FPGA による電子指紋検索の高速化事例

井口 寧[†1]　　福田 真啓[†1†2]

FPGA による処理の高速化事例として音楽電子指紋検索の FPGA 実装の例を紹介する．FPGA に適する利用分野として，積極的に再構成可能な特性を利用するケース，需要が少量である場合に設計コストの低減を目的とするケース，および利用する演算や処理構造が単純であり，専用回路とすることによって大きな性能向上が図れる場合などがある．本報告では，演算を加減算など単純な回路とすることによって音楽電子指紋計算を高速化し，検索ではハミング距離が最短な指紋を探索する多段の回路とすることによって高速化した事例を示す．複雑な処理機構は不要とし，問題の特性を利用しながらいかに処理を高速化するかがポイントである．幾つかの高速化のアプローチを踏まえつつ，取り組みを紹介する．

High Speed Audio Fingerprint Detection and Search Using a FPGA

YASUSHI INOGUCHI[†1]　MASAHIRO FUKUDA[†2]

This paper introduces an implementation of audio fingerprint detection using a FPGA as an acceleration example. FPGA is effective when the circuit requires reconfiguration while the process, design cost reduction is required for a small number of products, and significant performance improvement can be expected when the algorithm uses only simple calculation.　This report shows an example of acceleration simplifying calculation such as fixed-point addition and subtraction, and introducing multistage hash searching. Adapting these schemes, processing speed is accelerated over than 1,000 times faster than the original algorithm executed by software.

1. はじめに

FPGA による処理の高速化事例として音楽電子指紋検索の FPGA 実装の例を紹介する．FPGA に適する利用分野として，積極的に再構成可能な特性を利用するケース，需要が少量である場合に設計コストの低減を目的とするケース，および利用する演算や処理構造が単純であり，専用回路とすることによって大きな性能向上が図れる場合などがある．本報告では，FPGA 向きの演算回路とすることで，大幅に処理速度を向上できた例として，音楽電子指紋の高速検索の事例を紹介する．音楽電子指紋は加減算など単純な演算で指紋を計算でき，検索ではハミング距離が最短な指紋を探索する動作となる．複雑な処理機構は不要であり，問題の特性を利用しながら，いかに処理を高速化するかがポイントとなる．幾つかの高速化のアプローチを踏まえつつ，取り組みを紹介する．

2. 音楽電子指紋とは

近年，Winny や WinMX など，インターネットでデジタル著作物を交換するソフトウェア(ファイル交換ソフト)が広く用いられ，デジタル著作物の著作権侵害が大きな社会問題となっている．このデジタル著作物を保護するための一手法として，電子指紋技術[1]が注目されている．電子指紋は，音楽や画像などのデジタル著作物の周波数帯域ごとのエネルギー遷移など，著作物ごとの特徴量を計算し，数百ビットのコンパクトな表現で著作物を特定する技術であり，著作物のコピーコントロールを成し得る重要な技術である．ところで，電子指紋検出は従来ソフトウェアで行われており，その場合の検出速度は高々数 Mbps 程度である．この程度の検出速度では，家庭に広く普及して音楽いる 1Gbps でのネットワークに適用することは全く不可能である．そこで本研究では，電子指紋検出・検索をハードウェア化することによって，検出速度を向上させ，ネットワークを介して流通するデジタル著作物を実時間で検出・特定しようと試みた．図 1 に本機構の適用例を示す．電子指紋(FP)が高速に検出できれば，ルータなどのネットワーク機器に検出機構を埋め込むことによって，コンテンツの流通をコントロールすることができる．具体的には，意図しないコンテンツの流出防止や，コンテンツごとのダウンロード数の計量によるライセンシングなどが考えられる．

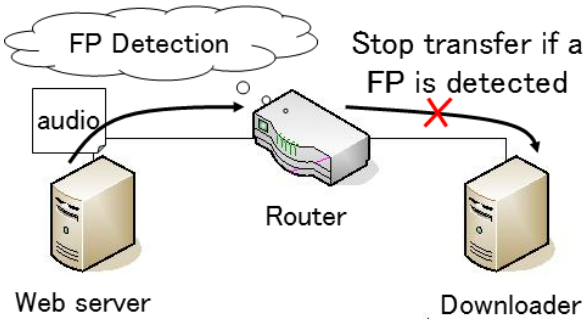


図 1 電子指紋の実社会での適用例

Figure 1 An example of use of audio fingerprint.

実際の音楽電子指紋は，音楽の特徴量を計算して計算される．音楽ファイルそのものは， mp3 圧縮されていたとしても，数分の楽曲で数 MByte の容量になるところ，電子

†1 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology
†2 石川工業高等専門学校
National Institute of Technology, Ishikawa College

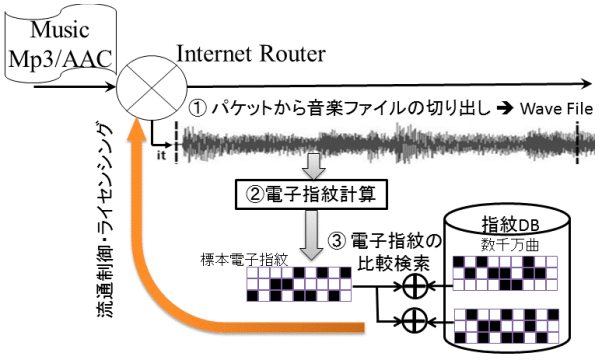


図 2 音楽ファイルから指紋の計算・比較

Figure 2　Audio fingerprint calculation and comparison.

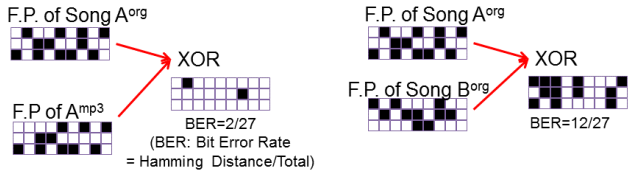


図 3 非可逆圧縮によるビットエラーレートの様子

Figure 3　Bit error rate caused by lossy compression.

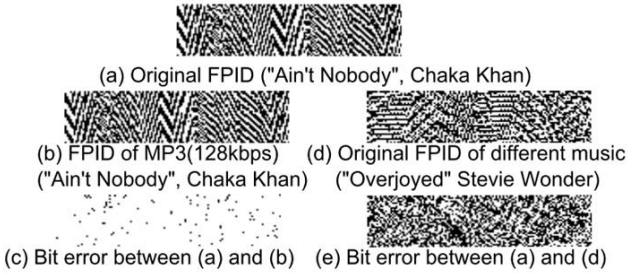


図 4 実際の楽曲から計算された音楽電子指紋

Figure 4　Audio Fingerprint calculated by an exist song.

指紋の形式にすることによって, 数百 Byte のコンパクトな形式に変換することができる．このことによって，比較段階では楽曲全体の数 MB を比較することなく数百 Byte の比較で楽曲を特定することが可能になり，速やかな特定ができるのである．図 2 にその様子を示す．

音楽電子指紋検索で問題となるのは, 楽曲が mp3 や AAC, WMA などによって非可逆圧縮されているため，その情報を元に計算される電子指紋も，たとえ元が同じ楽曲だとしても，完全一致とならないことである．この様子を図 3 に示す．左側の例は，同じ楽曲 Song A^{org} から計算された電子指紋 F.P. of Song A^{org} と，mp3 圧縮し，復号された楽曲から計算された電子指紋 F.P. of Song A^{mp3} を比較した場合である．元の楽曲が同一であったとしても，一旦 mp3 圧縮すると非可逆圧縮であるため，それを伸張しても全く同一の PCM 信号にはならない．そこから計算される電子指紋も，全く同一にはならない．但し特徴量を用いて計算しているため, 良く似たビット列となる．この F.P. of Song A^{org} と F.P. of Song A^{mp3} の比較，つまりハミング距離を測ると，僅かのビット配列の違いとなって表現される．図 3 左の例では BER (Bit Error Rate)が 2/27 となっている．一方，元の楽曲が異なっている場合，特徴も全く異なるので，2 つの異なる楽曲を基にした電子指紋 F.P. of Song A^{org} と F.P. of Song B^{org} の排他的論理和を取ると全く異なったビット配列となっており，BER は 0.5 に近くなる．図 4 は実際の楽曲から計算された音楽電子指紋の比較の様子である．

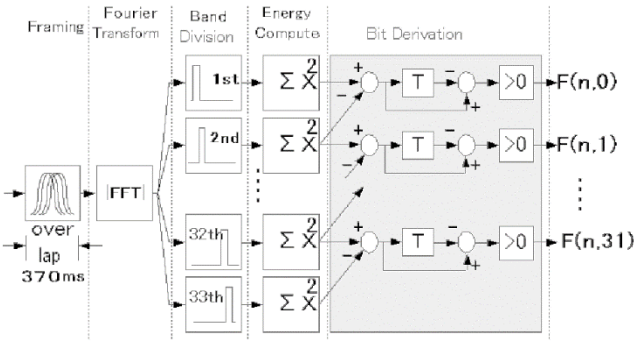


図 5 Haitsma らによる音楽電子指紋の計算

Figure 5　Audio Fingerprint Algorithm proposed by Haitsma.

3. 音楽電子指紋検出の FPGA 実装

(1) 関連研究

音楽電子指紋については，その有用性のため多くの方法が提案され研究されてきた．古典的手法としては Haitsma らによる FFT ベースの手法がある[1]．この処理の概略を図 5 に示す．PCM 展開された音楽ファイルは 370ms (44.1KHz サンプリングとして 16Ki サンプル)ごとにオーバーラップしながら 256 個のフレームとして切り出される．フレームごとに 33 点の FFT 計算がなされ，バンドごとのエネルギーが計算される．隣接バンドごとのエネルギー差の正負によって 1/0 のビットに対応づけられ，フレームごとに 32 ビットの指紋が計算される. これを 256 フレームに適用し，最終的に 8,192bit の電子指紋を得ている．他にも精度や処理速度，適用対象の点から様々なアルゴリズムが提案されているが，Haitsma らの方法は簡単な実装でありながら比較的精度も高い手法である．

(2) Haitsma らの方法の FPGA 実装

最初に Haitsma らの方法を直接 FPGA 実装することを試みた[2]．アルゴリズムはオリジナルのままで，VHDL にて FPGA 実装を行った．表 1 に実装環境，表 2 に速度評価の結果を示す．Radix-4 の FFT を用いた場合，FPGA 実装はソフトウェア処理に比べておよそ 10 倍の速度向上が得られた．

ルータに適用することを考える場合，音楽ファイル 1 つの通過に対しての処理時間と考えることができる．5 分程度の楽曲を 128Kbps の mp3 ファイルにすると，大体 4MiByte であるので，これを 1 楽曲のファイルサイズと仮

FPGA Board	Hitachi Logic Bench
HW/SW Interface	Hitachi Virtual Turbo PCI
FPGA chip	XC2V6000FG1156-4×4
Host PC CPU	AMD Athron 3.6GHz

表 1 Haitsma 法の FPGA 実装環境

Table 1　FPGA implementation environment for Haitsma.

	Time to AFP calculation	Search time	Performance ratio
Software	3,353ms	-	1.00
FPGA radix-2 FFT	650.24ms	0.543ms	5.15
FPGA radox-4 FFT	315.53ms	0.533ms	10.66

表 2 Haitsma 法の処理速度

Table 2　Processing speed of Haitsma method.

	4-input LUTs	Critical Path	Max Freq.
FP extract	3.0%	11.5ms	87.0MHz
Searcher (40 parallelized)	6.4%	26.6ms	40.0MHz
Total (40 parallelized)	9.9%	26.6ms	37.6MHz

表 3 FPGA 資源消費と動作周波数

Table 3　ULT Usage and critical path length.

定すると，この場合の処理速度は，$\frac{4MiByte\times 8}{315.53ms+0.533ms}=$ 101.2 Mibpsとなる．同様にソフトウェア実装では 9.54Mibps であり，全く Gbps クラスのネットワークには適さないことが分かる．FPGA 実装することによって，10Mbps 以下の処理性能だったところが 100Mbps 以上の処理性能となり，ルータでの適用が視野に入ってくるのである．表 3 に Haitsma らの方法を実装した時の資源消費量と最高動作周波数を示す．Haitsma らの方法では複数のフレームの処理が全く独立しているので，並列動作が可能である．表 3 に示される通り，40 並列だとしても，LUT 消費量は 10%程度である．

(3) Haar-Wavelet 変換に基づく電子指紋計算

Haitsma の方法で大きなボトルネックは，FFT 部分である. FFT を使うために, 浮動小数点演算部分が必須となり，処理速度の点でも資源消費量の点でも，大きく FPGA 実装による高速化の大きな障害となっている．

一方で Haar Wavelet Transform(HWT) は FFT に近い性質

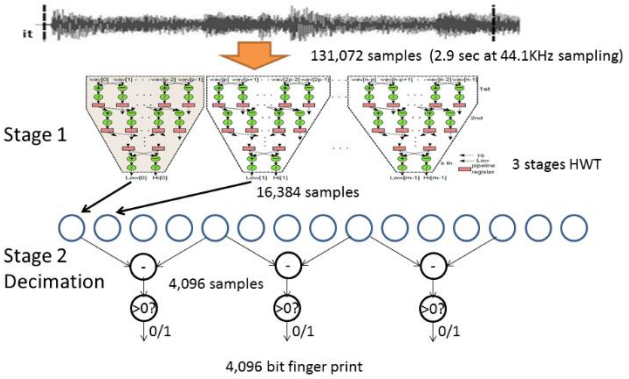


図 6 HiFP2.0 の概要

Figure 6　Outline of HiFP2.0.

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

表 4 精度比較のための圧縮手法

Table 4　Lossly compression to compare accuracy.

を持ちながら，演算はずっと簡単である．必要な演算はもっぱら加減算だけであり，かつ固定小数点で十分である．FPGA 実装を考えた場合，演算器の要求資源量が小さく Critical Path も短いことが期待できる. このことは FPGA 実FPGA 実装を考えた場合，演算器の要求資源量が小さく Critical Path も短いことが期待できる. このことは FPGA 実装で高速化する場合にとても重要な点である. そこで HWT に基づいた音楽電子指紋の計算法として HiFP 手法を提案した[3]．HiFP 法はフレームオーバーラップの方式によって 2 つの実装があるが，本稿では HiFP2.0 について紹介する．　図 6 に HiFP2.0 の概要を示す．Haitsma らの方法と同じく PCM 展開された音楽ファイルは 16Ki サンプルごとにフレームとして分割される．それぞれのフレームは 3 段の HWT を通して 4Ki サンプルを得る．このうち 4 サンプルごとに隣接するフレームとの差分を比較し，その正負によって 1/0 のビット列を得る．このビット列は 4,096 ビットの指紋となる．

HiFP2.0 は Haitsma らの方法をベースにしているが, 基本演算を FFT から HWT に変更しているため，精度の評価が不可欠である．表 4 に評価対象となる非可逆圧縮法とその指標を示した．ここでは，元の WAVE フォーマットの音楽

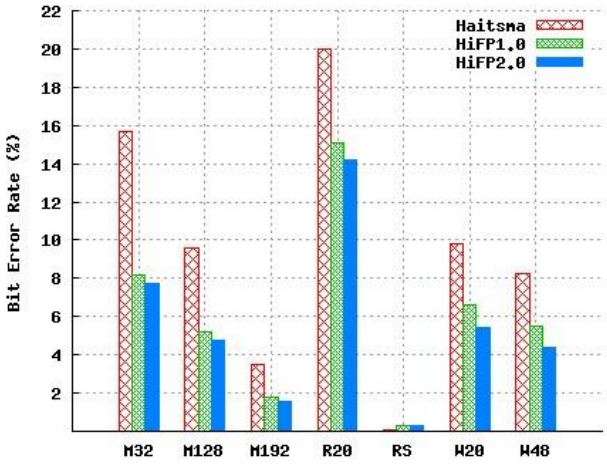


図 7 HiFP および Haitsma らの方法での識別精度

Figure 7　Accuracy of HiFP and Haitsma method.

Method	**Execution Time**	**Bit Throughput**
Haitsma (FFT Base)	1,701.741ms	0.0188Gbps
HiFP1.0 (5-level subband decomposition)	0.274ms	116.8 Gbps
HiFP2.0 (3-level subband decomposition)	0.095ms	336.8 Gbps

表 5 HiFP 法と Haitsma 法の処理速度

Table 5　Lossly compression to compare accuracy.

ファイルを一旦所定の非可逆圧縮法で圧縮したあと，それを元の WAVE ファイルに戻し，元の音楽ファイルから計算紋の BER を計算した．BER が小さいほど，非可逆圧縮があったとしても，元の楽曲から得られる電子指紋に近い指紋が得られるので，識別精度が高いと言える．圧縮ソフトウェアは，MP3 圧縮では lame 3.98.2，Real Media では ReadProducer Basic11 および dBpoweramp Music Converter，Resampling および WMA (Windows Media Audio)では Lilith 0.991b を用いた．図 7 に精度評価の結果を示す．Resampling (RS)を除く圧縮方法で, HiFP2.0 が Haitsma らの方法よりも良い(BER が低い)識別精度を有していることが分かる．RS の場合は Haitsma らの方法がより高い識別精度を示しているが，その差はごく僅かである．

HiFP2.0 の処理速度を表 5 に示す. Haitsma の FFT ベースの方法をそのまま FPGA 実装した場合よりも, およそ 2,000 倍弱に高速化されていることが分かる．この大きな原因は FFT による浮動小数点演算を排除できたことに起因する．336Gbps の検出速度があれば，基幹ルータにも適用でき，実用上の速度は十分であると言える．

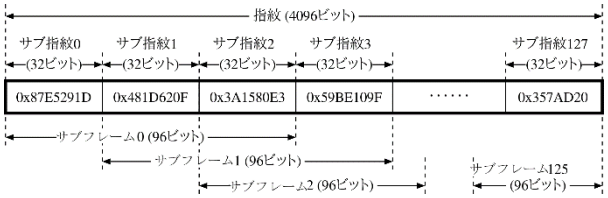


図 8 階層型 LSH のためのフレーム分割

Figure 8　Frame division for Staged LSH.

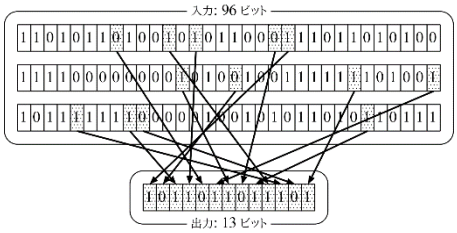


図 9 局所鋭敏性ハッシュ値の計算

Figure 9　Calculation of locality sensitive hash

(4) 電子指紋の検索

次に，電子指紋が計算できたとして，指紋をデータベースから検索し，楽曲のライセンス情報などと照合する操作が必要になる．数多くの楽曲の電子指紋を予め計算しておき，データベースに保存する．ネットワークを流通する音楽ファイルの電子指紋を計算したら，データベースから該当する電子指紋を検索し，必要なライセンス情報などを得る．ところで既に述べたように，電子指紋は 4Ki～8Ki ビットからなるビット列であり，同じ楽曲だとしても非可逆圧縮伸張過程を経た場合は全く同一のビット列にはならない．データベース検索とは，完全一致ではなくハミング距離が最小のエントリを検索することになる．完全一致検索であれば予めエントリをソートしたりハッシュ関数を用いたりすることができるが，ハミング距離最小の探索だとそのような手法を用いることができない．

単純な実装では，探索したい電子指紋(以後，クエリ指紋と呼ぶ)とデータベース中の電子指紋のハミング距離を全てのエントリについて計算し，最小のエントリを見つけることになる．しかしながら，音楽電子指紋をターゲットにした場合，既に Apple iTunes でも数千万曲が保存されており，これに相当する電子指紋のデータ量は一曲当り 4Kibit として, 例えば楽曲数が 1 億曲であれば, 約 50GB となる．楽曲がルータを通過する数十 ms の間で, 検索の度に 50GB 全てのエントリについてハミング距離を計算するのは不可能である．

そこで局所鋭敏製ハッシュを用いた階層型 LSH (locality sensitive hashing)探索を提案した[4][5]．　図 8 に示すように，階層型 LSH では 4,096 bit の指紋を 96 ビットごとのサブフレームに分割する．この中から図 9 の様に複数のビットを選び, 局所鋭敏性ハッシュ値を得る. この手法により，

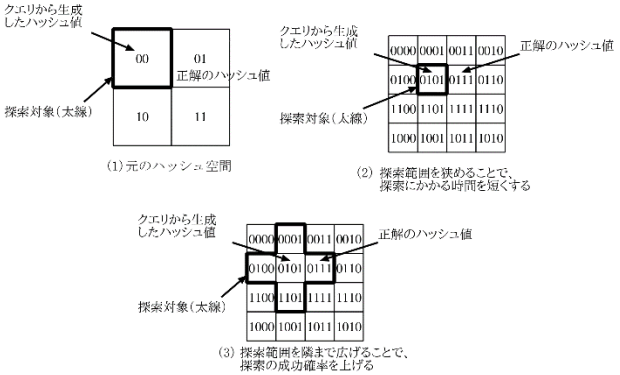


図 10 隣接 LSH の概要

Figure 10 Outline of the neighboring hash search.

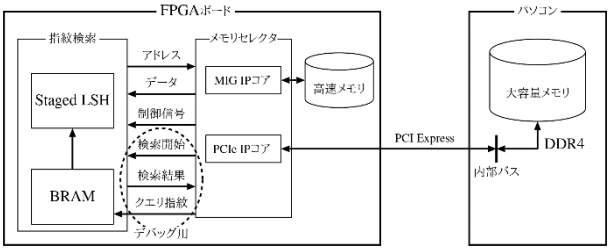


図 11 PCI-express を通じた大規模指紋検索

Figure 11 Large scale fingerprint search through PCI-express.

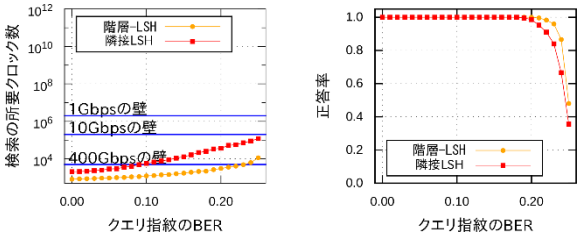
指紋データベースの容量はかなり小さく圧縮できる．更に検索範囲を小さくする手法として，隣接 LSH 法も提案した．これは図 10 に示す様に，探索範囲を小さくする代わりに，ハミング距離が 1 つ違う隣接するデータベース空間も探索する手法である．探索区間は増えるが，一つ一つの探索空間が小さくなるので，精度向上と探索速度向上の両方が期待できる．

階層型 LSH および隣接 LSH を用いたとしても，FPGA ボード上のメモリに搭載するのは困難である．そこで，頻繁に参照される指紋のみを FPGA ボード本体に格納し，その他の指紋情報をホスト PC 側のデータベースに置き，必要に応じて PCI-express を通じてホスト PC 側のメモリを参照する仕組みを構築した．この様子を図 11 に示す．この実装では，PCI-express を通じた参照を効率的に行うことが重要な解決課題となる．

これらの実装の速度評価と精度評価の結果を図 12 および図 13 に示す．図 12 には FPGA ボード上のメモリのみを用いて 300 曲を収容するデータベースをアクセスした場合の検索速度，図 13 は PCI-express を通じて一千万曲のデータベースを探索した場合の検索速度を示す．FPGA ボード上のメモリを用いる場合に比べて，PCI-express を通した検索になると，10 倍以上速度が低速になっていることが分かった．

4. FPGA によるアクセラレーションの課題

本報告に見られるように，FPGA には大きな発展の素養



(1) **高位合成環境の整備**

図 12 FPGA ボード内のメモリを用いた検索

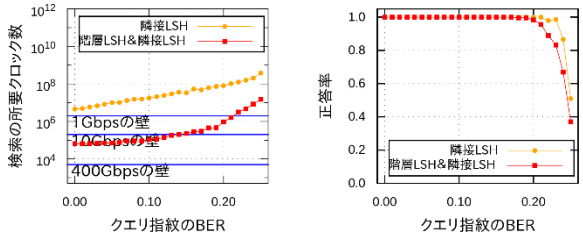


Figure 12 Audio fingerprint search using on-board memory.

図 13 PCI-express を通じた電子指紋の検索

Figure 13 Audio fingerprint search through PCI-express.

がある．演算を FPGA に最適化すれば，1,000 倍以上の高速化も可能である．ところが現実には期待されたほどの活用はなされていない．その原因について考察してみると，次のような課題が考えられる．

FPGA と並ぶアクセラレーション機構として，GPGPU (汎用グラフィックスプロセッサ)による計算が注目されている．2000 年前後に GPGPU や FPGA が次世代の計算機構として注目を集め始めた頃は，どちらも萌芽的な試みで活用状況は共に広くは無かった．しかしながら，特に 2010 年代に入り GPGPU は広く使われるようになり，例えば天河 1 号(中国)や TSUBAME 2.5 (東工大)などを始めとする大規模なスーパーコンピュータの付加機構として広く使われる様になってきている．一方で FPGA の利用は広まりつつも，広く一般のユーザーが利用できる状況にはないと言える．

この原因を考えてみると，GPGPU では早くからチップ製造元の nVIDIA 社による CUDA のサポートや OpenACC などの開発環境が整備されてきたことが上げられる．

FPGA についても以前から高位合成の取り組みが行われ，Impulse C や，近年では Vivado などの開発環境が整備されてきた．しかしながら，GPGPU の整備状況に比べるとユーザーにある程度のハードウェアの知識を要求する状況となっており，これが普及を阻んでいる一つの要因と考えられる．

GPGPU は，演算プロセッサとして見た場合，CPU に対して PCI-express を介したインタフェースとなっており，入出力設計の自由度が小さい．これに対して FPGA の利用形態は広範であり，設計の自由度が大きい．さまざまな利用

をサポートしようとするために，開発ツールでの選択肢やオプションが多くなりすぎて，計算だけ行いたいユーザーにとって難解になっているのではないかと考える．一つの解決法は，従来の広範な自由度をサポートする開発環境とは別に，PCI-express などの利用形態を絞り込んだ標準設計や標準プロトコルを用意し，これのみをターゲットとする機能限定型の開発環境を整備できれば，一般ユーザーの考慮すべきことが少なくなり，とっつき易くなるのではないかと期待している．より重要なことは，このような試みを Xilinx や Altera(現 Intel)などのチップベンダーや, 学術界の標準化団体が強力にサポートすることだと考えている．

(2) **標準化されたインタフェース**

インタフェースについても，高性能・高機能な標準化インタフェースが必要だと考える．今回の報告で見られたとおり, FPGA 自体はまだ能力的に余裕があるにも関わらず, PCI-express の I/O 性能に足を引っ張られ，性能を発揮できないケースがある．この制約は GPGPU でも同じ状況であり，特にレイテンシでの影響が大きい．このため GPGPU では例えば NV-LINK など，やはりチップベンダー主導の高速インタフェースが規格化され普及しつつある．FPGA においても UltraScale+などのインタフェースが用意されているが，汎用であり，PC との接続には特化されていない．FPGA による計算アクセラレーションの普及のためには，むしろ汎用ではなく, PC と直接高速にデータをやりとりする，限定かつ標準化されたインタフェースが必要ではないかと考える．

5. まとめ

近年の高速計算分野では，CPU の高速化だけでなく，GPGPU や FPGA などの付加計算機構によるアクセラレーションが大きな役割を持ち始めている. この背景としては, 単一 CPU の性能向上の限界や, 電力の制約などがあげられる．

計算アクセラレータとして見た場合，GPGPU と FPGA はほぼ同時に同程度のスタートであったにもかかわらず, 最近の Top500 などでは GPGPU 搭載システムの躍進など大きく差が開いている．

本報告では，音楽電子指紋を例として，FPGA 向きのアルゴリズム，つまり固定小数点演算や加減算だけの演算とすることによって，CPU に比べて 1,000 倍以上の性能向上が得られることを示した．その開発過程で，標準化および普及された開発環境の重要性を述べた．

DA シンポジウムの予稿として，今後の FPGA での開発環境の標準化および普及，機能特化に特に期待する．

謝辞 本報告の一部は科研費「ギガ帯域インターネットにおける電子指紋の超高速・高精度検出と超高速検索」の成果の一部である．関係各位に感謝する．

参考文献

1) Jaap Haitsma, Ton Klker “A Highly Robust Audio Fingerprinting System”,Proc. ISMIR 2002 3rd International Conference on Music Information Retrieval.
2) Y. Inoguchi and H. Isonaga, "High-Speed Detection System of Audio-Fingerprint with Hardware", Proc. of the International Workshop on High Performance and Highly Survivable Routers and Networks (HPSRN 2007) ISBN: 978-4-9900330-6-4, S. Horiguchi and X. Jiang 編, pp.1-8, Mar. 14, 2007
3) K. Araki, Y. Sato, V. Jain and Y. Inoguchi, “Performance Evaluation of Audio Fingerprint Generation using Haar Wavelet Transform,” Proceedings of the 2011 International Workshop on Nonlinear Circuits, Communications and Sig-nal Processing (NCSP11), pp.380-383, Tianjin SaiXiang Hotel, Tianjin, China, 2011.
4) 福田 真啓, 井口 寧, "大規模データベースの音楽指紋検索と確率的偏りを利用した性能向上", 信学技法 RECONF2015-80, Vol. 115, No. 400, pp.167-172, Jan. 21, 2016
5) M. Fukuda and Y. Inoguchi, “Probabilistic Strategies Based on Staged LSH for Speedup of Audio Fingerprint Searching with Ten Million Scale Database”, HEART2017, Jun 8, 2017.