胎児心音常時モニタリングシステムのハードウェア実装

舩越 雅[1]　大竹 哲史[2]

概要：胎児のモニタリングの手法の一つとして聴診があるが，妊婦の腹部から取得した音響信号は雑音を多く含むため，雑音除去を行う必要がある．その方法として，ウェーブレット変換とフラクタル次元解析を組み合わせた WT-FD 法が提案されている．しかし，ソフトウェアで実装した場合，心音の記録時間より処理時間の方が長く，リアルタイムモニタリングは実現出来ていない．そこで本研究では，心音検出処理の大部分を占めているフラクタル次元解析を FPGA 上のハードウェアとして実装し，処理時間をソフトウェアの 10 分の 1 に短縮した．これにより，胎児の常時リアルタイムモニタリングを実現した．

キーワード：IoT，胎児モニタリング，FPGA，ハードウェア・ソフトウェア・コデザイン，高位合成

Hardware Implementation of a System for Constant Monitoring of Fetal Heart Sounds

Abstract: Auscultation is one of the fetal monitoring methods. Since the acoustic signal acquired from the pregnant woman's abdomen contains a lot of noise, it is necessary to perform noise reduction. The WT-FD method, which combines the wavelet transform and fractal dimensional analysis, is one of such methods. However, this method is not effective for real-time monitoring because the processing time is longer than the recording time of the heart sound if it is implemented in software. In this study, fractal dimension analysis, which accounts for the majority of the heartbeat detection process, was implemented as hardware on an FPGA, reducing the processing time to one-tenth of that of software. This has realized real-time monitoring of the fetus at all times.

Keywords: IoT, fetal monitoring, FPGA, hardware-software co-design, high-level synthesis

1. はじめに

胎児は，様々なストレスを伴い原因不明の急変が起こりうる [1]．それに対して胎児モニタリングは，胎児の健康状態を把握する際の有効な手段である．現在胎児の健康状態の診療はドップラー超音波や心電図記録などの超音波を用いた装置によって行われているが，それらの装置はコストや使用方法の問題から専門家を必要とすることが多い．また，超音波エネルギーの頻繁かつ長時間の使用は，胎児にも母体にも影響を及ぼさないことが証明されていないため理想的でない．胎児の健康状態把握の代替手段は，胎児心音聴診である．これは，妊婦の腹部表面からの振動音響信号を記録する方法であり，胎児心音によって生成された音響信号から胎児心音図の表示や，その心音データから心拍数を計測することが可能である．そのうえ，妊婦の腹部表面に取り付けるだけで計測可能なため時間とともに変化する胎児の健康状態を記録することが容易である．上記の聴診が安全性も高く簡易的であるにも関わらず主流になっていないのは，胎児心音が大きな雑音にかき消されてしまい，検出が困難となるからである．近年では，胎児心音の検出精度向上ために様々な研究が行われ，改善されつつある．

文献 [2] では，Koutsiana らが雑音除去方法の一つとして WT-FD 法を提案し，ノイズの中から胎児の心音を効果的に検出することを試みた．PhysioBank で入手可能な模擬胎児心音に対しテストを行った．評価実験において，信号対ノイズ比が低いほど Ⅱ 音は正しい位置での検出でき，I 音はすべての信号に対して正しい位置での検出に成功できるという結果が得られた．

文献 [3] では，Ⅱ 音の検出率を向上させるため，WT-FD

[1] 大分大学大学院工学研究科
Graduate School of Engineering, Oita University
[2] 大分大学理工学部
Faculty of Science and Technology, Oita University

法を行う前に雑音除去の手法の一つであるスペクトルサブトラクション法を適用した手法が提案された．全体の結果として胎児心音検出精度の向上に成功した．しかしながら，これらの処理をソフトウェアで実装すると，最近の標準的なパーソナルコンピュータ上で心音録音時間に対して約 3 倍時間を要し，常時リアルタイムモニタリングを実現できないという課題がある．

本研究では現在ソフトウェアで実装されている心音検出処理のうち，多くの時間を要するフラクタル次元解析 (FD) 部をフィールドプログラマブルゲートアレイ (FPGA) 上のハードウェアとして実装し，処理時間の短縮を図る．実験により，FD 部の処理時間を 10 分の 1 以下に短縮でき，胎児の常時リアルタイムモニタリングを実現できることを示す．

本稿では，第 2 節で心音処理技術を解説し，第 3 節で従来手法の実行時間について述べる．第 4 節で提案システムを概説し，第 5 節でその実装方法を述べる．第 6 節で実装システムの回路規模，実行時間の評価実験結果を示し，考察する．第 7 節でまとめと今後の課題及び展望を述べる．

2. 心音処理技術

2.1 離散ウェーブレット変換

ウェーブレット変換は周波数解析の一つであり，既定関数としてウェーブレット関数を用いる．この変換は広い周波数領域に対して，周波数特性を求める際に時間領域の情報を残すことができるのが特徴である．信号に対してこの変換を行うとウェーブレット係数を得ることが出来き，それらの係数を用いて元の信号を再構成することを逆変換という．離散ウェーブレット変換は，離散的にサンプリングされたウェーブレットを用いたウェーブレット変換のアルゴリズムである．逆変換を考慮した離散ウェーブレット変換は，一度変換した情報を加工して逆変換することでノイズ除去に応用されている．本稿ではこの離散ウェーブレット変換をウェーブレット変換と呼ぶ．

2.2 フラクタル次元解析

フラクタルとは複雑な形状を同じパターンの図形で表す数学的概念である．

フラクタル次元解析は，$W = \lfloor 0.05 * Fs \rfloor$ サンプル長のスライディングウィンドウを使用して実行される．ここで Fs は信号のサンプリング周波数を表す．処理を行う信号を N サンプルの入力ベクトルとすると，フラクタルの値を得るために W ウィンドウを N サンプルの入力ベクトルに沿って 1 サンプルずつシフトして計算が行われる．また W ウィンドウで得られたフラクタルの値はウィンドウの中点の値に該当する．このようにしてフラクタルの値を求めるとベクトルの長さは N より短くなるので，始めと終わりの欠損部分は $FD(1)$ と $FD(N-W+1)$ の値で補われる．N サンプルの信号に対してフラクタルの値は以下のように定義されている．

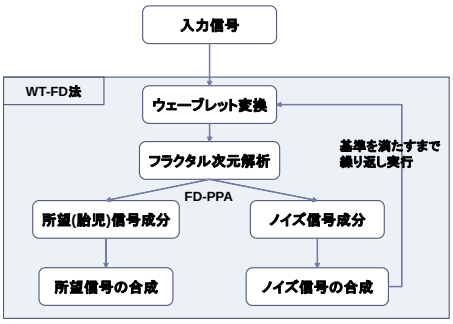

図 1 WT-FD 法反復手順
Fig. 1 Iteration of WT-FD method

$$FD = \frac{\log_{10}(n)}{\log_{10}(\frac{d}{Lc}) + \log_{10}(n)} \tag{1}$$

ここで，Lc は曲線の全長であり，連続する点の距離の合計となる．

$$Lc = \sum_{i=1}^{N-1} dist(i, i+1) \tag{2}$$

ここで $dist$ は 2 点の距離を表すものとし，最初の点と最も遠い点の距離 d は以下のように推定される．

$$d = max[dist(1,i)] \quad i \in [2, N] \tag{3}$$

2.3 WT-FD 法

ウェーブレット変換とフラクタル次元解析を組み合わせた心音検出手法である．胎児心音とノイズを分離するために繰り返し実行される．図 1 は WT-FD 法における反復手順を示す．

2.4 FD ピークピーリングアルゴリズム（FD-PPA）

フラクタル次元解析の結果から信号のピークを構成する部分を徐々に集め心音の位置を特定するアルゴリズムである．このアルゴリズムを使用することで胎児心音に関係するウェーブレット係数のセグメント化に必要な信号と雑音に関連するウェーブレット係数のセグメント化に必要な信号の 2 つが得られる．

2.5 スペクトルサブトラクション（SS）法

雑音処理の手法の一つであり，雑音を含んだ信号データの周波数スペクトルから，雑音の周波数スペクトルの平均値を引くことで雑音を除去する方法である．

3. 従来手法の実行時間

文献 [3] のプログラム使用して心音検出処理全体，SS 法，ウェーブレット変換，フラクタル次元解析，FD-PPA を Matlab を用いて実行した際の処理時間を計測する実験を行った．この実験には，Windows10，プロセッサ Intel(R)

表 1　ソフトウェアでの各処理の実行時間と出現回数
Table 1　Operation time and the number of occurrences of each process in software

処理	1 回の実行時間 (s)	出現回数	合計実行時間 (s)
SS 法	0.006	1	0.006
WT 変換	0.005	3	0.016
FD 解析	3.701	7	25.930
FD-PPA	0.002	7	0.012
その他	2.638	1	2.638
全体	–	–	28.602

Corei7, メモリ 8.00GB を搭載したパーソナルコンピュータ上で Matlab を使用し，サンプリング周波数が 1kHz，SNR 値が −6.6dB である 10 秒分の模擬心音データを使用した．

それぞれの処理を 3 回実行し，心音検出処理 1 回あたりの実行時間と合計実行時間に対して平均を算出したものを表 1 に示す．表 1 より，心音検出処理に要する時間が約 29 秒と胎児リアルタイムモニタリングの実現にはほど遠い．また，心音検出処理の中でもフラクタル次元解析に一番時間が掛かっており，出現回数も多いため，どんな信号に対しても処理が支配的になっている．

4. 提案システム

提案する心音検出処理の構造を図 2 に示す．3 節より，フラクタル次元解析が心音検出処理において最も支配的であることが分かっており，フラクタル次元解析の実行時間を短くすることが出来ればリアルタイム処理が実現できると考える．フラクタル次元解析のみ FPGA 上で実装し，それ以外の処理はソフトウェアで実装するものとする．

本研究では，心音データはサンプリング周波数 1kHz とし，10 秒ごとのファイルとして入力されるものとする．従来手法において，ソフトウェア上でデータは Double 型で扱われている．本研究においても同様の精度で処理を行うため，FPGA 上でも Double 型として扱う．

次にフラクタル次元解析をハードウェア上に実装する際の FPGA 上のシステム構成を図 3 に示す．フラクタル次元解析は，加算，減算，乗算，除算，平方根，自然対数，比較の演算を行う．これらには FPGA ベンダーより提供されている IP コアを利用する．制御回路は，メモリからのデータ受け取りと，これらの演算の実行タイミングを指定する．はじめに，ホスト計算機のソフトウェア上で計算したウェーブレット変換後の入力データを，USB ポートを使いメモリへ格納する．次に，FPGA 上の CPU から演算開始命令を送信することで，順次メモリからデータを取得し制御回路に渡し処理を行う．演算後得られたデータは制御回路から再度メモリへ格納する．最後に，CPU によって USB ポートからメモリの内容をホスト計算機に転送し，ソフトウェア上の処理に戻る．

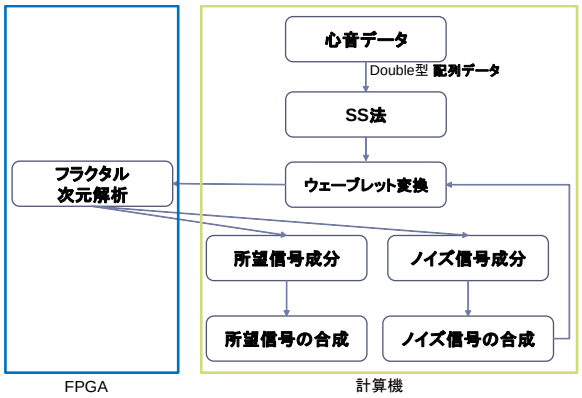

図 2　心音検出処理の構造
Fig. 2　Structure of the heartbeat detection process

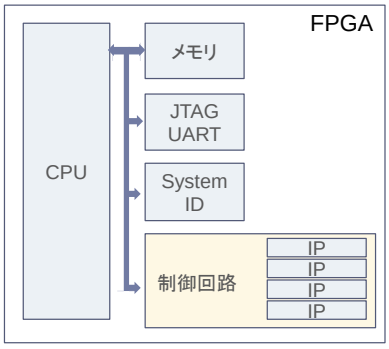

図 3　FPGA 上のシステム構成
Fig. 3　Architecture of the system on an FPGA

5. システムの設計

5.1 構成要素

本研究では，Intel 社製の CycloneV を搭載した Terasic 社製 DE0-CV ボードを用いて開発を行った．開発ツールとして Intel 社製の Quartus Prime 18.1 を使用した．

フラクタル次元解析を構成する各演算を行うための IP コアとして，ALTERA_FP_FUNCTIONS を使用した．ALTERA_FP_FUNCTIONS では，加算，減算，乗算，除算，平方根，自然対数，比較などの演算回路が提供されている．本研究では，FPGA のクロック周期を 20ns に設定して動作させるため，IP 作成時に動作周波数を 50Mhz に設定した．その際の演算子のレイテンシを表 2 に示す．

int 型から double 型への変換についても，IP コア ALTFP_CONVERT を使用した．入力データと演算後のデータを格納するメモリとして RAM:2-PORT 及び ROM:1-PORT を使用した．RAM は NiosⅡ システムのバスからの読み出しと演算側からの書き込みが独立したデュアルポート RAM であり，それぞれ独立したアドレスを持つ．double 型の 1 万個のデータを扱うため，RAM と ROM どちらもビット幅を 64bit，ワード数を 16,384 に設定した．これらの IP は周波数に関わらずレイテンシは固定で，

表 2　周波数 50Mhz に対する IP コアのレイテンシ
Table 2　Latency for each IP core with frequency 50 MHz

機能 (IP)	レイテンシ
加算	4
減算	4
乗算	3
除算	16
平方根	12
自然対数	15
比較	0

ALTFP_CONVERT は 6，RAM，ROM は書き出し読み出しに関わらず 1 である．

5.2　回路の詳細

フラクタル次元解析は式 1 から式 3 で求められる．式より dist や加算，最大値など同じ演算が繰り返し使われていることから，機能ごとにモジュールに分けて回路を作成した．以下では，式 2 と式 3 の dist を計算するモジュールを mod，式 2 と式 3 の加算・最大値を求めるモジュールを dl，式 1 の演算を行うモジュールを fd，メモリからデータを読み書き，3 つのモジュール同士を繋ぐ回路を ctrl と呼ぶこととする．

これらのモジュールをそれぞれデータフローグラフ (DFG)[4] で表現した．5.1 節の IP コアのサイズを考慮し，全体の回路面積が DE0-CV にプログラムできるよう，加算器，乗算器，減算器，平方根の演算器はそれぞれ 2 個，その他の演算器は 1 個のアロケーションを行った．それぞれ，表 2 に示したレイテンシを持つ．

モジュール mod のスケジューリング済 DFG を図 4 に示す．図の横線はコントロールステップの境界を表している．変数 i は 1 から 49 の値をとり，data はメモリから対応するアドレスに格納されたデータである．図 4 より mod を 1 回動作させるためには 26 クロック必要であることがわかる．また，データは全てレジスタに格納されており，入力データが与えられて値が反映されるまでに 1 クロック要するため，データが入力された時点から処理を終えるまでに 27 クロック必要であるとわかる．

はじめに回路が動作を開始してから，入力変数の浮動小数点型への変換と，メモリからデータを読み出して mod にデータを受け渡すまでに 7 クロックかかることが分かっている．mod の演算 2 回目からは，前の mod 演算中にこれらの処理を行うため，データの受け渡しに要するのは 2 クロックとなる．

モジュール dl のスケジューリング済み DFG を図 5 に示す．dl を 1 回動作させるためには 4 クロック必要であることがわかる．mod から演算終了信号を受け取った後入力データが反映されるまでに 1 クロック，dl の演算終了後結果がレジスタに格納されるまでに 1 クロック必要なため，

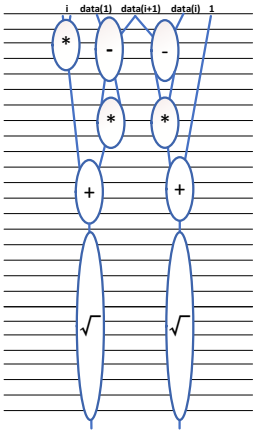

図 4　モジュール mod のスケジューリング済み DFG
Fig. 4　A scheduled DFG of module mod

データが入力された時点から処理を終えるまでに 6 クロック必要である．

mod から dl を終えるまでの処理を制御する ctrl の DFG を図 6 に示す．図 6 に表記されている mod は図 4 のモジュール，dl は図 5 のモジュールを表している．mod と dl は並列に動作している．dl の加算器は mod 中の加算器を再利用しているため，競合しないようにスケジューリングを行う．

1 つのフラクタル値を求めるためにはデータ 50 個に対して mod の演算を行い，mod と dl は 49 回実行される．dl の処理は 48 回目までは mod の実行中に動作が終了するため，実行時間を計算する際は 49 回目のみ考慮すれば良い．回路が動作を開始してから d，Lc を求めるまでに 1,432 クロックかかることがわかる．

モジュール fd の DFG を図 7 に示す．fd の加算器は mod 中の加算器を再利用する．

fd を 1 回動作させるためには表 2 より，51 クロック必要である．計算途中の値はレジスタに格納している．dl から演算終了信号を受け取り入力データが反映されるまでに 1 クロック要するため，データが入力された時点から結果が出力されるまでに 55 クロック必要であることがわかる．

10 秒分の心音データに対し，フラクタル値を求めるためにはデータ約 1 万個に対して上記の処理を行う．fd の演算は次のスライディングウィンドウに対してフラクタル値を求めている間に動作が終了するため，全体の実行時間を計算する際は最後の 1 回分のみ考慮すれば良い．

上記より 10 秒分のデータに対して合計クロック数は 1,432 万クロックであることから，結果が得られるまで 0.28 秒とわかる．これはソフトウェア動作時の 10 分の 1 以下である．

5.3　Quartus Prime によるシステムの実装

提案するフラクタル次元解析を実現する回路を Verilog HDL で作成した．回路を構成する各モジュールについて，

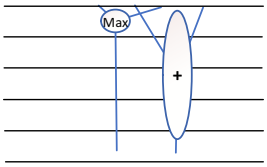

図 5 モジュール dl のスケジューリング済み DFG
Fig. 5 A scheduled DFG of module dl

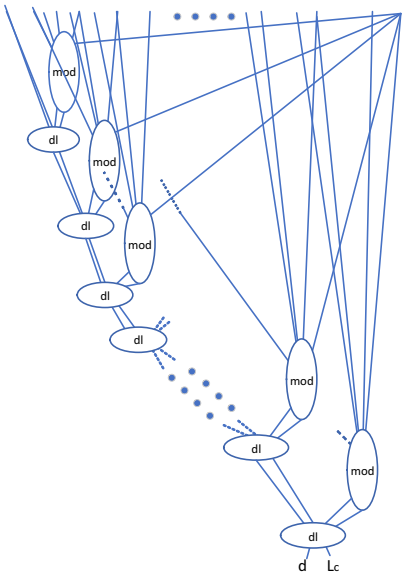

図 6 モジュール ctrl の DFG
Fig. 6 A DFG of ctrl

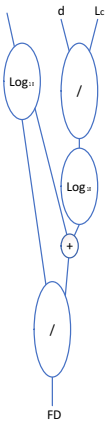

図 7 モジュール fd の DFG
Fig. 7 A DFG of module fd

5.2 節に示した DFG に沿って，それぞれ ctrl.v，mod.v，dl.v，fd.v として記述した．提案回路を一つのコンポーネントとして NiosⅡ システムに組み込む．システム構築には，回路やアドレス割り当てなどを設定することが出来る Platform Designer を利用した．Platform Designer によりシステムに NiosⅡ (Fast) コア，On-Chip Memory，JTAG，SystemID，提案回路を追加した．ここでの On-Chip Memory は NiosⅡ のブート及び実行するためのメモリである．提案回路は ctrl.v を最上位階層として設定している．これら回路とメモリに対してアドレスの割り当てやクロック，リセットの接続を行い NiosⅡ システム階層を生成した．さらに，NiosⅡ システム階層と入出力を接続する最上位階層も Verilog HDL で記述した．また，ピンアサインは文献 [5] のピンアサインファイルを指定している．

表 3 10 秒データを処理するために必要な回路規模
Table 3 Circuit size required to process 10 seconds of data

FPGA リソース	提案回路
メモリブロック	3,730k ビット
ALM	8,047 個
DSP ブロック	55 個

NiosⅡ のソフトウェア開発は NiosⅡ Embedded Design Suite(EDS) というツールを用いる．このツールで作成するプログラムやプロジェクトテンプレートの種類によって含まれる関数やサイズが異なるため，作成するシステムに合った On-Chip Memory のサイズを指定する．

本研究では回路の動作確認をするプログラムを C 言語で作成し，演算開始と回路の演算結果を読み出すために使用している．NiosⅡ システムのバスは 32 ビット幅であるため，1 つの演算データを 2 回に分けて読み出すようにした．

6. 実装評価

5 節で設計した回路について，ロジックアナライザ [6] を用いることで各モジュールの正常な動作を確認している．作成した回路を基に，10 秒の心音データを扱う際に必要な回路規模や，ソフトウェア実行との精度比較を行った際の手順と結果について述べる

6.1 実装に必要な回路規模

10 秒の心音データに対して処理を行う際の回路規模について調査した．NiosⅡ のプロジェクトでは演算結果を標準出力させる関数 (IORD_32DIRECT，printf) が含まれたテンプレート，Hello World Small を使用した．On-Chip Memory は NiosⅡ のプログラムを格納するのに十分な 1kB とした．

上記の設定で Compilation Report により得られた回路規模を表 3 に示す．本研究で用いた FPGA は，内臓 RAM が 3,080k ビットであるため，提案回路の秒分のデータを格納するための RAM を配置出来ないことがわかったが，10 秒の心音データに対して処理を行うために必要なリソースが確認できた．なお，これは一度に FPGA に転送する心音データ量のみに依存するもので，演算を実現する回路は配置できる．

6.2 精度評価

提案回路に対して，Matlab でソフトウェアとして実装されたフラクタル次元解析と精度が同程度か比較を行った．

表 4　ソフトウェアと提案回路における出力データの比較
Table 4　Table comparing output data with software and self-made circuit

総データ数	一致数	不一致数
4,092	4,080	12

6.1 節より 10 秒の心音データを処理する回路は本研究で用いた FPGA では実装できないことが分かっているため，心音データと演算結果を格納する RAM と ROM の規模を小さくすることで精度の評価を行った．

使用する RAM:2-PORT 及び ROM:1-PORT のビット幅を 64 ビット，ワード数を 4,092 に設定した．本評価では，Matlab で実行したフラクタル値と数値の比較が出来るようにテキストファイルに出力結果を保存するようにする．NiosⅡ EDS に用意されている 2 種類のファイルシステムの内 Host file system を利用した．Host file system を使用するために altera_hostfs の Enable を有効にした．これらの評価用の処理のため，NiosⅡ のプログラム格納のための On-Chip Memory は 16kB とした．

精度比較のための心音データとして PhysioBank で入手可能な模擬胎児心音を利用した．この心音データはノイズによって劣化した信号である．入力データとして，サンプリング周波数は 1kHz，4,092 個のデータに対して Matlab 内でウェーブレット変換を適用した後のデータを使用する．ソフトウェアと本研究で実装した提案回路それぞれに対してフラクタル次元解析を行い，得られた結果を比較した．ソフトウェアと提案回路における出力データの比較結果を表 4 に示す．演算結果の不一致は 12 個あることが分かった．

不一致があったデータが格納された配列番地とソフトウェアと提案回路の演算結果を表 5 に示す．演算結果は IEEE754 標準の倍精度浮動小数点型を 16 進数で表現している．不一致のデータが格納された配列番地に規則性がないことから回路の設計に問題はないと考えられる．また，値を確認したところ仮数部の最下位ビットが 1 ビット分異なっている．これは除算について，ソフトウェアと IP コアの演算アルゴリズムが異なるからと考えられる．得られたフラクタル値について，その後に行われる FD-PPA には影響しない誤差であると考えられる．

7. おわりに

本稿では，心音データによる胎児のリアルタイムモニタリングを実現するシステムの提案を行った．胎児の常時リアルタイムモニタリングを実現する際に問題となっている心音検出の処理時間に対して，処理が支配的になっているフラクタル次元解析を FPGA 上のハードウェアとして実装することで処理の高速化を図る方法を述べた．

提案回路を実装し，10 秒の心音データに対して，従来のソフトウェアで実装したフラクタル次元解析の演算時間を，小規模な回路で従来の約 10 分の 1 以下まで短縮できることを示した．また，提案回路と従来のソフトウェア実装の演算結果の精度の評価においては，4 秒のサンプルデータについて 99.7%が一致した．一致しなかった値については，誤差は 1 ビットの範囲内であり，ソフトウェアとハードウエアによる演算アルゴリズムの違いによるものと考えられ，無視できるものである．

今後の課題として，本稿で示された FPGA の回路規模を基に条件を満たす FPGA を選定し，回路を組み込み動作検証を行うことや，ホスト計算機との間の送受信の実装などが挙げられる．

表 5　不一致箇所に対するソフトウェアと提案回路の演算結果
Table 5　Calculation results of software and home-built circuit for discrepancies

番地	ソフトウェア	提案回路
620	3ff00002a61a0f58	3ff00002a61a0f57
750	3ff0000bc165da98	3ff0000bc165da97
802	3ff00002ee7cec15	3ff00002ee7cec14
970	3ff0014ce454aa3c	3ff0014ce454aa3d
982	3ff002153697ce9b	3ff002153697ce9c
984	3ff00216079c6ca7	3ff00216079c6ca8
1157	3ff0000615b26be0	3ff0000615b26bdf
2568	3ff001d25483662c	3ff001d25483662d
2903	3ff00000e98eebea	3ff00000e98eebe9
3132	3ff00146f402c7d0	3ff00146f402c7cf
3255	3ff0000127ae77c2	3ff0000127ae77c3
3967	3ff0021ce5604651	3ff0021ce5604652

謝辞　本研究に際して，使用した心音抽出プログラムに関して多くの議論や意見をいただきました本学理工学部の古家賢一教授に深謝致します．また，システム作成の際に御助言いただきました同技術部の松原重喜技術職員に深謝致します．

参考文献

[1] 佐藤昌司．“日本の死産の疫学 日本産科婦人科学会周産期登録データベースから，” 産科と婦人科, 75（4）, pp.413–417, Apr. 2008.
[2] K. Elisavet, H. J. Leontios, C. Loanna and K. H. Ahsan, “Fetal heart sounds detection using wavelet transform and fractal dimension,” Frontiers in Bioengineering and Biotechnology, vol.5, no.49, pp.1–9, Sep. 2017.
[3] 田中真央．“ウェーブレット変換とフラクタル次元解析を用いた胎児心音検出の精度向上の研究，” 大分大学学士論文, 2020.
[4] D. D. Gajski, N. D. Dutt, A. C-H Wu and S. Y-L Lin, *High-Level Synthesis*, Springer New York, NY, 1992.
[5] 改訂 2 版 FPGA ボードで学ぶ組込みシステム開発入門 [Intel FPGA 編] サポートページ, 技術評論社, 入手先 ⟨https://gihyo.jp/book/2018/978-4-7741-9388-5/support#supportDownload⟩, Jan. 21, 2023.
[6] 小林優．【改訂 2 版】FPGA ボードで学ぶ組込みシステム開発入門 [Intel FPGA 編], 株式会社技術評論社, 2018.