder	lame 3.98.2[9]
Real Media	Encoder	RealProducer Basic 11[10]
	Decoder	dBpoweramp Music Converter[11]
Resampling	Encoder	Lilith 0.991b[12]
	Dncoder	Lilith 0.991b[12]
WMA	Encoder	Lilith 0.991b[12]
	Dncoder	Lilith 0.991b[12]

4.1　HiFP の楽曲識別検証

オーディオフィンガープリンティング技術（AFT）は以下の 4 つ特性が要求される[2]．

- ロバスト性：エンコードなどによる音質の劣化が生じても同一楽曲と識別する
- 識別の信頼性：多量の楽曲の中から同一楽曲を発見する
- 入力データ量　：楽曲の中の最小限の長さで楽曲を識別する
- FPID サイズ：データベースのことを考慮して，FPID のサイズはより小さくする

本節では，HiFP のロバスト性と識別の信頼性の検証，入力データ量と FPID サイズに関しての議論を行う．

4.1.1　検証条件

HiFP の検証では 200 楽曲を利用した．サウンドや演奏形態は楽曲のジャンルにより異なるので，様々なジャンルの楽曲を検証に使用した．一般的に，インターネット上の楽曲転送は，ファイルサイズを削減するために圧縮フォーマットが使用されるので，音質に劣化が生じる．従って，各圧縮フォーマットへのエンコードによる音質の劣化があっても楽曲を識別できる必要があるため，様々な圧縮フォーマットに関して検証を行った．**表 1** に，本稿の検証で利用した圧縮フォーマットを示す．Index は，各圧縮フォーマットの簡略名である．**表 2** に，使用したエンコーダとデコーダを示す．各圧縮

表 3　オリジナル FPID と各圧縮フォーマットから生成された FPID の平均 BER(%)
Table 3　Average of BER between original FPID and FPID generated from each format (%)

Format	HiFP (2.97sec)	HiFP (5.94sec)	Haitsma
M32	8.2	8.2	15.9
M128	5.2	5.2	9.8
M192	1.8	1.8	3.8
R20	15.1	15.8	19.7
RS	0.3	0.3	0.8
W20	6.6	6.6	9.2
W48	5.5	5.5	7.9

フォーマットは，リファレンスとなる WAVE フォーマットを入力として各エンコーダで生成した．各デコーダにより 44.1kHz-16bit-Stereo の WAVE フォーマットにデコードし，先頭から 2 秒後から 2.97 秒間の左チャネルの PCM データ（wav[2, 2.97]．ここで，wav[スタート，入力データ量] である）を取得して FPID を生成した．入力データ量について議論するために先頭から 2 秒後から 5.94 秒間（wav[2, 5.94]）の FPID を用いた．5.94 秒間の FPID 生成は，図 5 の 2 行目の n に 262,144 を，3 行目の m に 8,192 を代入して生成された．従って，FPID サイズは，Haitsma ら[2] と同様の 8,192bit である．

楽曲の識別は，リファレンスとなる WAVE フォーマットから生成したオリジナル FPID と各楽曲の各圧縮フォーマットから生成した FPID を比較して算出した BER で行う．式 (3) に，BER の計算式を示す．

$$BER(M_1, f_1, M_2, f_2) = \left(\frac{1}{4096} \sum_{i=0}^{4096-1} (FPID(M_1, f_1)[i] \oplus FPID(M_2, f_2)[i]) \right) \times 100 \quad (3)$$

ここで，M は楽曲ナンバー（1-200）であり，f は圧縮フォーマット名（M32, M128, M192, R20, RS, W20, W48, ORG）である．例えば，$FPID(1, M32)$ は楽曲ナンバー 1 の 32Kbps の MP3 から生成した FPID を表す．i は FPID のビット数である．$FPID(M_1, f_1)[i] \oplus FPID(M_2, f_2)[i]$ はハミング距離に相当する．BER が低ければ同一楽曲であり，高ければ相違楽曲である．

4.1.2　ロバスト性

同一楽曲であっても，図 6(c) のように音質に劣化によりオリジナル FPID と各圧縮フォーマットの FPID は全く同様でないが，知覚的特徴により FPID を生成しているので非常に類似している．しかし，音質に大きな劣化がある場合，オリジナルと各圧縮フォーマットの FPID 間の BER は高くなる．そのため，音質の大きな劣化に対してもより低い BER となる FPID を生成することが重要である．**表 3** に，HiFP と Haitsma ら[2] の手法による 200 楽曲分の各圧縮フォーマットの平均 BER を示す．各圧縮フォーマットの平均 BER は，式 (4) で求めた．f は表 3 の Format である．

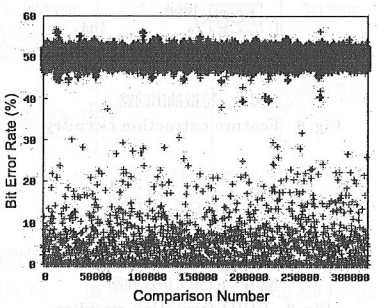

図 10　HiFP(2.97 秒) の BER
Fig. 10　BER of HiFP (2.97sec)

$$BER = \left(\frac{1}{200} \sum_{j=1}^{200} (BER(M_j, f, M_j, ORG)) \right) \quad (4)$$

Haitsma らの手法と比較して，HiFP は全圧縮フォーマットにおいて BER が低いことが分かる．このことから，HiFP は音質の劣化に対してのロバスト性が高いと言える．MP3 と WMA におけるビットレート間の比較では，ビットレートを低下すると音質の劣化が大きくなるため BER も高くなってしまっている．また，入力データ量の比較では，2.97 秒と 5.94 秒間の BER は Real Media 以外では同等の BER である．従って，入力データ量を拡大させても大幅にロバスト性は向上しないことが言える．

4.1.3　識別の信頼性

AFT において識別の信頼性は最も重要な要素である．識別の失敗率が高い AFT は信頼性が低いため利用できない．**図 10** に入力データ量 2.97 秒の HiFP による 320,000 パターン（=200music×200music×8format（M32, M128, M192, R20, RS, W20, W48, ORG））の BER 示す．2 つの FPID が相違楽曲である場合，図 6(e) のように Bit Error が多くなるため，BER は高くなる．図 10 中の 50% 前後にプロットされている FPID の組み合わせは相違楽曲である．逆に，同一楽曲である場合は BER が低くなるので，低い BER にプロットされている FPID の組み合わせが同一楽曲である．相違楽曲は高い BER に，同一楽曲は低い BER になる．すなわち，楽曲を識別するには，BER を用いて判別する閾値を決定すればよい．

図 11 に入力データ量 2.97 秒と 5.94 秒の BER から算出された同一楽曲と相違楽曲の正規分布を示す．文献 3) で述べられているように，2 つの FPID が相

表 4　閾値と識別失敗数の比較
Table 4　Comparison of the threshold and the number of failures

Method	Input data size	Threshold (%)	Number of failures in the same music data	Number of failures in the different music data
HiFP	2.97sec	44.3	1/320,000	36/320,000
	5.94sec	44.3	1/320,000	35/320,000
Haitsma	3.3sec	35.0	6/320,000	625/320,000

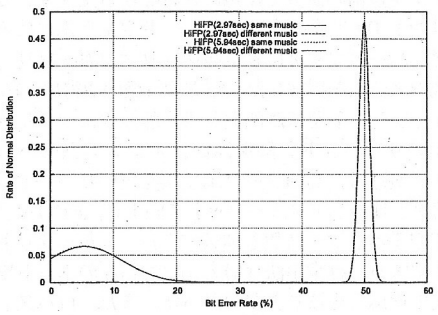

図 11　正規分布による入力データ量間の BER の比較
Fig. 11　Comparison of BER between input data size using normal distribution

違楽曲である場合は，両入力データ量共に BER は約 50% を中心に密集していることが分かる．同一楽曲の場合は，両入力データ量共に BER は約 6% を中心に密集している．入力データ量が 2.97 秒と 5.94 秒の正規分布はほぼ同様でなので，入力データ量を拡大させても同一楽曲間と相違楽曲間の BER はほぼ変化しないことが言える．

データベース内に 200 楽曲の FPID があり，200 楽曲 8 圧縮フォーマット（M32，M128，M192，R20，RS，W20，W48，ORG）の 1,600 パターンの楽曲識別すると想定して検証を行った．表 4 に入力データ量間の閾値と楽曲識別失敗数の比較を示す．閾値は同一楽曲と相違楽曲の正規分布の交差する BER としたところ，各入力データ量ともに 44.3% が閾値となった．2 つの FPID の BER がこの閾値より低いければ同一楽曲であり，高ければ相違楽曲である．同一楽曲の識別失敗数とは，2 つの FPID は同一楽曲のものであるが，BER が閾値 44.3% より高いため相違楽曲として識別された数である．逆に，相違楽曲の識別失敗数とは，2 つの FPID は相違楽曲のものであるが，BER が閾値 44.3% より低いため同一楽曲として識別された数である．同一楽曲の識別失敗数は，両入力データ量共に 1 パターンだけ同一楽曲としての識別に失敗した．相違楽曲の識別失敗数は，入力データ量が 2.97 秒では 35 パターン，5.94 秒では 36 パターン相違楽曲としての識別に失敗した．BER に加えて識別面においても，両入力データ量間で傾向の変化はなかった．さらに，Haitsma ら[2] の手法と比較して，HiFP の識別失敗数を削減できた．相違楽曲の識別失敗数では約 11% まで削減できた．このことから，HiFP の FPID 生成アルゴリズムの識別の信頼性は高いと言える．

4.1.4　FPID サイズと入力データ量

データベースは多量の FPID を保存するため，1 楽曲当たりの FPID サイズが重要になる．ただし，文献 1) で述べられているように，過度に小さい FPID はロバスト性と識別の信頼性を失う可能性がある．従って，理想的な FPID は，小さいサイズでロバスト性と識別の信頼性が高いものである．FPID サイズが 8192bit の Haitsma ら[2] と比較して，HiFP は表 3 と表 4 で示したようにロバスト性と識別の信頼性を低下させることなく FPID サイズを 50% 削減できる．また，入力データ量を 5.94 秒に拡大して 8,192bit の FPID にしても，ロバスト性と識別失敗数はほぼ変化しないので，入力データ量は HiFP が採用している 2.97 秒間で十分と言える．

4.2　ハードウェア化の評価

4.2.1　評価条件

FPGA でハードウェア化した HiFP を評価するために実行時間，スループットと回路規模の比較を行った．回路規模は，ISE9.2_04i_PR11 で XC2VP7-FP456-5 を対象として測定した．ハードウェアの実行時間は，ModelSim SE 6.2e で測定したクロック数と論理合成後の動作周波数から算出した．ソフトウェアは C 言語で実装し，gcc 4.1.2 でオプション O3 を付属してコンパイルした．ソフトウェア実行の測定環境は，Linux PC（Pentium4 2.8GHz，メモリ 1GB）である．比較対象とする磯永ら[8] は，Synplify Pro7.3.4 で XC2V6000-FG1146-4 を対象として測定した．磯永らと HiFP の実装で使用した論理合成ツールは同一でないため，磯永らとの比較の数値は参考値である．

4.2.2　実行時間とスループット比較

表 5 に，動作周波数，実行時間とスループットを示す．式 (5) と式 (6) に，楽曲スループットと Bit スループットの計算式をそれぞれ示す．

$$MusicThroughput = \frac{1,000,000}{e} \quad (5)$$

$$BitThroughput = b \times s \times MusicThroughput \quad (6)$$

ここで，e は実行時間であり，単位は us である．また，s は入力信号のサンプル数であり，HiFP が 131,072 サンプル，磯永らは 146,944 サンプルである．b は 1

表 5　動作周波数，実行時間とスループットの比較
Table 5　Comparison of the frequency, the execution time and the throughput

Method	Frequecy (MHz)	Execution Time (us)	Music Throughput (music/sec)	Bit Throughput (Gbps)
Isonaga[8]	86.9	750.5	1,332.4	3.1
HiFP (SW)	2,800.0	10,600.0	94.3	0.2
HiFP (HW)	137.5	29.8	33,557.0	70.3

表 6　使用リソース数の比較
Table 6　Comparison of the required resources

Method	Slice	Multiplier	Block RAM
Isonaga[8]	998	5	3
HiFP	856	0	0

サンプルのビット数であり，HiFP と磯永ら共に 16bit である．FFT を利用する磯永ら[8] と比較して，HiFP の動作周波数は 50.6MHz 向上し，実行時間は 3.9% まで短縮できた．HiFP のソフトウェア処理との比較では，実行時間を 0.2% まで短縮できた．楽曲スループットでは磯永ら[8] と比較して約 29 倍，HiFP のソフトウェア処理と比較して約 355 倍向上した．また，Bit スループットも磯永ら[8] と比較して約 22 倍，HiFP のソフトウェア処理と比較して約 351 倍向上した．

現在の規格で最も高速な 10Gbps のネットワークにおいて，ブロードキャストモニタリングなどのインターネットアプリケーションでオーディオフィンガープリンティング技術を用いる場合，3.1Gbps のネットワークまでしか対応しない磯永らのハードウェアシステムでは，4 並列で実行する必要がある．一方，HiFP は 70.3Gbps のネットワークまで適応できるので，容易に 10Gpbs のネットワーク上で適応できる．規格化が行われている 40Gbps や 100Gbps のネットワークにおいても，HiFP は容易に適応することができる．100Gbps のネットワークでは，HiFP を 2 並列で処理させることで容易に適応できる．

4.2.3　使用リソースの比較

表 6 に，各手法における FPGA 実装の使用リソース数の比較を示す．FFT を利用する磯永ら[8] と比較して，DWT を利用する HiFP は使用する Slice，18x18 乗算器，Block RAM の数を削減できた．磯永ら[8] は，FFT に Xilix 社の IP コアを使用しているため，18x18 乗算器を 2 つと Block RAM を 3 つ使用し，FFT から出力される実数部と虚数部の 2 乗にも各 1 つの 18x18 乗算器を使用してる．しかし，HiFP は加減算器のみで構成されているので，18x18 乗算器と Block RAM を使用する必要がない．また，他の回路も加算器，レジスタなど比較的 Slice を使用しない構成なので，Slice 数を約 13.3% 削減できた．

5. ま と め

本稿では，これまでのオーディオフィンガープリンティング技術の研究で焦点にされなかったフィンガープリント ID 生成の高速化とハードウェア化に焦点を置き，ハードウェアによる高速なフィンガープリント生成システムを提案した．知覚的特徴の取得に，Haar ウェーブレットを基底関数とする離散ウェーブレット変換のサブバンド分解を用いた本システムのハードウェア処理は，高速フーリエ変換を用いたハードウェア処理と比較してスループットを向上し，FPGA 実装で使用するリソース数も削減できた．スループットの向上により，提案手法は最大 70.3Gbps のネットワーク上で適応できることが分かった．また，ロバスト性と識別の信頼性に関しても，本システムは既存のシステムと比較して向上できた．

参 考 文 献

1) Cano, P., et al.: A Review of Audio Fingerprinting, *The Journal of VLSI Signal Processing*, Vol. 41, No. 3, pp. 271–284 (2005).
2) Haitsma, J. and Kalker, T.: A highly robust audio fingerprinting system, *Proc. ISMIR*, pp. 107–115 (2002).
3) Lu, C.-S.: Audio fingerprinting based on analyzing time-frequency localization of signals, *Proc. MMSP*, pp. 174–177 (2002).
4) Baluja, S. and Covell, M.: Content Fingerprinting Using Wavelets, *Proc. CVMP*, pp. 198–207 (2006).
5) Neuschmied, H., Mayer, H. and Batlle, E.: Content-based identification of audio titles on the Internet, *Proc. International Conference on Web Delivering of Music*, pp. 96–100 (2001).
6) Kalker, T., Epema, D., Hartel, P., Lagendijk, R. and VanSteen, M.: Music2Share - copyright-compliant music sharing in P2P systems, *Proc. the IEEE*, Vol. 92, No. 6, pp. 961–970 (2004).
7) Ghouti, L. and Bouridane, A.: A Robust Perceptual Audio Hashing using Balanced Multiwavelets, *Proc. ICASSP*, Vol.5, pp.V–V (2006).
8) 磯永久史, 井口寧: FPGA を利用した高速オーディオフィンガープリントシステムの構築, 信学技報, RECONF2005-77, pp. 1–6 (2006).
9) http://lame.sourceforge.net: (2009).
10) http://www.realnetworks.com: (2009).
11) http://www.dbpoweramp.com: (2009).
12) http://www.project9k.jp: (2009).