修 士 論 文

FPGAによる一般化調和解析の高速化手法に関する研究

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

南里 洋亮

2005年3月

修 士 論 文

FPGAによる一般化調和解析の高速化手法に関する研究

指導教官　田中清史 助教授

審査委員主査　田中清史 助教授
審査委員　日比野靖 教授
審査委員　井口寧 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

310079 南里 洋亮

提出年月: 2005年2月

概 要

窓長に依存しない周波数解析法として、一般化調和解析が注目されている。しかし、計算量が膨大なため、ほとんど実用されていない。本論文では、一般化調和解析において最も計算負荷となるコスト値演算を FPGA でハードウェア化し、それを用いた解析アルゴリズムを提案する。

提案手法で音響信号に対し窓長 512 サンプルの解析を行ったところ、ハードウェアを用いた解析速度はソフトウェアのみで解析した速度の約 10 倍となった。また、従来の一般化調和解析アルゴリズムと比較し高精度な周波数解析が可能となった。

目 次

第 1 章　序論 **1**
1.1　研究背景と目的 1
1.2　本論文の構成 1

第 2 章　一般化調和解析 **2**
2.1　一般化調和解析とは 2
2.2　従来手法 3
2.2.1　ABS 法 3
2.2.2　平田らによる一般化調和解析 4
2.3　提案手法 4
2.3.1　パラメータの推定 4
2.3.2　ハードウェア化部分 6

第 3 章　一般化調和解析ハードウェアの設計 **8**
3.1　ハードウェアの概要 8
3.2　FPGA 9
3.2.1　FPGA の構成 9
3.2.2　ブロック SelectRAM 11
3.2.3　エンベデッド 18×18 乗算器 12
3.3　フォーマット 13
3.4　コサイン演算 13
3.4.1　コサイン演算器 13
3.4.2　コサインテーブルを用いた設計 14
3.5　コスト値計算器の構成 15
3.6　データバッファ 19
3.6.1　データバッファ 19
3.6.2　コスト値計算器とデータバッファの接続 20
3.7　比較器 21
3.8　パラメータ生成器 22
3.9　全体の制御 23
3.10 コスト値計算器の並列化 25
3.10.1 コスト値計算器の並列化 25

3.10.2 4 並列パイプライン二乗誤差演算器 25
3.10.3 16 並列二乗誤差演算器 28
3.11 合成結果 28

第 4 章 ソフトウェアとの連携 **31**
4.1 ソフトウェアとハードウェアのパイプライン化 31
4.2 ソフトウェア処理の手順 32
4.2.1 ソフトウェア処理の手順 32
4.2.2 収束範囲の評価 32
4.3 減速ニュートン法の適用 33
4.3.1 3パラメータ同時推定 33
4.3.2 パラメータ F、ϕ と A を分けて推定 36
4.4 実装 37

第 5 章 評価 **40**
5.1 ハードウェア部の速度評価 40
5.2 ハードウェアとソフトウェアのパイプライン実行速度 41
5.3 精度評価 42

第 6 章 結論 **49**
6.1 まとめ 49
6.2 今後の課題 49

第1章　序論

1.1　研究背景と目的

音響信号の解析には、フーリエ級数展開理論を応用した STFT(Short Time Fourier Transform) が主に用いられている。しかし、STFT は窓長によってきまる基本周波数とその倍音成分のみの周波数解析しかできないという欠点がある。近年、解析窓の長さに依存せず、自由な周波数を持つ正弦波の和で表現するモデルの研究が進められており、音声や楽器音の符号化などの応用に期待されている。

一般化調和解析は、信号モデルのパラメータを最小二乗法を基本とした推定によりもとめ、周波数解析を行う手法である。STFT に比べ高い周波数分解能を持ち、窓長に依存しない解析が可能であるが、音響モデルのパラメータ推定に膨大な時間が掛かり実用には問題がある。

現在、一般化調和解析を効率的に解析するアルゴリズムが研究されているが、解析速度と精度を両立した手法は見つかっていない。本論文では、一般化調和解析の音響モデル信号のパラメータ推定で最も計算負荷となる二乗誤差の総和計算を FPGA(Field Progamable Gate Array) によりハードウェア化する。ハードウェアは固定小数点で演算を行い、音響モデルのパラメータを大雑把に推定しその結果を元にソフトウェアによる非線形最小二乗法の解法をパラメータ推定に適用する。このハードウェアとソフトウェアの連携により、高精度かつ高速な一般化調和解析手法を実現する。

1.2　本論文の構成

本論文は全 6 章からなる。第 2 章では一般化調和解析について述べる。第 3 章では一般化調和解析のパラメータ推定を行うハードウェアの設計について述べる。第 4 章ではハードウェアとソフトウェアのパイプライン実行とソフトウェアが行うニュートン法を用いたパラメータ推定について述べる。第 5 章で、本提案手法のハードウェアを追加した場合の速度とソフトウェアのみの実行の速度比較および、従来手法とと本提案手法の解析精度比較を行う。第 6 章で本論文の結論を述べる。

第2章　一般化調和解析

2.1　一般化調和解析とは

一般化調和解析は 1958 年に wiener[1] によって提唱された信号の周波数解析手法である。窓長の影響を受けることなく非定常信号の解析を行うことが可能であり、観測区間を越えて信号の予測が可能であるなどの特徴がある。

一般調和解析は次に示す自由な周波数成分をもつ正弦波の和により解析対象信号を表す。

$$\hat{x} = \sum_{k=0}^{J-1} \Big\{ A_k \cos(2\pi(f_k t + \phi_k)) \Big\} \tag{2.1}$$

J:正弦波の数, A_k:振幅, f_k:周波数, ϕ_k:位相差

J 組のパラメータを一度に求めることは困難である。そのため、解析対象信号 x と正弦波モデル信号の二乗誤差総和 W（以下、コスト値と呼ぶ）

$$W = \sum_{i=1}^{N} \Big\{ A_k \cos(2\pi(f_k t_i + \phi_k)) - x_i \Big\}^2 \tag{2.2}$$

N:解析区間長

が最小となるパラメータ組 A_k,f_k,ϕ_k $(k = 0...J)$ を 1 組ずつ求める。具体的には、推定したパラメータ組で表される正弦波を解析対象信号から差し引き、その残差信号に対して再度最小パラメータの組を求める。このようにして、逐次的に正弦波を任意の数だけ抽出することによって周波数解析をおこなう。図 2.1 に分析の様子を示す。右側は逐次推定された正弦波の合成波、左側は対応する右側の波形を原波形から差し引いた誤差信号である。

一般化調和解析は、このパラメータ推定に膨大な計算量が必要であるという問題から、ほとんど実用されていない。

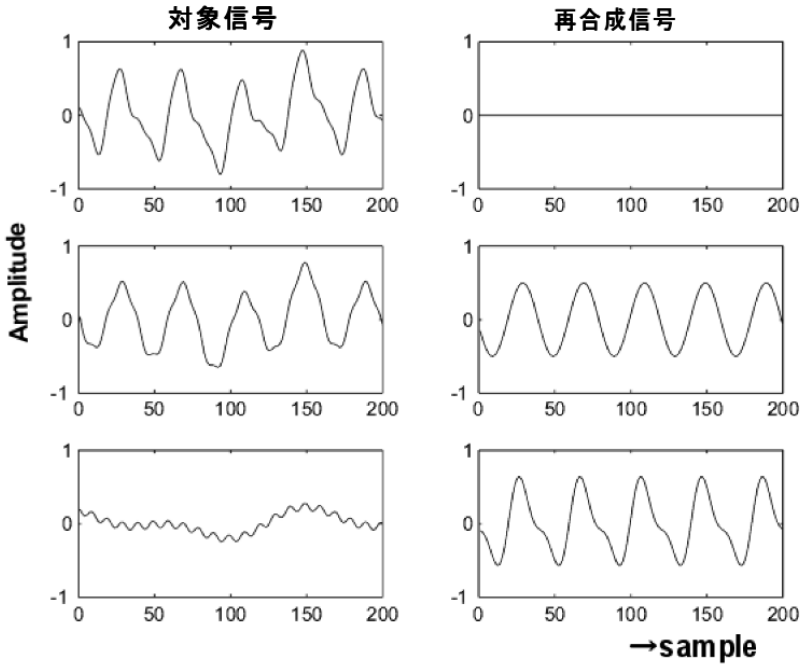


図 2.1: 一般化調和解析

2.2 従来手法

一般化調和解析の従来手法として、ABS(Analysis By Synthesis) 法 [2] [3]、平田らによる一般化調和解析 [4] [5] がある。これらの手法はモデル信号を

$$\hat{x} = \sum_{i=0}^{J} A_i \sin(2\pi f_i t) + B_i \cos(2\pi f_i t) \tag{2.3}$$

のように変形し、解析対象信号との二乗誤差総和（コスト値）を最小とするパラメータ振幅 A、振幅 B、周波数 f の推定を行うことで解析する。

2.2.1 ABS 法

ABS 法ではこのパラメータ推定において、パラメータ f を固定し振幅 A、B を最小二乗法で求める。周波数 f を一様な範囲で分割させる。f_s をサンプリング周波数、M を分割数とすると、

$$f_i = \frac{f_s}{2i} \qquad (i = 1, 0, \ldots, M)$$

のように、周波数は分割される。それぞれの f_i に対し A_i, B_i を求め、そのコスト値を計算し、コスト値が最小となるパラメータ組 A_i、B_i、f_i を見つける。

ABS 法の問題点は、周波数の分割数 M に周波数分解能が依存することである。周波数分解能をあげるために分割数 M を大きくすると、A、B 推定のための最小二乗法の適用回数が増え、計算量が膨大になる。M が小さすぎると、正確な周波数推定が行えず、周波数分解能の精度が下がる。

2.2.2 平田らによる一般化調和解析

平田らによる一般化調和解析は、周波数の周期を解析区間長から 1 サンプル分ずつ変化させていき、それぞれの f でのパラメータ A、B をフーリエ係数として近似的にもとめ、パラメータ推定を行う。解析窓のサンプル数を N、周期を T とすると、

$$f_i = \frac{fs}{T_i} \qquad (i = 1, 2, \ldots, N-2)$$

T_i を 1 サンプル分ずつの時間で変化させ、その周波数 f_i における、パラメータ A_i、B_i を

$$A_i = \frac{2}{rT} \sum_{n=0}^{rT-1} data(n) \sin(2\pi f_i t(n))$$

$$B_i = \frac{2}{rT} \sum_{n=0}^{rT-1} data(n) \cos(2\pi f_i t(n))$$

とする。ここで r は解析区間長に含まれる f_i の周期の数である。すなわち、A_i、B_i はフーリエ係数となり、パラメータは近似値として表される。求まった A_i、B_i、f_i それぞれのコスト値を計算し、コスト値が最小となるパラメータ組を解析結果として抽出する。

この手法は、高速なパラメータ推定が行えるが、近似的なパラメータ推定をしているため精度には問題がある。また、ABS 法と同様にあらかじめ設定された周波数上でしか検波できていないため、信号を構成する正確な周波数成分を推定できていない。

2.3 提案手法

2.3.1 パラメータの推定

従来手法では、周波数の推定に自由度を与えていないため、正確な周波数成分の解析が不可能であった。本論文では正確な周波数成分の推定のために。従来手法では行っていなかった周波数 f の推定にも非線形最小二乗法を適用する。モデル信号を式 (2.1)、コスト関数 (モデル信号との二乗総和誤差) を式 2.4 で表す。

パラメータ組を 1 組ずつ求める場合でも、二乗誤差総和が最小となる A、f、ϕ を非線形最小二乗法の解法により一度に推定することは、非常に困難である。非線形最小二乗法

は任意に与えた初期値を出発点とした反復計算により、コスト値が最小となるパラメータ値を推定する。非線形性の強いコスト関数の場合、局所的な解（local minimam）に陥りやすい。本コスト関数は、非常に非線形性が強く局所的な解が数多く存在する。図 2.2 に音響信号に対して、パラメータ A の値を固定し、f、ϕ の値を変化させたときのコスト値を平面的にプロットしたものを示す。

本コスト関数に非線形最小二乗法を適用させるには、ある程度真値（コスト値が最小となるパラメータ値）に近い初期値を与える必要がある。また、同時に推定させるパラメータが多いほど真値の近くに初期値を与える必要がある。

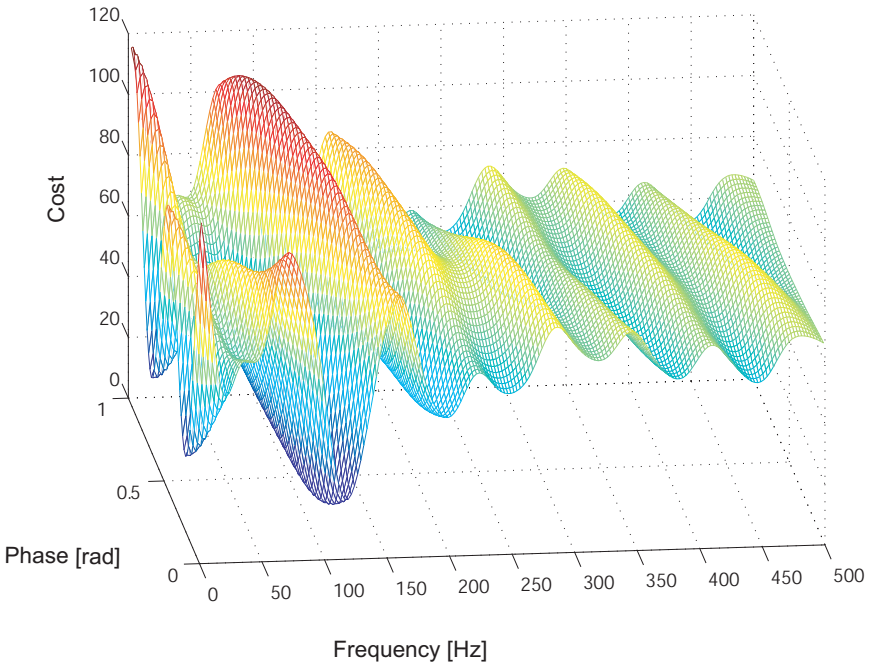


図 2.2: コスト平面

パラメータ A、f、ϕ を最小二乗法を用いて推定する方法として、次の 2 種類が考えられる。

- 多数の初期値に対して非線形最小二乗法の解法を適用し、それぞれの局所的な最小パラメータ組から最も最小となるものを選ぶ方法
- パラメータ A、f、ϕ を一様に細かく変化させ、最も二乗誤差総和が最小となるパラメータ組を選び、選び出したパラメータを初期値として、非線形最小二乗法を適用する方法

前者の方法は、最小二乗法を用いて周波数推定を行うために精度は高くなるが、多数の初期値に対して最小二乗法を適用するため、必然的に二乗誤差総和の計算回数が多くなり、計算量が膨大になる。後者の方法は、計算量は前者のものに比べ小さくできるが、はじめに A、f、ϕ を分布させる量が少ないと、局所的な最小値に陥りやすい。

本手法では、後者の方法において、一様に分布させたパラメータ A、f、ϕ のコスト演算を FPGA を用いてハードウェア化することにより高速化する。その後、ハードウェアにより推定されたパラメータを初期値としてソフトウェアによる非線形最小二乗法を適用する。以上により高速かつ周波数推定の高精度を両立した解析を実現する。

2.3.2 ハードウェア化部分

本提案手法では、一般化調和解析で最も計算負荷となる二乗誤差の総和計算

$$W = \sum_{i=0}^{N} \Big\{ A\cos(2\pi(ft_i + \phi)) - data_i \Big\} \tag{2.4}$$

を FPGA を用いてハードウェア化し、解析の高速化を計る。具体的には、ハードウェアは一様な分布でのパラメータ A、f、ϕ の組でコスト値を計算し、そのコスト値が最小値となるパラメータ組を推定する。また、計算の高速化とハードウェア量の削減の観点から、固定小数点で実装する。ハードウェアで探索する範囲を図 2.3 に示す。パラメータ A、f、ϕ をそれぞれ任意の分割数で分割する。A の範囲は $0 \sim A_{max}$ の範囲で分割された範囲を探索する。解析対象信号のサンプリングデータを $data$、解析サンプル数を N とした場合 A_{max} は以下の式から導く。

$$A_{max} = \frac{1}{N} \sum_{i=0}^{N-1} \sqrt{data_i^2} \tag{2.5}$$

f は 0[Hz] からサンプリング周波数 f_s までの間を分割して探索、ϕ は $0 \sim 1$[rad] の範囲を分割して探索を行う。

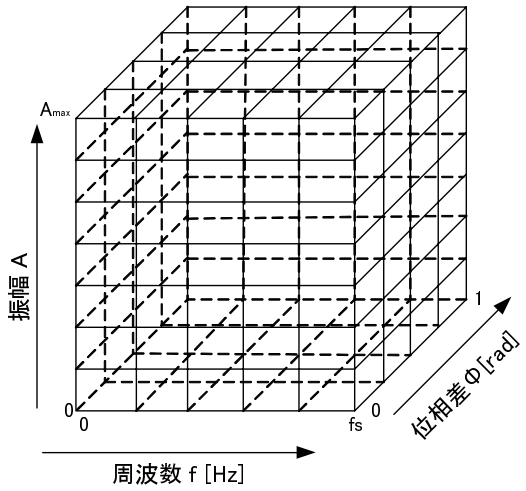


図 2.3: パラメータ A、f、ϕ の探索範囲

第3章　一般化調和解析ハードウェアの設計

この章ではFPGAを用いた一般化調和解析のハードウェアについてその概要と設計法を述べる。

3.1　ハードウェアの概要

ハードウェアの役割はソフトウェアで推定するための、大雑把な値を推定することである。パラメータ A, f, ϕ の値を一様な範囲の組み合わせでそれぞれのコスト値を求め、コストが最小値となるパラメータを推定する。そのような動作を実現するためには以下のモジュールが必要である (図3.1)。

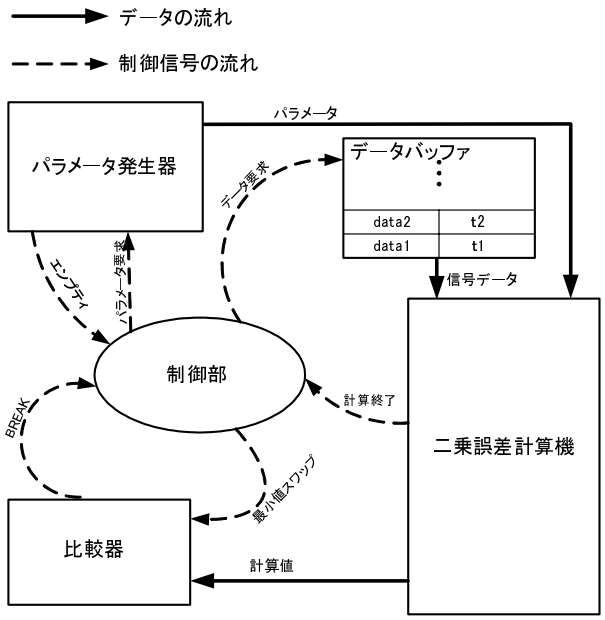


図 3.1: ハードウェア構成

- コスト値を計算する二乗誤差総和演算器

- サンプリングデータ、時間情報を格納するためのデータバッファ
- 最小値のコストとそのパラメータをレジスタに保持し（以下、最小パラメータレジスタ)、その値と計算結果のコスト値を比較する比較器
- 指定された一様な範囲でのパラメータの組み合わせでパラメータを生成するパラメータ生成器

推定を行うためのハードウェアの動作を次に示す。

1. パラメータ生成器にFPGA チップの外部からパラメータ A, f, ϕ の推定範囲が指定される。
2. FPGA 外部からデータバッファに窓長分のデータ $(data)$、時間 (t) が書き込まれる
3. 計算開始信号が外部からアサートされると、パラメータ生成器の示すパラメータ値とデータバッファ内のデータを用いて、コスト演算を開始する。
4. コスト値が比較器内の最小値より大きくなると、演算器は演算を中止し、パラメータ生成器は次のパラメータを呼び出し新たなコスト値演算を行う。コスト値が比較器内の最小値より大きくならずに計算終了すると、そのコスト値およびパラメータ値を比較器内の最小パラメータレジスタ値とスワップし、次パラメータをパラメータ生成器が呼び出し新たなコスト値演算を行う。
5. パラメータ生成がすべての組み合わせで生成され、全ての計算が終了した時点で比較器内の最小パラメータレジスタが推定結果となる。

本章では以下、FPGA を使ったハードウェアの構成法について述べる。

3.2 FPGA

FPGA は、ハードウェア構成をユーザが自由に変更できるPLD（Programmable Logic Device）の一種である。近年、FPGA に内部RAM、乗算器などが組み込まれた製品が生産され、数値演算、画像処理、音声処理アルゴリズムのハードウェア化などが比較的容易にできるようになってきている。

3.2.1 FPGA の構成

本論文では、Xillinx 社 VirtexⅡ シリーズの FPGA(XC2V6000)[6][7] を使用した。その FPGA 内部の構成を図 3.2 に示す。

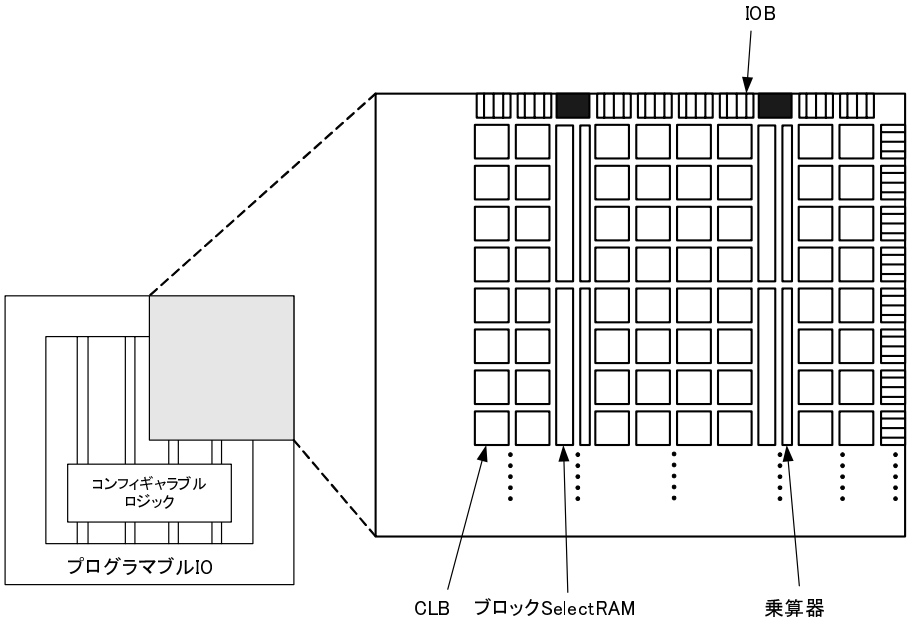


図 3.2: vertexII アーキテクチャの概要

内部は主に以下の要素から構成される。

- Configurable Logic Block(CLB)
  回路を構成するために必要なロジックの役割を果たす。組み合わせロジック、単一ロジックのファンクションエレメントを提供する。

- Input/Output Block
  CLB と FPGA チップのピンとの仲介をする。

- Block SelectRAM
  FPGA チップ内のメモリ。同期型 RAM として任意の入出力ポート幅で構成することができる。

- 18X18bit Multiplier
  FPGA チップ内部の組み込み乗算器、18 ビット ×18 ビットの符号付乗算（２の補数）が可能

CLB は 4 つのスライスと呼ばれる構成要素からなる。さらに各スライスは、4 入力 LUT(Look Up Table)、16 ビットシフトレジスタ、16bit 分散 SelectRAM としてコンフィギュレーションできるファンクションジェネレータが 2 つ、その他マルチプレクサや四則演算ロジックゲートなどで構成されている。

FPGA の回路規模はスライス単位で計算される。XC2V6000 では全体で 33,792 スライスが使用可能である

3.2.2 ブロックSelectRAM

Block SelectRAM は 18Kb のデュアルポート RAM で、FPGA 内で高速な大容量ブロックメモリとして利用できる。図 3.3 に Block SelectRAM のデュアルポートデータフローを示す。

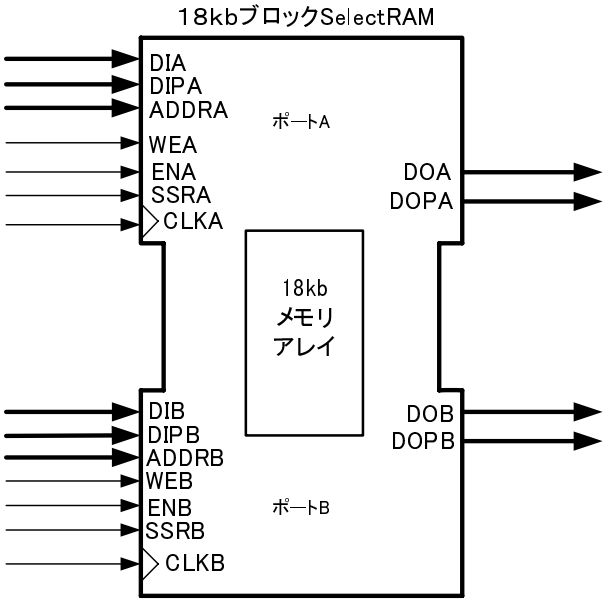


図 3.3: Block SelectRAM デュアルポートデータフロー

CLK[A|B]:クロック	EN[A|B]:イネーブル
WE[A|B]:ライトイネーブル	SSR[A|B]:セット/リセット
ADDR[A|B]:アドレスバス	DI[A|B]:データ入力バス
DO[A|B]:データ出力バス	
DIP[A|B]:データ出力バス（パリティ）	DOP[A|B]:データ出力バス (パリティ)

Block SelectRAM は独立した 2 つのアクセスポート (A,B) から構成される。データの書き込みはいずれか 1 つのポート、データの読み出しは、同時に 2 つのポートから行うことができる。

Block SelectRAM の大きな特徴は、出力データバスを任意のポート幅にコンフィグレーションできることである。ワード数 (アドレス幅) を変化させることにより、メモリ容量 (18kB) を固定したまま、ポート幅を変化させることができる (表 3.1)

Xilinx 社の FPGA では、HDL で Block SelectRAM を呼び出すためのプリミティブが用意されている。プリミティブには、デュアルポート Block SelectRAM だけでなく図 3.4 に示すシングルポートの Block SelectRAM も用意されている。

XC2V6000 では、最大 144 個 (2592Kbit) の Block SelectRAM が使用可能である。

出力ポートのデータ幅 [bit]	ワード数	ADDR バス幅 [bit]
1	16,384	14
2	8,192	13
4	4,096	12
8	2,048	11
16	1,024	10
32	512	9

表 3.1: ポート比率

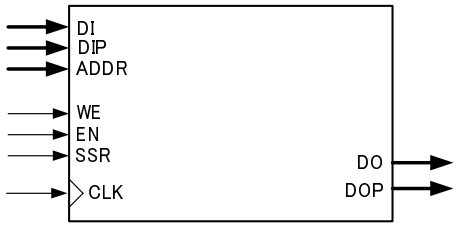


図 3.4: シングルポート Block SelectRAM

3.2.3 エンベデッド 18×18 乗算器

VirtexⅡ には、18 ビット ×18 ビットの符号付乗算器（２の補数）が組み込まれている。（図 3.5）この乗算器を使用すれば、CLB を使って乗算器を構成するよりも、速い速度で積を得ることができる。

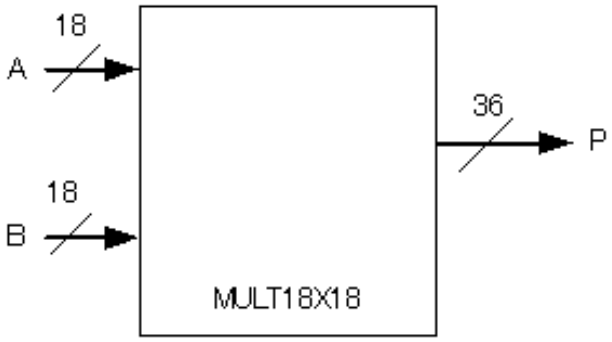


図 3.5: エンベデッド乗算器

エンベデッド乗算器で符号なし乗算を行う場合は、入力 17 ビットまでのデータが扱える。入力 18 ビット以上の乗算を行う場合には、複数のエンベデッド乗算器と CLB で構成した加算器を使用して、35 ビット ×35 ビット（符号付）までの乗算器を構成することがで

きる。エンベデッド乗算器も Block SelectRAM と同様に、Xilinx 社から HDL 用のプリミティブが用意されている。XC2V6000 では、最大 144 個までのエンベデッド乗算器が使用できる。

3.3 フォーマット

本論文のハードウェアは固定小数点（整数）演算を行う。固定小数演算器での実装は回路規模を小さくすることが可能である。しかし、固定小数点は扱える数値の範囲が限られているため、計算に使用するフォーマットを選定する必要がある。図 3.6 に、本論文で選定したフォーマットを示す。

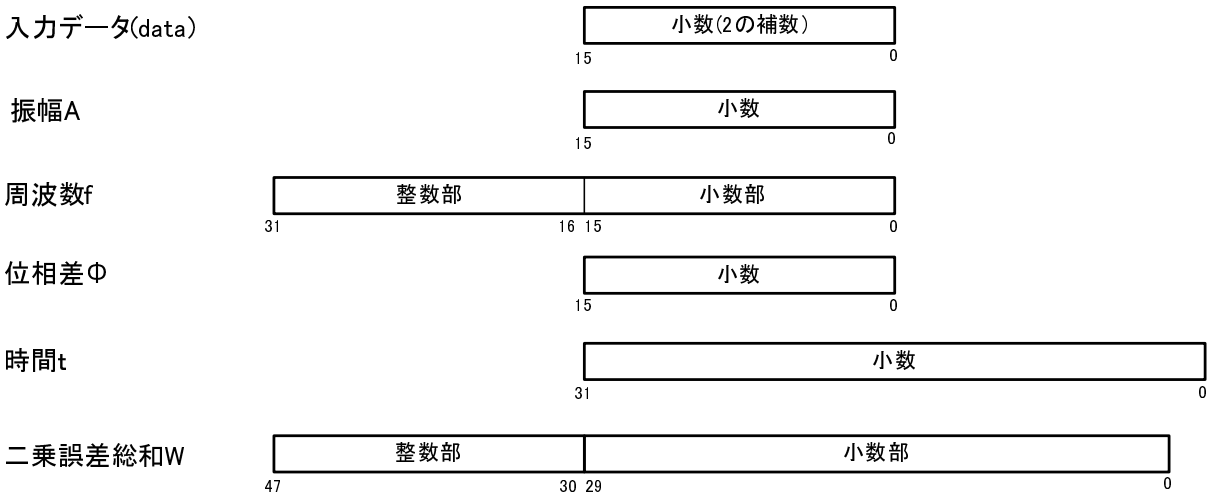


図 3.6: データフォーマット

解析対象とする音声信号はサンプリング周波数 44.1kHz 量子化ビット数 16 ビットの PCM 形式のデータである。これはオーディオ CD の規格である。

3.4 コサイン演算

3.4.1 コサイン演算器

一般調和解析のコスト値演算にはコサイン演算が必要である。このコサイン演算をハードウェアで実現する方法として、以下のものがある。

1. 三角関数の級数展開の式をハードウェア化
   三角関数の級数展開式

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + (-1)^k \frac{x^{2k}}{(2k)!} - \cdots$$

をハードウェア化する

2. テーブルを用いる方法
   ROM を用いて、コサイン演算値をテーブル化する。

1 の方法は高精度な演算値を得ることが可能である。しかし、1 サイクルで演算を行うことができない、ハードウェア規模が大きくなるなどの欠点がある。2 の方法は 1 サイクルで演算値を得ることができる。しかし演算精度を向上させるには、テーブルサイズ（ROM）の容量を大きくする必要がある。本論文のハードウェアは、大雑把なパラメータ推定を目的としているため高精度なコサイン演算を必要としない。そのため、2 のテーブルを用いたコサイン演算を採用する。

3.4.2 コサインテーブルを用いた設計

本論文では FPGA 内部のメモリである Block SelectRAM のシングルポートのみを用いてコサインテーブルを作成する。。Block SelectRAM1 個を出力データ長 16bit で使うと、1024 エントリー（アドレス入力 10bit）のテーブルとなる。正弦波の対象性を利用すれば、コサインテーブルは 4 分の 1 にまでサイズを小さくすることができる（図 3.7、図 3.8）。Block SelectRAM1 個を用いて、入力 12bit 出力 16bit 相当のコサイン演算モジュールを作成することができる。

3.5 コスト値計算器の構成

コスト値計算器の構成図を 3.9 に示す。それぞれのモジュールの間に、ラッチを挿入しパイプライン動作させる。パイプライン総段数は 10 段である。高速化の観点から、すべての乗算器は CLB で構成せずに前述した FPGA に内蔵されている 18×18 エンデベッド乗算器を用いて構成する。

以下、それぞれの演算器について説明する。

- 32bit 乗算器
  パラメータ f と t の乗算は図 3.10 のようになる。整数部は切り捨て、小数部の上位 16bit のみを結果として出力する。このモジュールには 32bit 乗算器が必要となるが、エンデベッド乗算器は 1 つで 18bit までの乗算しかできない、そのため、エンデベッド乗算器を 4 つと CLB で構成する加算器を組み合わせて 32bit 乗算器を構成する。$f \times t$ の乗算器の構成を 3.11 に示す。演算器の途中にラッチを入れることで 3 段パイプラインとして動作させる。そのため入力から出力までに 2 サイクルのレイテンシが生じる。

- 16bit 加算器
  16bit の加算器を構成。小数部から整数部への桁上げは無視するため、オーバーフローは許される。

- コサイン LUT(Look Up Table)
  前述した Brock SelectRAM を使用したコサイン LUT を使用する。入力は 12 ビット、出力は符号付小数 16bit（２の補数）である。Brock SelectRAM は同期型 RAM なので、入力から出力までに 1 サイクルのレイテンシが生じる

- 符号付 16bit 乗算器 (2 の補数)
  エンデベッド乗算器を 1 個用いて構成する。符号付 16 ビット × 符号なし 16 ビットの乗算となる。エンデベッド乗算器は 18 ビットなので、符号付 16 ビットは上位 2 ビットを符号拡張、符号なし 16 ビットは上位 2 ビットに 0 を付加して使用する。

- 16bit 減算器 (2 の補数)
  16 ビットの減算器。オーバーフローをふせぐために 17 ビット出力としている。

- 17bit 二乗算器 (2 の補数)
  エンデベッド乗算器 1 個で構成する。乗算器の 2 つの入力ポートに同じ値を入れる（符号拡張あり）ことで、二乗演算を行う。出力結果はすべて正の値になるので符号なしとなる。

- 48bit 総和加算
  クロックサイクルごとに入力値を加算する総和加算器。

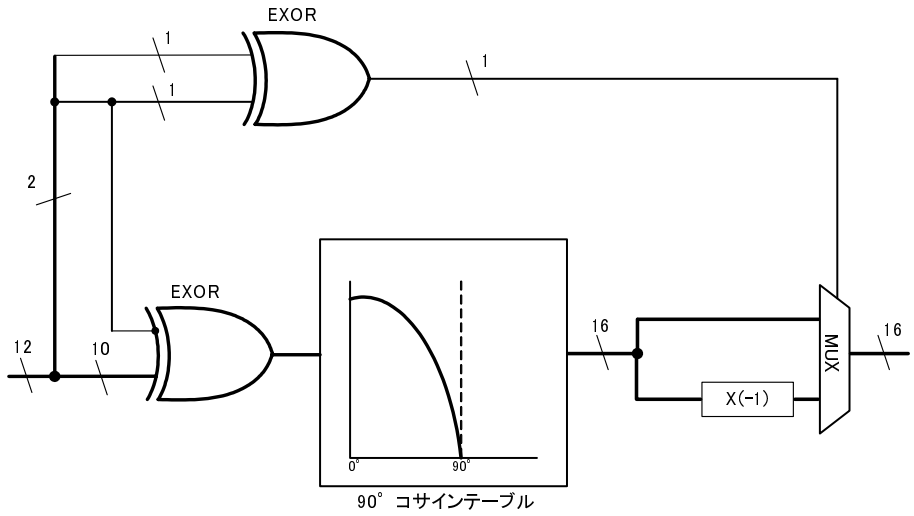


図 3.7: コサイン演算モジュール

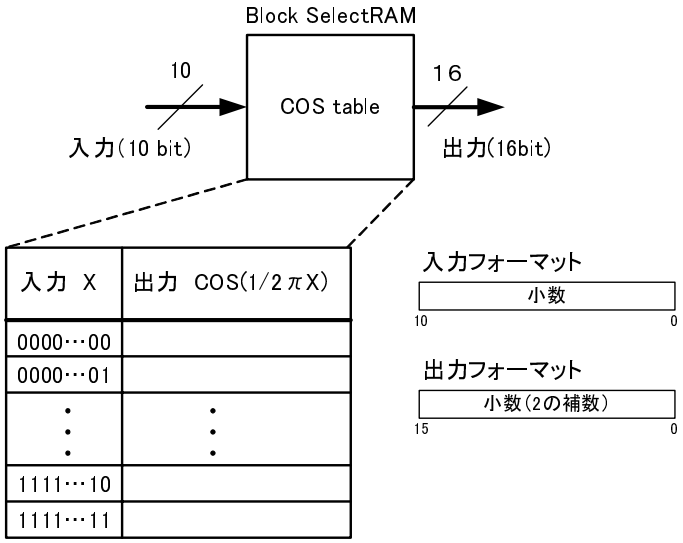


図 3.8: 90 度コサインテーブル

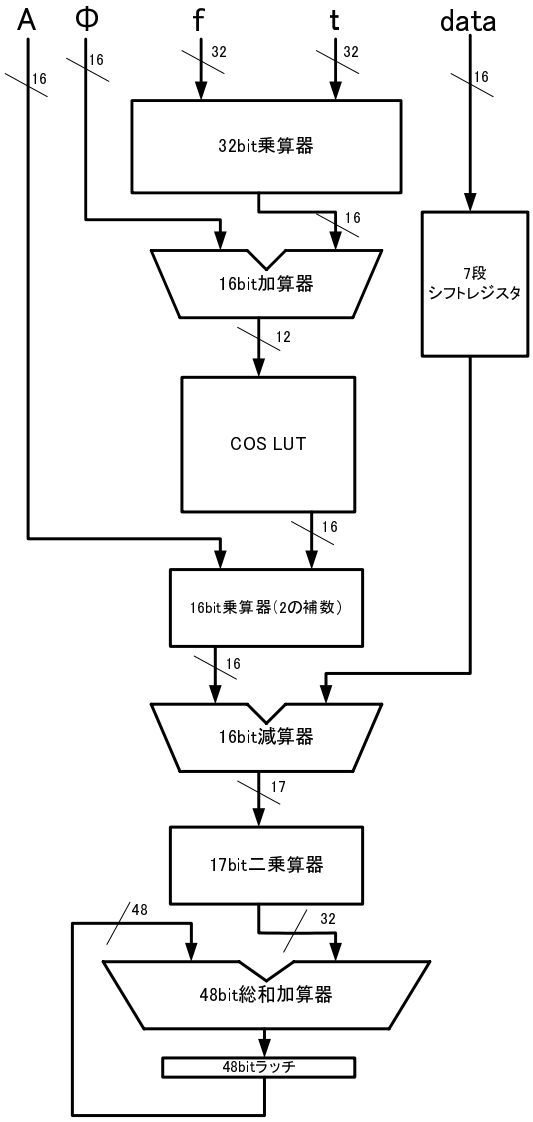


図 3.9: コスト値演算器の構成

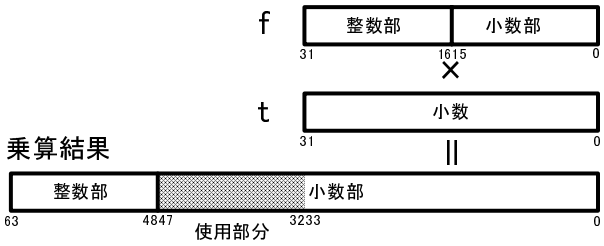


図 3.10: f × t の結果

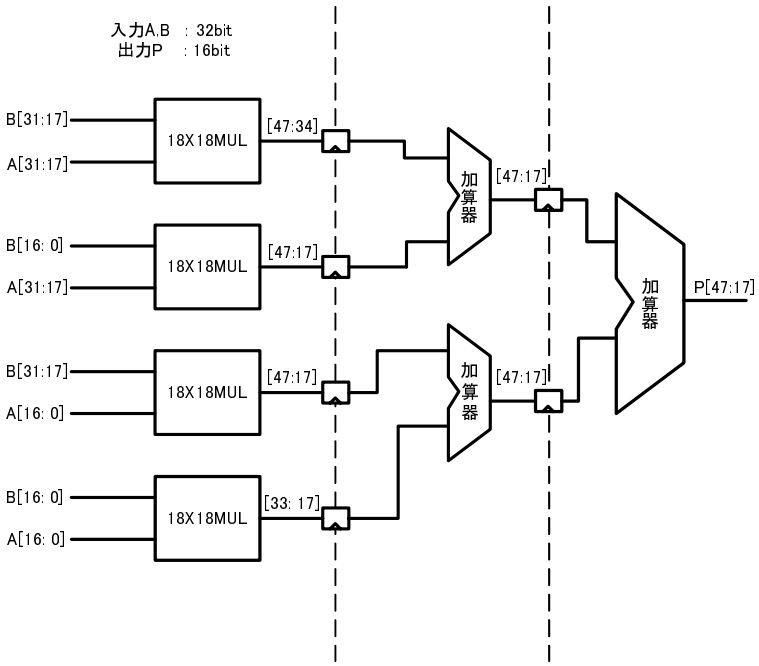


図 3.11: 32bit 入力 16bit 出力 (小数部) の乗算器

- 7段シフトレジスタ
  サンプリングデータ ($data$) の値を入力するタイミングを合わせるために7段16bitのシフトレジスタを設ける。

3.6 データバッファ

3.6.1 データバッファ

ハードウェア内にサンプリングデータ $data$ とそれに対応する時間パラメータ t を保持するバッファが必要である。このデータを格納するために、Block SelectRAM を使用して専用のバッファを構成した（図3.12)、以下これをデータバッファと呼ぶ。データバッファは $data$ 格納用にポートを出力16bitでコンフィグレーションしたBlock SelectRAM、$time$ 用に出力32bitでコンフィグレーションしたBlock SelsctRAMを用いる。Block SelectRAMをカスケード接続することにより、必要なエントリー数のデータバッファを構成することができる。$time$ 用のバッファは $data$ 用のバッファよりエントリ数が少ないため、2倍のBlock SelectRAMが必要となる。

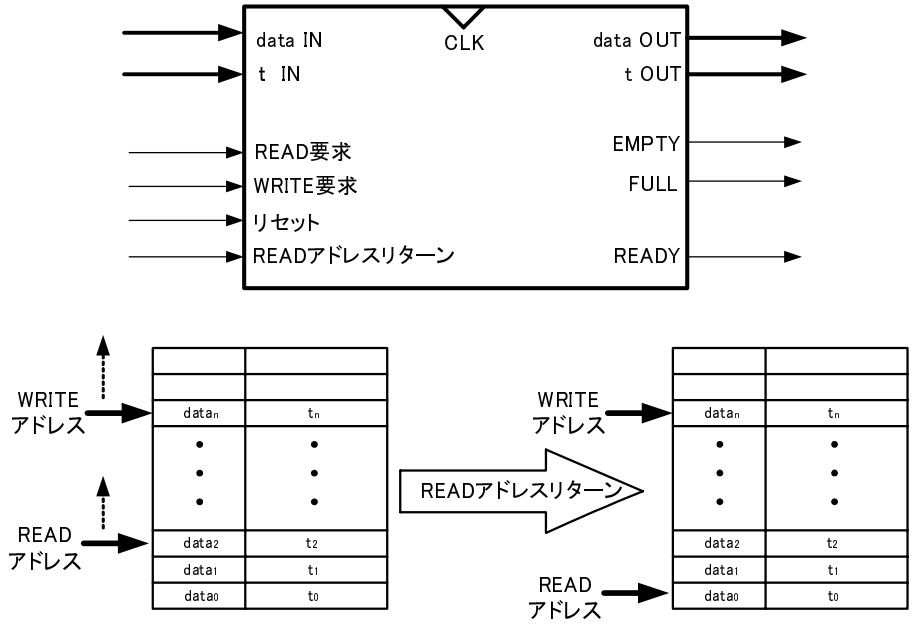


図 3.12: データバッファ

t バッファ用、$data$ バッファ用それぞれのBlock SelectRAMはデュアルポートを用いて構成する。それぞれのポートを書き込み専用のポートと読み込み専用のポートに分け、同クロックでread/writeができるようにする。データバッファはwrite要求がアサートされると、内部のwriteアドレスをインクリメントし、入力データポートの $data$、t の値をそ

のアドレスに書き込む。read 要求がアサートされると、write 要求で書き込まれた値を順番に出力する。FIFO バッファに似た動作をするが、すでに read されたデータを上書きすることなく保持し、強制的に一番はじめに write されたデータに戻って read しなおすことができる。この機能は、1 つの窓長のサンプリングデータに対して、いくつものパラメータ A, f, ϕ でコスト値計算を行うために必要である。write されたデータがすべて read されると empty 信号がアサートされる。またすべての容量がバッファリングされると FULL 信号がアサートされる。READY 信号は read 要求に沿ったデータの出力が成立したときにアサートされる。

3.6.2 コスト値計算器とデータバッファの接続

データバッファとコスト値計算器を組み合わせることで、任意のサンプル数（窓長）のコスト値を計算することができる (図 3.13)。データバッファに要求信号が出されると、その要求信号に応じたデータ f, t と READY 信号が計算器パイプラインに流れる。計算の終了は、エンプティ信号をパイプライン内に伝播させることによって判断する。計算器はパイプラインがストールすることなく動くため、総和加算器で演算とは関係のない値が加算さないように、READY 信号により、総和加算器に加算される計算値とそうでないものを区別する。具体的には総和加算器の直前のラッチで、READY が 1 ならば計算値はそのまま総和加算器に入力し、READY が 0 ならば計算値をリセットして総和加算器に 0 を加算するよう。パイプラインストールは、イネーブル信号を分配させるために生じる配線遅延の影響から、本計算器では導入しない。

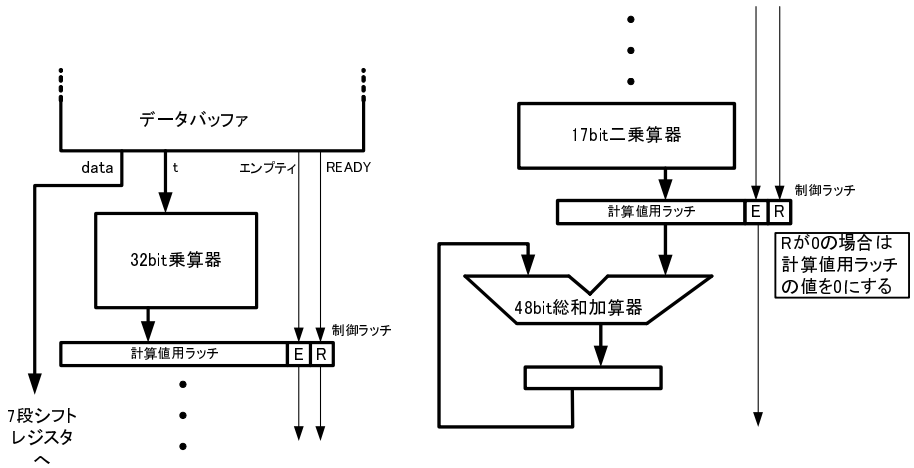


図 3.13: コスト値計算器とデータバッファの接続

3.7 比較器

コスト値が最小となるパラメータ組を推定する過程において、ほとんどの場合、計算途中のコスト値がその時点での最小コスト値の値より大きくなる。この場合そのパラメータ組でのコスト値計算を最後まで行う必要はなく、終了して次パラメータ組の計算を開始するほうが効率的である。それを実現するには、計算途中のコスト値とその時点での最小コスト値を比較し、計算途中結果のほうが大きい場合、終了信号を出力するモジュールが必要である。そのためのモジュールが比較器 (図 3.14) である。また、比較器はパラメータ推定中におけるコスト値の最小値とそのパラメータ A, f, ϕ を、内部レジスタに保持する。レジスタの最小コスト値と計算結果コスト値を比較し、計算結果が小さければそのコスト値とパラメータ A, f, ϕ の値を、比較器内のレジスタに書き込む動作を実現する。

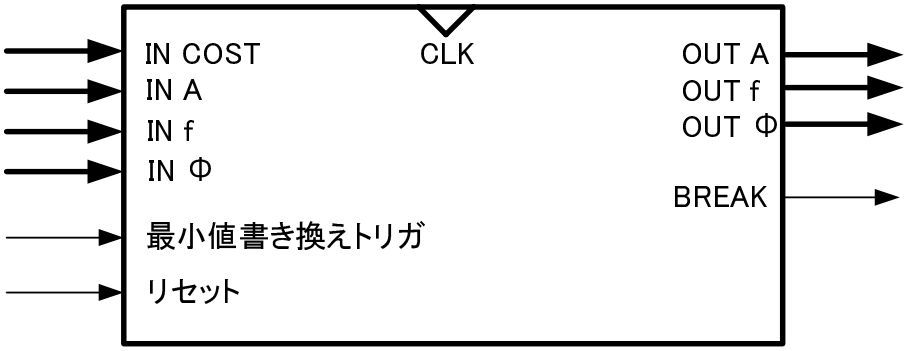


図 3.14: 比較器

各ポートの説明を以下に示す。

- IN COST
  計算中の二乗誤差総和の値が入力される
- IN A、IN f 、IN ϕ
  計算中のパラメータ A, f, ϕ が入力される。
- 最小値書き換えトリガ
  この信号がアサートされると、IN A、IN f、IN ϕ に入力されているパラメータ、IN COST に入力されている二乗誤差総和の値をモジュール内の最小値パラメータレジスタ、最小二乗誤差総和レジスタに書き込む。この信号は、外部の制御部からアサートされる。計算した二乗誤差総和がその時点での最小値より小さかったときのみ行われる。
- リセット
  リセット信号がアサートされると、内部パラメータレジスタは 0x0 に、最小コスト値のレジスタは最大値 0xffffffffffff にセットされる。

- OUT A、OUT f、OUT ϕ
  モジュール内の最小値パラメータレジスタの値が出力される。

- BREAK
  モジュール内の最小二乗誤差総和レジスタより IN COST の値が大きい場合、この信号がアサートされる。アサートされた場合、制御部の指示によってそのパラメータでの計算は終了され、次パラメータでの計算状態に移る。

3.8 パラメータ生成器

パラメータ生成器 (図 3.15) は、一様に分布する推定パラメータ A, f, ϕ を発生させる。パラメータ発生器はパラメータ間隔値、パラメータ分割数、パラメータ絶対値を指定することによって任意の範囲と分割数でパラメータ A, f, ϕ の組み合わせを発生する。生成されるパラメータの総数は、A の分割数 $\times f$ の分割数 $\times \phi$ の分割数である。パラメータはパラメータ要求信号がアサートされるたびに 1 組ずつ出力される。

パラメータ要求がアサートされると、パラメータ生成器は 3 サイクル後に次パラメータを出力する。パラメータ生成器は要求を受け付けてから、次パラメータが出力されるまでの間はパラメータ要求を受け付けることができない。しかし、制御部からパラメータ生成器へ短い間隔でのアクセスは発生しないため、要求待ちは考慮しなくてよい。各ポートの

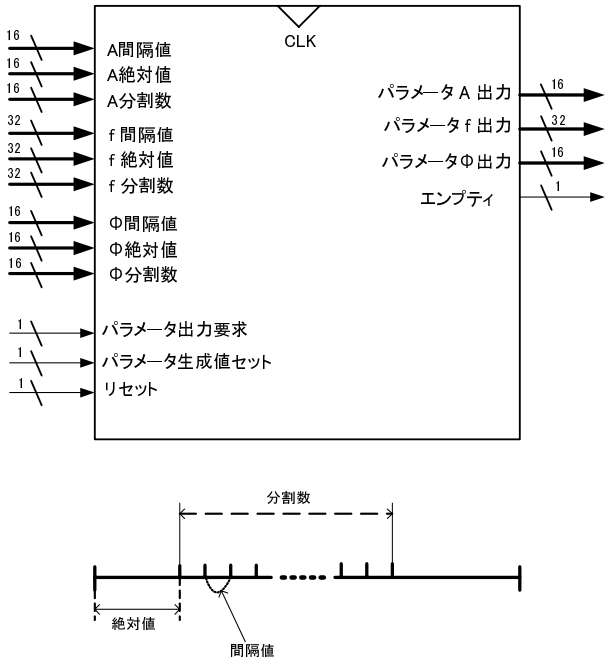


図 3.15: パラメータ生成器

説明を下記に示す。

- A, f, ϕ分割数
  各パラメータの間隔値、絶対値によって指定した範囲を、いくつに分割するかを示す値、A, f, ϕの分割数はそれぞれ、16bit、32bit、16bit の整数値で指定される。CPU により FPGA チップの外部から指定される。

- A, f, ϕ間隔値
  各パラメータの間隔値を示す。A, f, ϕの間隔値は図 3.6 に示したそれぞれのパラメータと同じフォーマットで指定される。CPU により FPGA チップの外部から指定される。

- A, f, ϕ絶対値
  範囲指定をするために、それぞれのパラメータの分割をどこから始めるかを示す定数値。A, f, ϕの絶対値はそれぞれ 16bit、32bit、16bit の整数値で指定される。CPU により FPGA チップの外部から指定される。

- パラメータ出力要求
  次パラメータ組を呼び出すための入力。制御部の指示により信号がアサートされる。

- パラメータ生成値セット
  この信号がアサートされると、A, f, ϕの分割数、間隔値、絶対値に入力されている信号を、パラメータ生成器内部のレジスタに取り込む。FPGA チップの外部から指定される。

- リセット
  パラメータ生成器内部のレジスタに格納されている A, f, ϕの値をリセットする。

- A, f, ϕ出力
  各パラメータ出力ポート

- エンプティ
  指定されたすべての組み合わせの A, f, ϕのパラメータ組を出力し終わったときにアサートされる。

3.9 全体の制御

最小コスト値となるパラメータ推定を実現するには、多数のパラメータ組での計算と最小コスト値の管理を行う必要がある。そのためには、コスト値計算器、パラメータ生成器、データバッファ、比較器を制御する制御部が必要である

制御の流れを以下に示す。

- FPGA 外部から、パラメータ生成用の値と信号データが書き込まれ、計算スタート信号により最小値パラメータ推定が開始される。

- データ要求信号を出しデータバッファからコスト値演算器に信号データが流れることで、コスト値計算が行われる。データバッファのエンプティ信号がコスト値演算器のパイプラインの最終点まで伝播すると、データ要求信号を止め、比較器の最小値書き換えトリガをアサートし、最小値の更新を行い、次パラメータ要求を出し、データバッファのの read アドレスを初期番地に戻すために read アドレスリターン信号をアサートする。計算途中結果が比較器内の最小コスト値より大きくなりブレーク信号がアサートされると、計算途中で次パラメータを呼び出し、read アドレスリターン信号をアサートする。

- 次パラメータの呼び出し待ちが生じる。パラメータの呼び出しが行われると、新たなパラメータで、計算を開始する。パラメータ生成器がすべてのパラメータを生成し終えて、エンプティ信号がアサートされるとパラメータ推定が終了し、FPGA 外部に終了信号を出す。そのときの比較器内にある最小値パラメータレジスタの値が推定された最小コスト値のパラメータとなる。

以上から、制御部の状態遷移図を図 3.16 に示す。

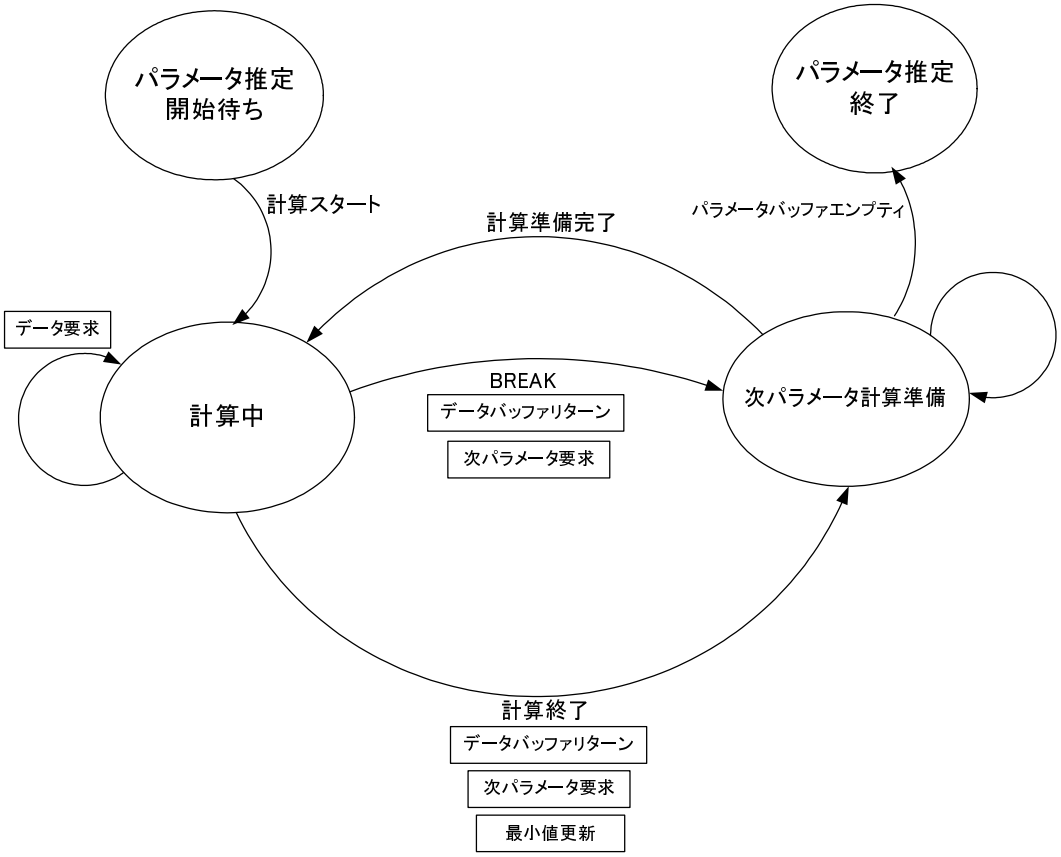


図 3.16: 状態遷移図

3.10　コスト値計算器の並列化

3.10.1　コスト値計算器の並列化

前述したコスト値計算器では、対象信号とモデル信号の二乗誤差を１サンプルごとに演算して総和加算を行い、コスト値を求めていた。この方法だと一つの窓長のコスト値を求めるために、(窓長サンプル数＋パイプライン段数) 分のサイクル数が必要である。そこで、演算器の並列化を行いスループットの向上を図る。このコスト演算はサンプル間同士の依存がないことから、演算器の並列化が可能である。1 サイクルで並列度に応じたサンプル分の二乗誤差を同時に求めることができる。(図 3.17)

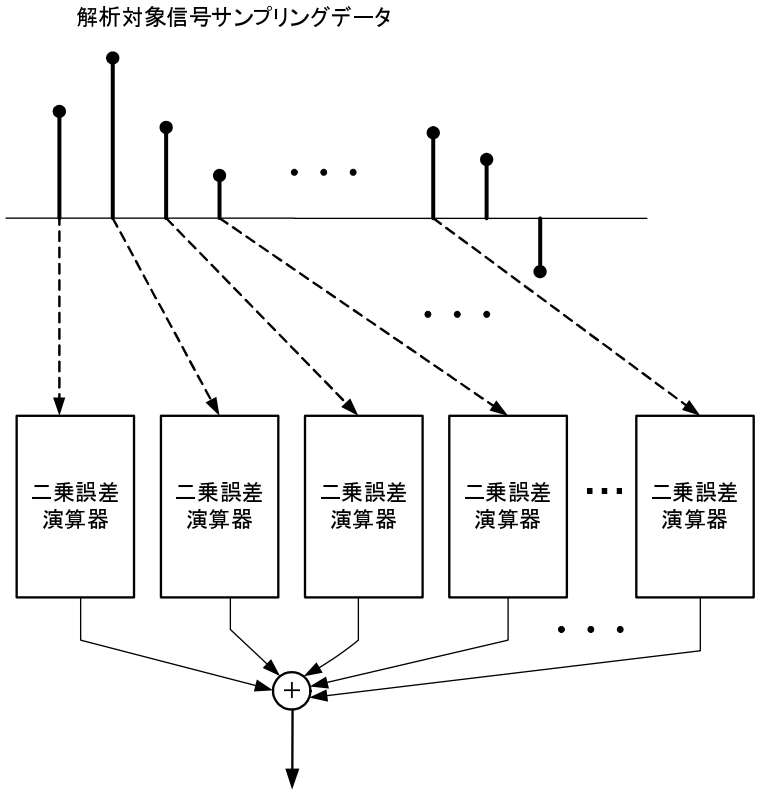


図 3.17: 演算器の並列化

3.10.2　4 並列パイプライン二乗誤差演算器

まず、4 並列パイプライン二乗誤差演算器を構成した。(図 3.18)。パイプライン段数はシングルパイプラインに比べ、２段増加するが、1 サイクルに 4 サンプル分の二乗誤差総和演算が可能である。それぞれの演算器には同じ値のパラメータ A, f, Φ が入力される。

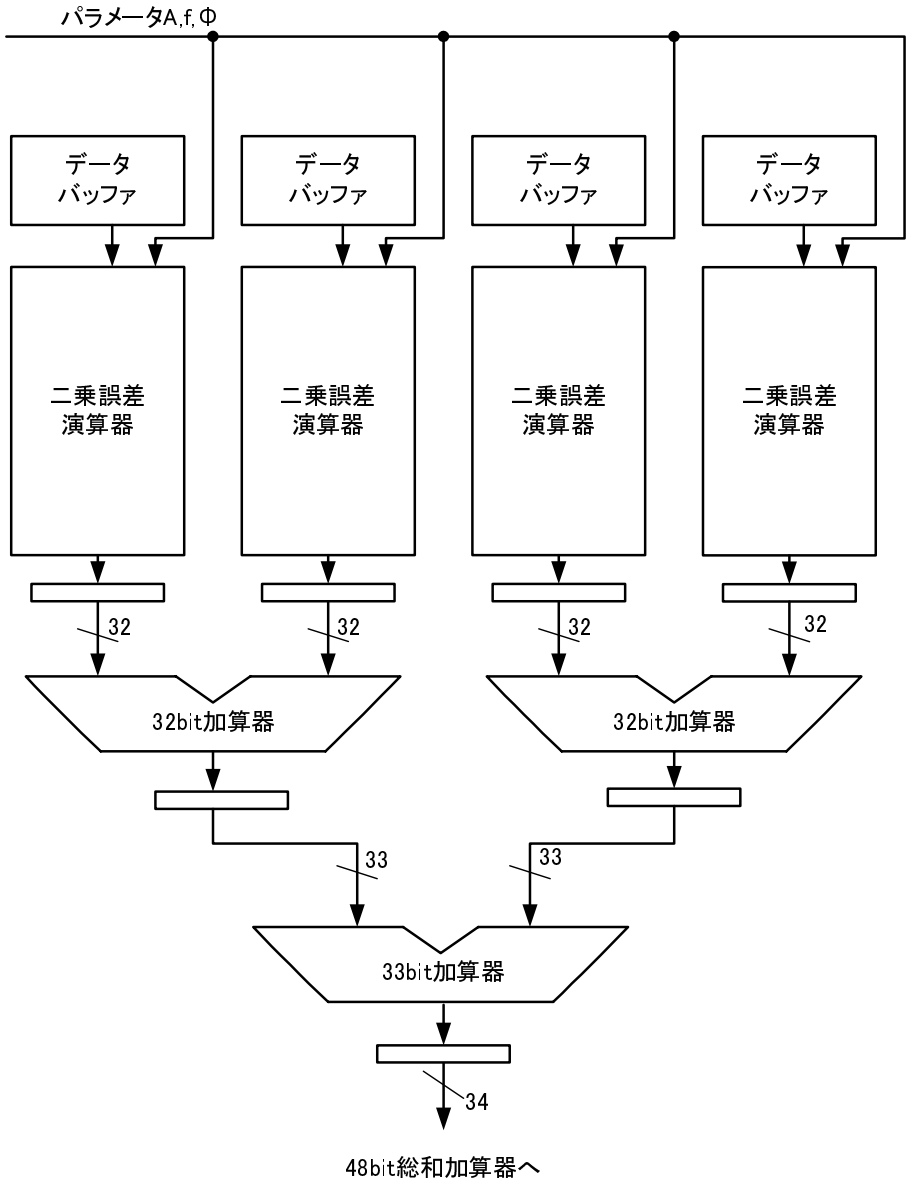


図 3.18: 4 並列二乗誤差演算器

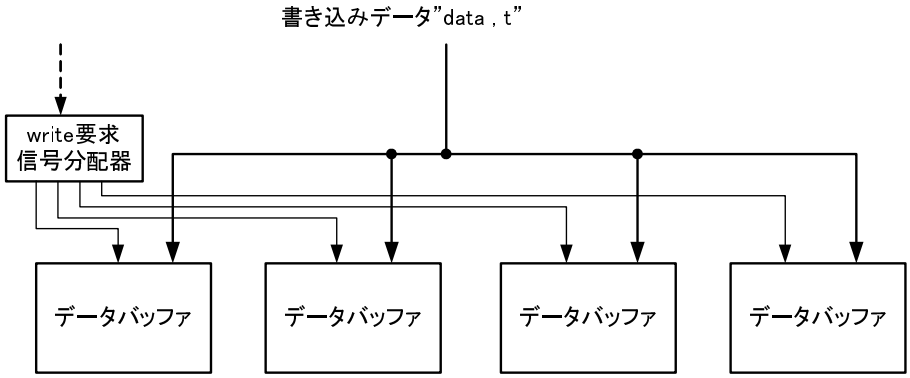


図 3.19: データ分配

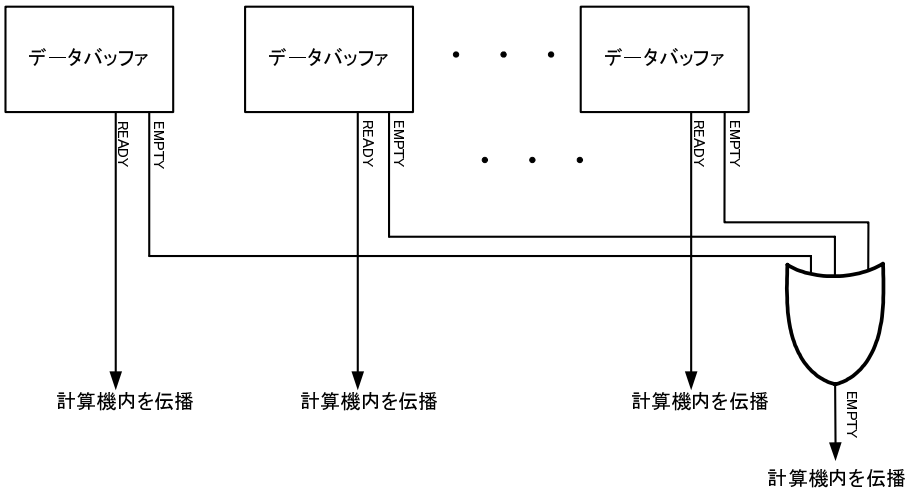


図 3.20: 並列計算器における制御信号の伝播（エンプティ）

それぞれの演算器のデータバッファには振り分けられたサンプリングデータが保持されている必要がある。そのために、データ write 要求分配器を用意し、データを書き込む際にデータ書き込み要求信号をそれぞれのデータバッファに分配して出すことにより、サンプリングデータを振り分けて書き込むことができる。(図 3.19)

演算器内に制御用として伝播させていたデータバッファのエンプティ信号は、それぞれデータバッファ出口で OR ゲートでまとめて、以下パイプラインを伝播させる。(図 3.20) また、READY 信号はそれぞれ二乗誤差演算器の計算結果ラッチの出力から OR ゲートでまとめて、以下伝播させる (図 3.21)。

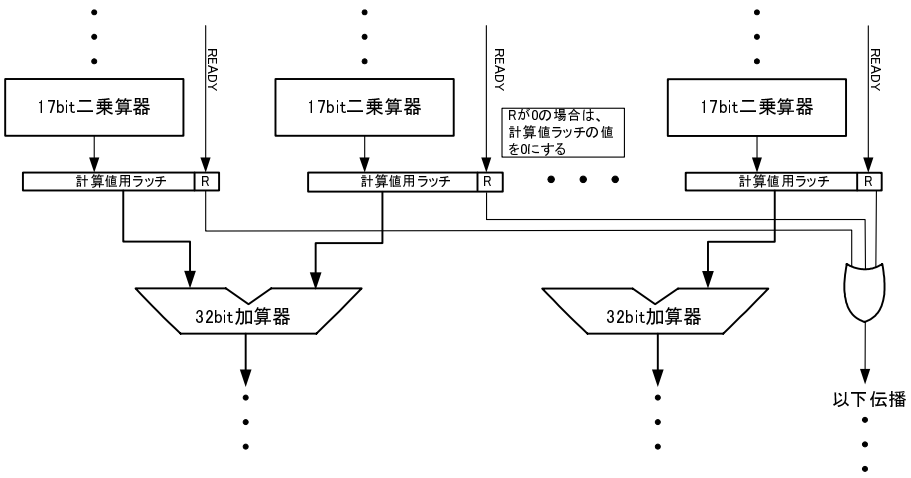


図 3.21: 並列計算器における制御信号の伝播（READY）

3.10.3 16 並列二乗誤差演算器

4 並列パイプライン二乗誤差演算器をさらに 4 つ使用して、16 並列パイプラインの二乗誤差計算器を構成した。（図 3.22）

それぞれの 4 並列演算器内にサンプリングデータを振り分けるために、データ書き込み要求をさらに分配する。データバッファから出力される制御用信号は 4 並列計算器の場合と同様な方法 (図 3.20、図 3.21) で伝播させる。

3.11 合成結果

本章で述べた一般化調和解析ハードウェアを Verilog-HDL で記述し、Xilinx 社の FPGA 統合開発ツール ISE version 6.3i で論理合成を試みた。表 3.2 にその結果、スライス数、動作周波数を示す。ISE でマッピング後の結果を示す。

並列度が大きくなるに従い、動作周波数が低下する傾向にある。これは並列化によるハードウェア量増大に伴い FPGA 内部の配線遅延が増大するためである。

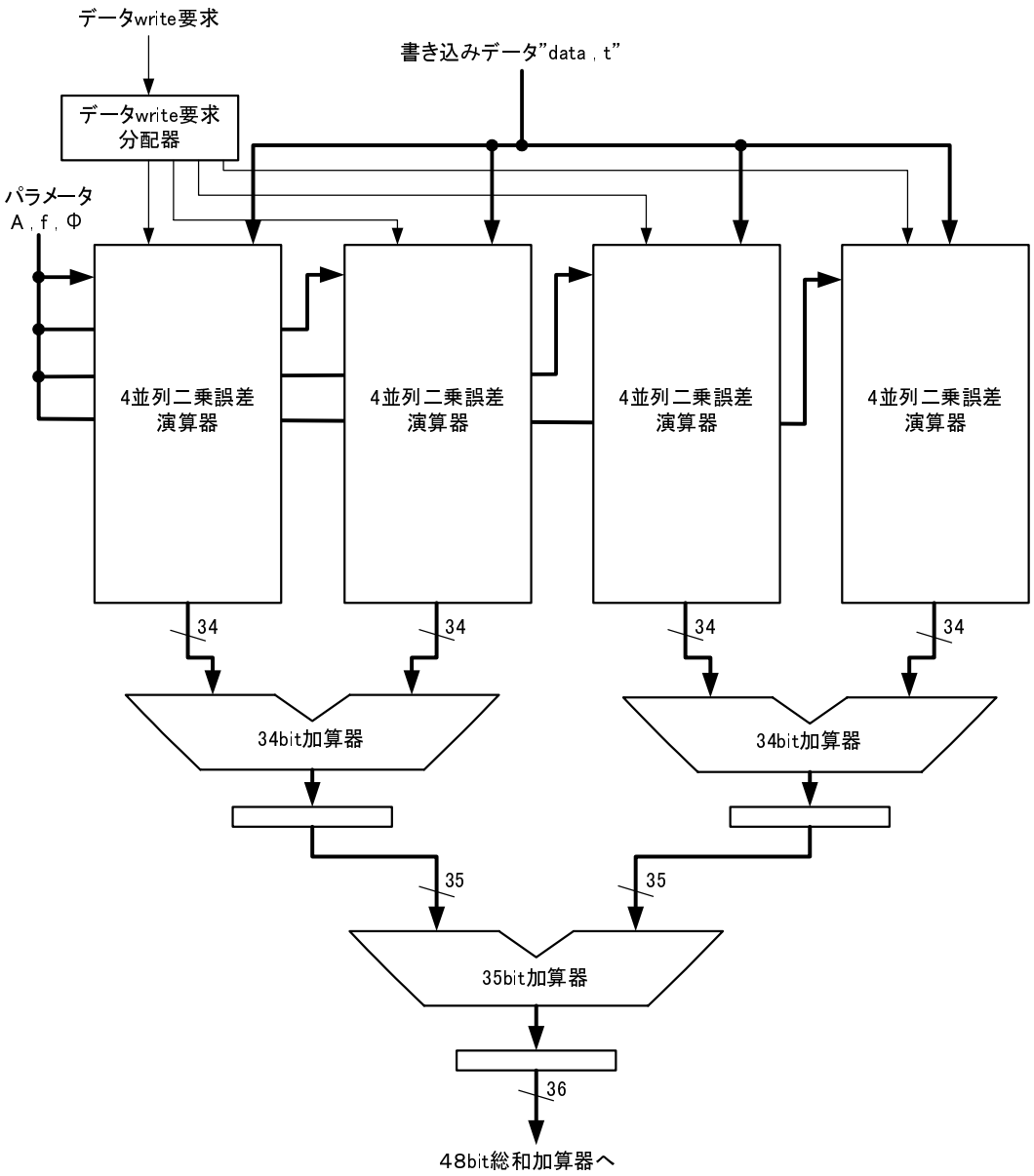


図 3.22: 16 並列パイプライン二乗誤差演算器

並列度 [way]	Slice 数 [slice]	動作周波数 [MHz]
1	911	99.1
4	2228	71.6
8	4115	54.6
12	5910	56.0
16	7242	48.0

表 3.2: 合成結果

第4章　ソフトウェアとの連携

4.1　ソフトウェアとハードウェアのパイプライン化

本論文ではハードウェアとソフトウェアの実行を並列に行い、パイプライン化することが可能である。

信号の解析は短いフレームに区切って行われる。一般化調和解析ではフレーム内で１組目のパラメータを抽出するとそれを原信号から差し引き、その残差信号に対して再度パラメータを抽出する。ソフトウェア処理とハードウェア処理のパイプラインに関して、１つのフレーム内でパラメータを連続して抽出すると、パラメータ抽出結果待ちの依存が発生し、パイプライン化することができない。

そこで、本論分では依存関係のないフレーム同士の処理のパイプライン化を行う。具体的には、現在のフレームから１組のパラメータを抽出し終えたら、次のフレームに移りパラメータを１組抽出するといった処理を行う。

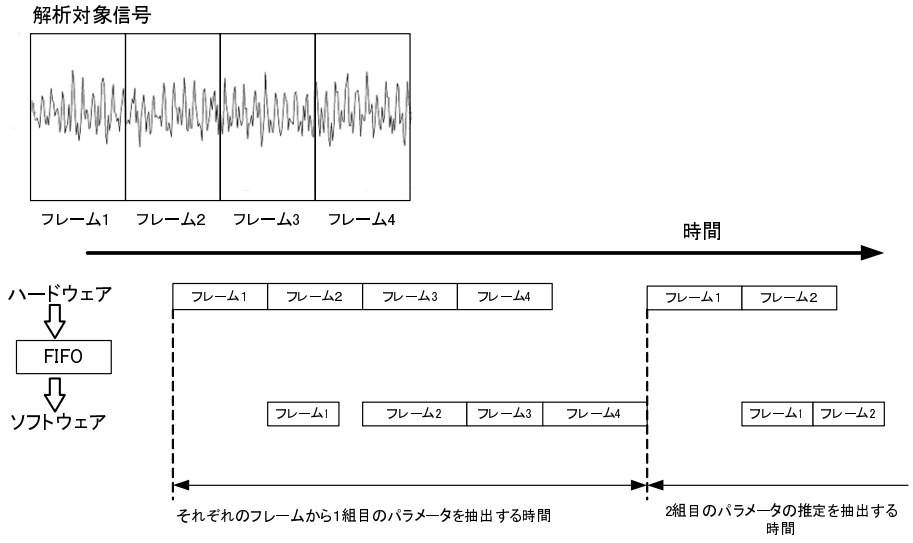


図 4.1: ソフトウェアとハードウェアのパイプライン化

ハードウェアとソフトウェアのパイプライン化について図 4.1 に示す。ハードウェアの結果は、計算の速度差を吸収するために設けられたソフトウェアによる FIFO バッファを

介して、ソフトウェアに渡される。ハードウェアは 1 番目のフレームの処理が終わると、FIFO にその推定値を入れ、2 番目のフレームの処理を始める。ソフトウェアはハードウェアが推定したパラメータ値を FIFO から取り出し、それを初期値としてニュートン法によるパラメータの推定を行う。ハードウェアは任意に定めた数のフレーム処理が終わると、ソフトウェアの実行を待ち、それぞれのフレームから 2 組目のパラメータ抽出を行う。

4.2 ソフトウェア処理の手順

4.2.1 ソフトウェア処理の手順

ハードウェアによって大雑把に推定されたパラメータを初期値として、ソフトウェアによる非線形最小二乗法の解法を適用する。非線形最小二乗法の解法として、本論文では減速ニュートン法を用いる。パラメータの推定手順を図 4.2 に示す。まず、f、ϕ をハードウェアから受け取った値に固定して、パラメータ A に対して 1 変数の減速ニュートン法を適用し推定を行う。その後、A を固定して、パラメータ f、ϕ の推定を行う。3 つのパラメータ A、f、ϕ の同時推定を減速ニュートン法で行うには、初期値が真値のごく近傍になければ収束しないため、この 2 回の手順により 3 つのパラメータを真値のごく近くまで近づける。更にそれを初期値として、3 変数のニュートン法を適用し、3 パラメータ A、f、ϕ を同時に解へ収束させることができる。

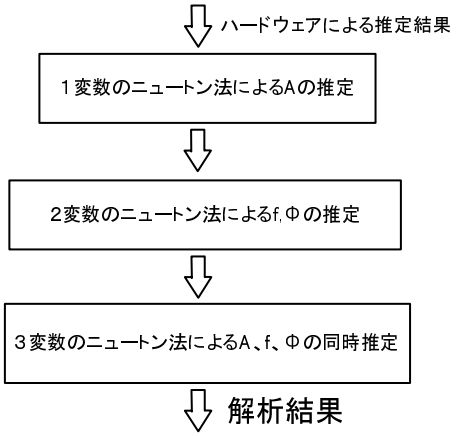


図 4.2: ソフトウェア処理のパラメータ推定手順

4.2.2 収束範囲の評価

3

ニュートン法によるパラメータ値推定を本手法で用いる 3 回の手順で行った場合と、三変数のニュートン法のみで行った場合の収束範囲の評価を行った。周波数構成が既知であ

る信号を用いて、初期値のパラメータ組を等間隔にばら撒き、収束範囲 A が ± 0.05 以内、f が ± 1[Hz] 以内 ϕ が 0.05[rad] 以内に収束したパラメータ組をカウントする。既知信号は長さ 512 サンプル $A = 0.3$、$f = 450$、$\phi = 0.3$ の正弦波信号である。推定範囲ろ $A = 0 \sim 1$、$f = 350 \sim 550$、$\phi = 0 \sim 1$ とし、範囲内で f を 200 分割、A、ϕ を 10 分割したパラメータ総数 20000 組を初期値とした。。図 4.3(a)(b) に結果を示す。ニュートン法の初期値としてパラメータ推定を正確に行えたパラメータ組がプロットされている。総数 20000 個の初期値において、パラメータ推定が正確に行えたパラメータ組の数は、3 手順に分けた推定では 5069 組、A、f、ϕ 同時に収束させた場合には 103 組となった。パラメータ推定を 3 手順に分けることで、収束範囲を広げることができる。

4.3 減速ニュートン法の適用

4.3.1 3パラメータ同時推定

本論文で使用した減速ニュートン法について説明する。ここでは 3 変数の場合の減速ニュートン法について説明する。解析窓長を N, 解析データを $data_i$ としてコスト関数は次のように表される

$$W(A, F, \phi) = \sum_{i=0}^{N-1} \left\{ A\cos(2\pi(ft_i + \phi)) - data_i \right\} \tag{4.1}$$

この関数が極地をとるためには、

$$\begin{cases} \dfrac{\partial W(A,F,\phi)}{\partial A} = 0 \\ \dfrac{\partial W(A,F,\phi)}{\partial F} = 0 \\ \dfrac{\partial W(A,F,\phi)}{\partial \phi} = 0 \end{cases} \tag{4.2}$$

が成り立てばよい、4.2 のごく近くの点 (A_n, F_n, ϕ_n) で 1 次までのテイラー展開を行うと、

$$\begin{cases} \dfrac{\partial W(A_{n+1},F_{n+1},\phi_{n+1})}{\partial A} = \dfrac{\partial W(A,F,\phi)}{\partial A} + \Delta A\dfrac{\partial^2 W(A,F,\phi)}{\partial A^2} + \Delta F\dfrac{\partial^2 W(A,F,\phi)}{\partial A\partial F} + \Delta\phi\dfrac{\partial^2 W(A,F,\phi)}{\partial A\partial\phi} \\ \dfrac{\partial W(A_{n+1},F_{n+1},\phi_{n+1})}{\partial F} = \dfrac{\partial W(A,F,\phi)}{\partial F} + \Delta A\dfrac{\partial^2 W(A,F,\phi)}{\partial A\partial F} + \Delta F\dfrac{\partial^2 W(A,F,\phi)}{\partial F^2} + \Delta\phi\dfrac{\partial^2 W(A,F,\phi)}{\partial F\partial\phi} \\ \dfrac{\partial W(A_{n+1},F_{n+1},\phi_{n+1})}{\partial A} = \dfrac{\partial W(A,F,\phi)}{\partial \phi} + \Delta A\dfrac{\partial^2 W(A,F,\phi)}{\partial A\partial\phi} + \Delta F\dfrac{\partial^2 W(A,F,\phi)}{\partial F\partial\phi} + \Delta\phi\dfrac{\partial^2 W(A,F,\phi)}{\partial \phi^2} \end{cases} \tag{4.3}$$

ここで,

$$\Delta A = A_{n+1} - A_n \ , \ \Delta F = F_{n+1} - F_n \ , \ \Delta\phi = \phi_{n+1} - \phi_n$$

である。4.3 に 4.2 を代入すると次の方程式が得られる

$$\begin{pmatrix} \dfrac{\partial^2 W(A,F,\phi)}{\partial A^2} & \dfrac{\partial^2 W(A,F,\phi)}{\partial A \partial F} & \dfrac{\partial^2 W(A,F,\phi)}{\partial A \partial \phi} \\ \dfrac{\partial^2 W(A,F,\phi)}{\partial A \partial F} & \dfrac{\partial^2 W(A,F,\phi)}{\partial F^2} & \dfrac{\partial^2 W(A,F,\phi)}{\partial F \partial \phi} \\ \dfrac{\partial^2 W(A,F,\phi)}{\partial A \partial \phi} & \dfrac{\partial^2 W(A,F,\phi)}{\partial F \partial \phi} & \dfrac{\partial^2 W(A,F,\phi)}{\partial \phi^2} \end{pmatrix} \begin{pmatrix} \Delta A \\ \Delta F \\ \Delta \phi \end{pmatrix} = - \begin{pmatrix} \dfrac{\partial W(A,F,\phi)}{\partial A} \\ \dfrac{\partial W(A,F,\phi)}{\partial F} \\ \dfrac{\partial W(A,F,\phi)}{\partial \phi} \end{pmatrix} \tag{4.4}$$

$\Delta A, \Delta F, \Delta \phi$ について解くと

$$\begin{cases} \Delta A &= -\dfrac{1}{J}\left\{(W_{ff}W_{\phi\phi} - W_{f\phi}^2)W_a + (W_{a\phi}W_{f\phi} - W_{af}W_{\phi\phi})W_f + (W_{af}W_{f\phi} - W_{a\phi}W_{f\phi})W_\phi\right\} \\ \Delta F &= -\dfrac{1}{J}\left\{(W_{a\phi}W_{f\phi} - W_{af}W_{\phi\phi})W_a + (W_{aa}W_{\phi\phi} - W_{a\phi}^2)W_f + (W_{af}W_{a\phi} - W_{aa}W_{f\phi})W_\phi\right\} \\ \Delta F &= -\dfrac{1}{J}\left\{(W_{af}W_{f\phi} - W_{ff}W_{a\phi})W_a + (W_{af}W_{a\phi} - W_{aa}W_{f\phi})W_f + (W_{aa}W_{ff} - W_{af})^2\right\} \end{cases} \tag{4.5}$$

$$J = W_{aa}W_{ff}W_{\phi\phi} + 2W_{af}W_{a\phi}W_{f\phi} - W_{a\phi}^2 W_{ff} - W_{f\phi}^2 W_{aa} - W_{af}^2 W_{\phi\phi}$$

ここで、

$$\begin{aligned} W_a &= \frac{\partial W(A,F,\phi)}{\partial A} &, W_f &= \frac{\partial W(A,F,\phi)}{\partial F} &, W_\phi &= \frac{\partial W(A,F,\phi)}{\partial \phi} \\ W_{aa} &= \frac{\partial^2 W(A,F,\phi)}{\partial A^2} &, W_{af} &= \frac{\partial^2 W(A,F,\phi)}{\partial A \partial F} &, W_{a\phi} &= \frac{\partial^2 W(A,F,\phi)}{\partial A \partial \phi} \\ W_{ff} &= \frac{\partial^2 W(A,F,\phi)}{\partial F^2} &, W_{f\phi} &= \frac{\partial^2 W(A,F,\phi)}{\partial F \partial \phi} &, W_{\phi\phi} &= \frac{\partial^2 W(A,F,\phi)}{\partial \phi^2} \end{aligned}$$

である。一般化調和解析コスト関数にあてはめると、以下のようになる。

$$W_a = \sum_{i=0}^{N-1} \left\{ 2\cos(2\pi(Ft_i + \phi)) \cdot (A\cos(2\pi(ft_i + \phi)) - data_i) \right\} \tag{4.6}$$

$$W_f = \sum_{i=0}^{N-1} \left\{ -4A\pi t_i(\sin(2\pi(Ft_i + \phi))) \cdot (A\cos(2\pi(ft_i + \phi)) - data_i) \right\} \tag{4.7}$$

$$W_\phi = \sum_{i=0}^{N-1} \left\{ -4A\pi(\sin(2\pi(Ft_i + \phi))) \cdot (A\cos(2\pi(ft_i + \phi)) - data_i) \right\} \tag{4.8}$$

$$W_{aa} = \sum_{i=0}^{N-1} \left\{ 2\cos^2(2\pi(Ft_i + \phi)) \right\} \tag{4.9}$$

$$W_{af} = \sum_{i=0}^{N-1} \left\{ -4\pi t_i \sin(2\pi(Ft_i + \phi)) \cdot (2A\cos(2\pi(ft_i + \phi)) - data_i) \right\} \tag{4.10}$$

$$W_{a\phi} = \sum_{i=0}^{N-1} \left\{ -4\pi \sin(2\pi(Ft_i + \phi)) \cdot (2A\cos(2\pi(ft_i + \phi)) - data_i) \right\} \tag{4.11}$$

$$W_{ff} = \sum_{i=0}^{N-1} \left\{ -8At_i^2\pi^2 \cos(2\pi(ft_i + \phi)) \cdot (A\cos(2\pi(ft_i + \phi)) + A\sin^2(2\pi(ft_i + \phi)) - data_i) \right\} \tag{4.12}$$

$$W_{f\phi} = \sum_{i=0}^{N-1} \left\{ -8At_i\pi^2 \cos(2\pi(ft_i + \phi)) \cdot (A\cos(2\pi(ft_i + \phi)) + A\sin^2(2\pi(ft_i + \phi)) - data_i) \right\} \tag{4.13}$$

$$W_{\phi\phi} = \sum_{i=0}^{N-1} \left\{ -8A\pi^2 \cos(2\pi(ft_i + \phi)) \cdot (A\cos(2\pi(ft_i + \phi)) + A\sin^2(2\pi(ft_i + \phi)) - data_i) \right\} \tag{4.14}$$

$$\tag{4.15}$$

ここから、ΔA、ΔF、$\Delta\phi$ を修正量とする、非線形方程式に対する反復公式

$$\begin{cases} A_{n+1} = A_n + \Delta A \\ F_{n+1} = F_n + \Delta F \\ \phi_{n+1} = \phi_n + \Delta\phi \end{cases} \tag{4.16}$$

を得る。これがニュートン法である。しかし、このまま修正量を使用して反復計算を行うと収束安定性が悪い。ニュートン法の収束を安定化する方法として、減速係数を導入する

手法がある（減速ニュートン法)。減速係数を μ とすると、

$$\begin{cases} A_{n+1} = A_n + \mu_n \Delta A \\ F_{n+1} = F_n + \mu_n \Delta F \quad (0 < \mu_n < 1) \\ \phi_{n+1} = \phi_n + \mu_n \Delta \phi \end{cases} \tag{4.17}$$

具体的に減速係数 μ は次のような使い方をする。

1. $\mu_n = 1$ と置く。式 4.17 により $\Delta A, \Delta F, \Delta \phi$ を計算する。
2. $A_{tmp} = A_n + \mu_n \Delta A, F_{tmp} = F_n + \mu_n \Delta F, \phi_{tmp} = \phi_n + \mu_n \Delta \phi$ とする。このとき条件

   $$W(A_{tmp}, F_{tmp}, \phi_{tmp}) < W(A_n, F_n, \phi_n) \tag{4.18}$$

   が満足されれば 4. へ, そうでなかったら 3. へ
3. μ_n の値を半分にして 2. へ
4. $A_{n+1} = A_{tmp}, F_{n+1} = F_{tmp}, \phi_{n+1} = \phi_{tmp}$ とする

μ が反復計算のたび確実にコスト関数を減少させる働きをするので、パラメータ A、f、ϕ が局所的な解に近づくことが期待できる。

4.3.2 パラメータ F、ϕ と A を分けて推定

パラメータ A を固定して、ニュートン法で f, ϕ を推定する場合も 3 パラメータ同時推定と同様な方法で、修正量 Δf、$\Delta \phi$ を求めることができる。後は前述したように減速係数を使用した推定法でパラメータ f、ϕ の値を反復計算により求めてゆく

$$\begin{cases} \Delta F = -\dfrac{1}{J}\left(W_F W_{\phi\phi} - W_\phi W_{F\phi}\right) \\ \Delta \phi = -\dfrac{1}{J}\left(W_{FF} W_\phi - W_F W_{F\phi}\right) \end{cases} \tag{4.19}$$

$J = W_{FF} W_{\phi\phi}$

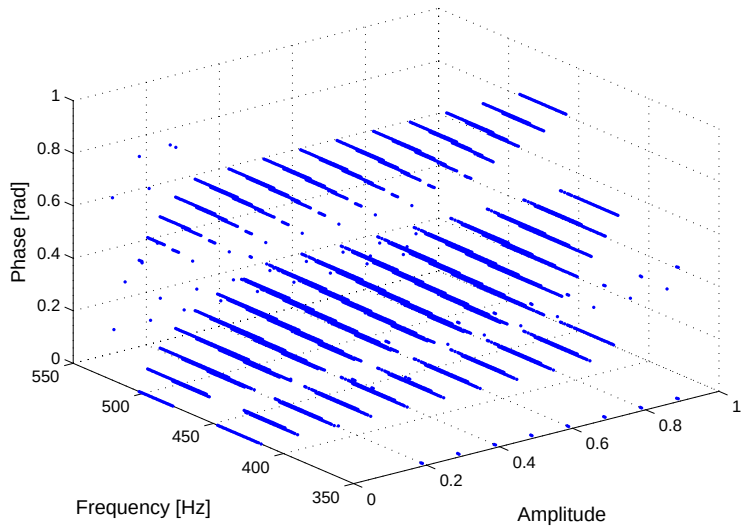
パラメータ A のみのニュートン法も同様に修正量を求めて反復計算を行う。

$$\Delta A = -\frac{W_a}{W_{aa}} \tag{4.20}$$

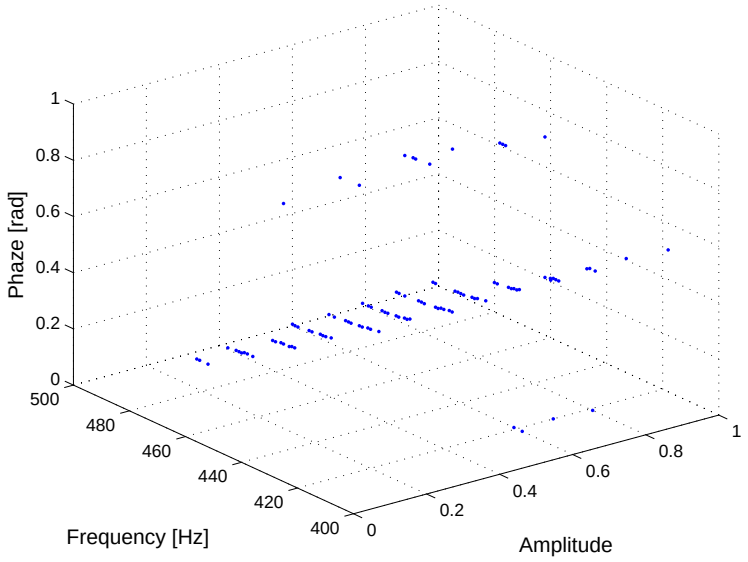
4.4 実装

実装した減速ニュートン法でパラメータ A、f、ϕ を同時に収束させたときのフローチャートを図 4.4 に示す。f、ϕ のみの推定、A のみの推定の場合も同様な処理で実装を行う。実装は倍制度の浮動小数点演算によって行う。パラメータの収束を判断するために収束判定値 er を導入する。この er の値とあらかじめ定めた収束精度を比較し、収束精度より er の値が小さくなった場合、計算を打ち切る。すなわち、反復計算においてパラメータ A、f、ϕ の変化量が少なくなったところで、パラメータが収束したとみなす。

また、減速係数 μ に縮小制限をかけ、減速係数が小さくなりすぎると計算を打ち切り、それまでに求まっていたコスト値が最小となるパラメータを結果として出力する。



(a) 3 回の手順に分けたパラメータ推定



(b) 3 パラメータを同時に推定

図 4.3: パラメータ推定が解に収束した初期値の分布

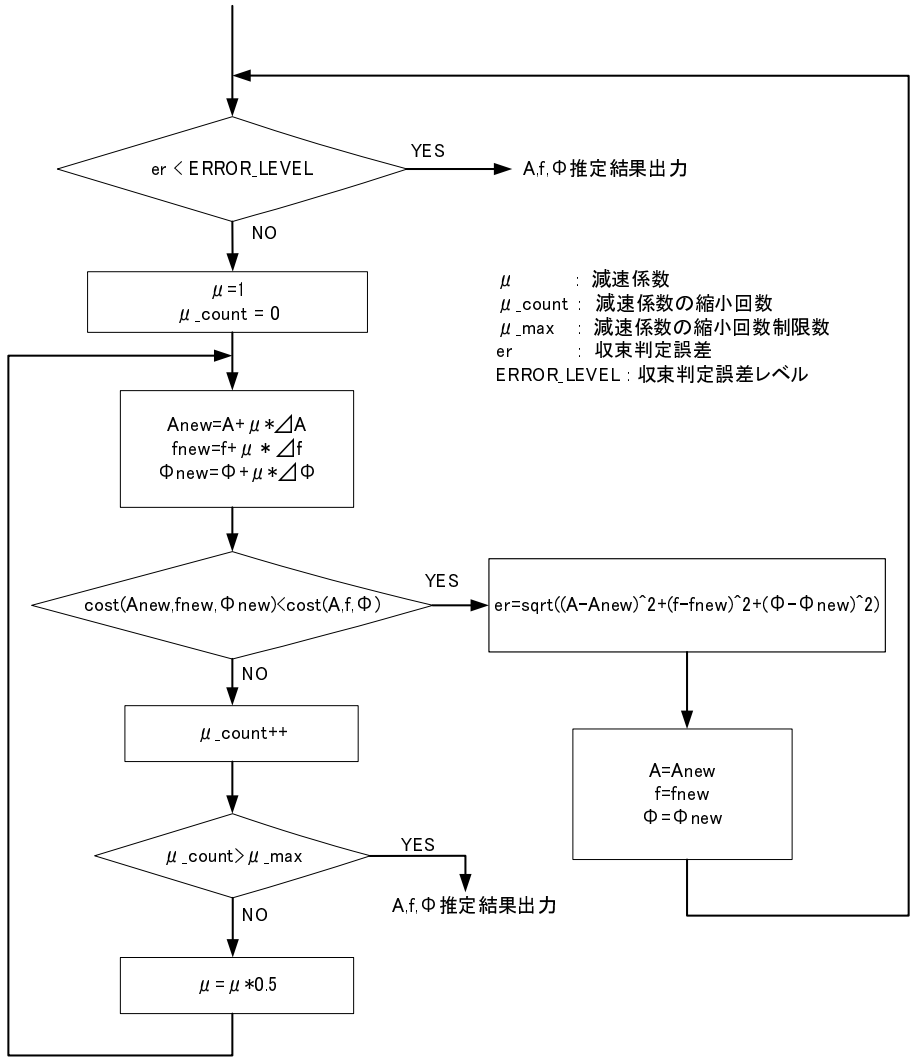


図 4.4: パラメータ A、f、ϕ を同時収束させる減速ニュートン法のフローチャート

第5章　評価

本論文で提案した手法の解析速度と解析精度を評価する。ソフトウェアの実行に用いたPC の規格を表 5.1 に示す

DELL OPTIPLEX GX60	
CPU	Intel Celeron 1.7GHz
RAM	DDR SDRAM 512MB

表 5.1: ソフトウェア実行に用いた PC の規格

また、評価データはすべて、サンプリング周波数 44100Hz、16bit で量子化された PCM データを扱う。

5.1　ハードウェア部の速度評価

提案手法の固定小数点実行部を FPGA により構成したハードウェアで実行した場合とソフトウェアで実行した場合の速度を比較する。評価条件を以下に示す。

- ハードウェアの実行評価は C 言語で実装されたハードウェアのシミュレータによって行い、FPGA の動作周波数をもとにパラメータ推定に必要な時間を算出する。コスト値計算機を 4,8,12,16 並列としたそれぞれについてシミュレータを作製し、並列度を変えた場合の評価を行う。
- ソフトウェアでの実行評価は、本論文でハードウェアが行う固定小数点演算部分を C 言語で実装し、その速度を実際に計測する
- 総数 100000 組 (A:10 分割、f:1000 分割、ϕ:10 分割) のパラメータの中からコスト値が最小値となるパラメータを求める時間を計測する。
- 対象信号は音響信号 5 種類から 4096 サンプルを切り出したものを用いる。それぞれ５種類の窓長に分割してハードウェアにデータを渡しパラメータ推定を行う。その音の内容を表 5.2 に示す。

音響信号番号	音響の種類
1	ビオラ
2	ヴァイオリン
3	チェロ
4	音声（女性）
5	金属音 (フライパンを叩く)

表 5.2: 音響信号の種類

- 窓長を 128、256、512、1024、2048 サンプルとした場合の、ハードウェア実行時間、ソフトウェア実行時間、ハードウェア、ソフトウェアの実行速度比を求めた。評価値はすべて平均値で示す。

表 5.3 から 5.7 に結果を示す。

解析窓長が短いほど、速度比が下がっている。これはハードウェア内でパラメータ推定を行う場合、コスト値計算に掛かる時間の他にオーバヘッドとして

- サンプリングデータの先頭がコスト値計算器パイプラインに入り最終ステージに達するまでのサイクル
- 次パラメータでの計算開始に必要な制御を実行するレイテンシ

の時間が必要となり、解析窓長の短い場合はその割合が顕著になるためである。

窓長 128 サンプル、256 サンプルの実行結果で、12 並列コスト計算器の計算速度が 16 並列計算器の計算速度を上回っている。短い窓長であるために、パイプライン内をサンプリングデータの先頭が入ってから最終ステージの演算器に達するまでのサイクル数が、パイプラインで計算を行っているサイクル数と比べて、その割合が大きくなり、16 並列演算器と 12 並列演算器のコスト値計算に必要なクロック数に差がなくなる。そのため、16 並列計算器より動作周波数の高い 12 並列計算器のほうが有利となる。

5.2 ハードウェアとソフトウェアのパイプライン実行速度

本論文の解析アルゴリズムをハードウェアとソフトウェアのパイプライン実行を行った場合と、すべてソフトウェアで実行した場合の速度比較を行う。パラメータ推定精度に影響するパラメータ f の分割数を変化させて、パラメータ推定精度とパイプラインの遅延均衡、両面の観点から適切な分割数を推定する。使用するハードウェアはコスト計算器が 16 並列のもので評価する。評価条件を以下に示す。

- ハードウェア実行部の速度は C 言語によるシミュレータで、実行速度を測る。

- 解析時間に msec オーダーで時間が掛かる。CPU と本ハードウェア間の入出力時間は、解析時間より十分小さいため、考慮しない。

- 減速ニュートン法の減速係数 μ の縮小制限は 20 回とする。また修正量の反復計算にも制限回数を設ける。制限回数を超えても収束条件に達しない場合、計算を打ち切る。この制限回数は、振幅 A のみの推定を行うニュートン法では 8 回、f, ϕ のみの推定を行うニュートン法、A, f, ϕ 3 つの推定を行うニュートン法で 50 回の制限を設ける。

- 解析対象信号としてヴァイオリンの実音響信号を用いた。信号の長さは 4096 サンプルである。解析対象信号の解析窓長を 128、256、512、1024 サンプルと変化させて解析を行う。1 つの窓から抽出する正弦波パラメータ組は 5 組とする。

- 推定において、f の分割数が重要となる。そのため、ハードウェア実行のパラメータ推定におけるパラメータ分割数は、A、ϕ を 10 に固定、f を 100、200、400、800、1600、3200 と変化させたときの評価値を求める。それぞれのパラメータ分割数での推定精度を評価するため SNR(S/N 比) を用いた。SNR は N をサンプル数、x を原信号、$\hat{x}$ を再合成信号として次式で表す。

$$SNR = 10 \log_{10} \frac{\sum_{i=0}^{N-1} x_i^2}{\sum_{i=0}^{N-1} \{x_i - \hat{x}_i\}^2} \tag{5.1}$$

- パイプライン実行の速度は、シミュレータによって算出したハードウェア実行時間と、ソフトウェア実行部の実測により絶対時間を算出した。またソフトウェアのみの実行速度は、実測とする。

表 5.8~5.12 に結果を示す。前述したように、本ハードウェアは、窓長が長くなるほどコスト値計算器の、計算効率が向上する。そのためハードウェアとソフトウェアを用いた解析でも、窓長が長いほど解析速度が速くなる。

窓長 256 サンプル以上の場合、f の分割数が小さいときに速度比が低いのは、ニュートン法による推定時間の方がハードウェア実行時間より大きくなり、ソフトウェアの実行速度にパイプライン実行速度が依存するためである。ハードウェアとソフトウェアの遅延均衡をとるためには、窓長が長いほど分割数を大きくする必要がある。

解析精度の観点で見ると、f の分割数は窓長が長いほど大きくしなければ十分な精度が得られない。解析精度がほぼ頭打ちになる f の分割数が、速度と精度の観点から、最も効率良い推定を可能とする。

5.3 精度評価

本提案手法と ABS 法、平田による一般化調和解析の解析精度を比較した。解析対象信号はサンプリング周波数 44100Hz、232msec(10240 サンプル) の実音響信号 (ヴァイオリン) で

ある。ABS 法の f の分割数は 1000 に設定した。それぞれの対象信号から窓長 86msec(512 サンプル) でパラメータ抽出 15 組で解析し、解析結果から式 5.1 を用いて S/N 比を算出した。また、解析実行時間を計測した。本提案手法の解析時間は全てソフトウェアで行った実行とハードウェアとソフトウェアのパイプライン実行を行った場合で計測している。パイプライン実行のハードウェア部分の速度計測はシミュレータを用いた。

速度・精度比較の結果を表 5.13 に、それぞれの解析法で信号解析した結果の再合成信号、残差信号の様子を図 5.1~5.3 に示す。本提案手法と ABS 法、平田らによる一般化調和解析を比較すると、本提案手法が最も高い精度で信号を抽出している。また、ハードウェアを用いた場合高速で高精度な解析が行える。

実行条件	処理時間 [msec]	ソフトウェア実行速度比
ソフトウェア	293.79	1.00
1way ハードウェア	109.95	2.67
4way ハードウェア	64.21	4.57
8way ハードウェア	66.23	4.43
12way ハードウェア	58.54	5.01
16way ハードウェア	64.79	4.53

表 5.3: ハードウェア処理の速度評価 解析窓 128 サンプル

実行条件	処理時間 [msec]	ソフトウェア実行速度比
ソフトウェア	608.67	1.00
1way ハードウェア	207.83	2.92
4way ハードウェア	97.83	6.22
8way ハードウェア	88.44	6.88
12way ハードウェア	73.02	8.33
16way ハードウェア	77.33	7.87

表 5.4: ハードウェア処理の速度評価 解析窓 256 サンプル

実行条件	処理時間 [msec]	ソフトウェア実行速度比
ソフトウェア	1251.08	1.00
1way ハードウェア	401.35	3.12
4way ハードウェア	164.94	7.58
8way ハードウェア	132.17	9.47
12way ハードウェア	101.51	12.32
16way ハードウェア	101.11	12.37

表 5.5: ハードウェア処理の速度評価 解析窓 512 サンプル

実行条件	処理時間 [msec]	ソフトウェア実行速度比
ソフトウェア	3109.13	1.00
1way ハードウェア	789.95	3.93
4way ハードウェア	301.81	10.30
8way ハードウェア	222.55	13.97
12way ハードウェア	160.27	19.40
16way ハードウェア	153.43	20.27

表 5.6: ハードウェア処理の速度評価 解析窓 1024 サンプル

実行条件	処理時間 [msec]	ソフトウェア実行速度比
ソフトウェア	6293.95	1.00
1way ハードウェア	1607.25	3.92
4way ハードウェア	582.08	10.81
8way ハードウェア	405.88	15.50
12way ハードウェア	279.40	22.53
16way ハードウェア	257.57	24.43

表 5.7: ハードウェア処理の速度評価 解析窓 2048 サンプル

f の分割数	HW+SW パイプライン実行時間 [msec]			SW のみの		SNR
	総実行時間	HW 実行時間	SW 総実行時間	実行時間 [msec]	速度比	[dB]
100	1,072	1,024	993	4,834	4.50	17.035
200	2,051	2,015	983	8,170	3.98	17.081
400	4,032	3,998	975	15,664	3.88	17.078
800	7,986	7,955	982	30,052	3.76	17.078
1600	15,900	15,866	980	58,269	3.66	17.079
3200	31,690	31,662	953	112,404	3.54	17.079

表 5.8: HW · SW パイプライン実行の速度評価 (窓長 128 サンプル)

f の分割数	HW+SW パイプライン実行時間 [msec]			SW のみの		SNR
	総実行時間	HW 実行時間	SW 総実行時間	実行時間 [msec]	速度比	[dB]
100	1,012	625	973	5,259	5.19	17.040
200	1,283	1,194	999	8,762	6.82	17.859
400	2,415	2,339	996	15,696	6.49	17.871
800	4,678	4,604	1,001	30,177	6.45	17.876
1600	9,189	9,123	984	56,824	6.18	17.879
3200	18,190	18,130	973	109,692	6.03	17.878

表 5.9: HW · SW パイプライン実行の速度評価 (窓長 256 サンプル)

f の分割数	HW+SW パイプライン実行時間 [msec]			SW のみの		SNR
	総実行時間	HW 実行時間	SW 総実行時間	実行時間 [msec]	速度比	[dB]
100	1016	451	958	6338	6.24	3.164
200	1052	846	910	10838	10.30	15.221
400	1659	1515	1002	16281	9.81	18.950
800	3092	2949	989	32401	10.47	18.951
1600	5900	5753	979	58283	9.87	18.951
3200	11420	11277	928	109656	9.60	18.950

表 5.10: HW · SW パイプライン実行の速度評価 (窓長 512 サンプル)

f の分割数	HW+SW パイプライン実行時間 [msec]			SW のみの		SNR
	総実行時間	HW 実行時間	SW 総実行時間	実行時間 [msec]	速度比	[dB]
100	1,052	348	964	5,917	5.62	0.4027
200	966	669	780	12,546	12.99	1.7160
400	1,291	1,106	789	19,068	14.77	18.826
800	2,373	2,193	783	35,877	15.11	18.826
1600	4,443	4,263	777	64,891	14.60	18.826
3200	8,299	8,109	772	114,555	12.83	18.826

表 5.11: HW · SW パイプライン実行の速度評価 (窓長 1024 サンプル)

f の分割数	HW+SW パイプライン実行時間 [msec]			SW のみの		SNR
	総実行時間	HW 実行時間	SW 総実行時間	実行時間 [msec]	速度比	[dB]
100	1113	297	963	6495	5.83	0.255
200	1076	574	787	12174	11.31	1.449
400	1401	935	856	24168	17.25	17.56
800	2300	1867	846	47303	20.57	17.56
1600	4089	3695	805	89524	21.89	17.56
3200	7396	6949	857	157272	21.26	17.56

表 5.12: HW · SW パイプライン実行の速度評価 (窓長 2048 サンプル)

一般化調和解析手法	SNR[dB]	実行時間 [msec]
提案手法 (すべて SW 実行)	25.577	379,967
提案手法 (パイプライン実行)	25.577	30,658
ABS 法	23.668	2,253,736
平田らによる手法	18.499	140,001

表 5.13: 従来手法と提案手法の精度・実行時間比較

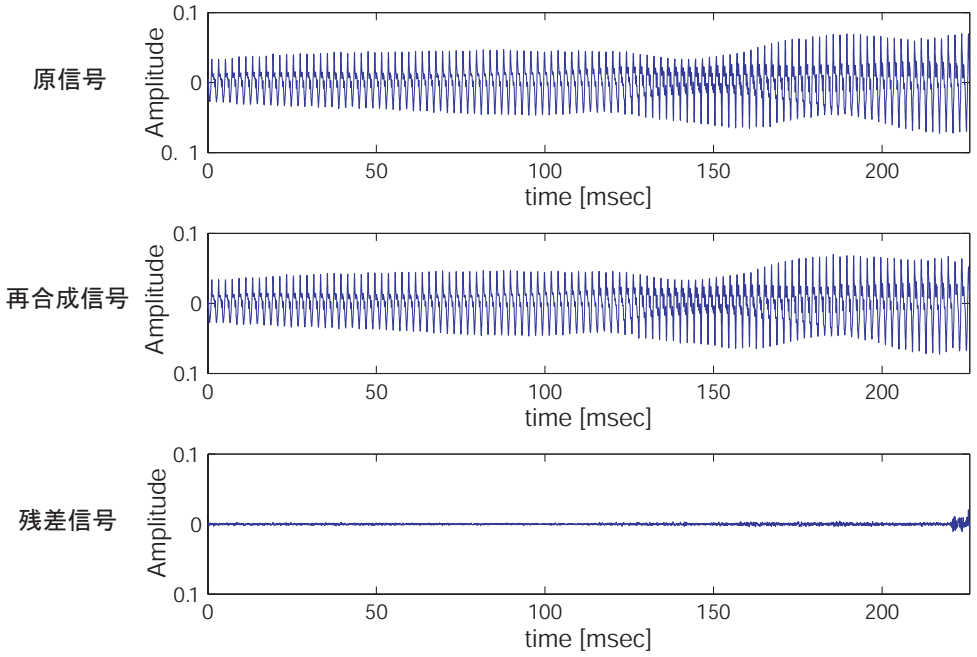


図 5.1: 本提案手法による信号解析結果

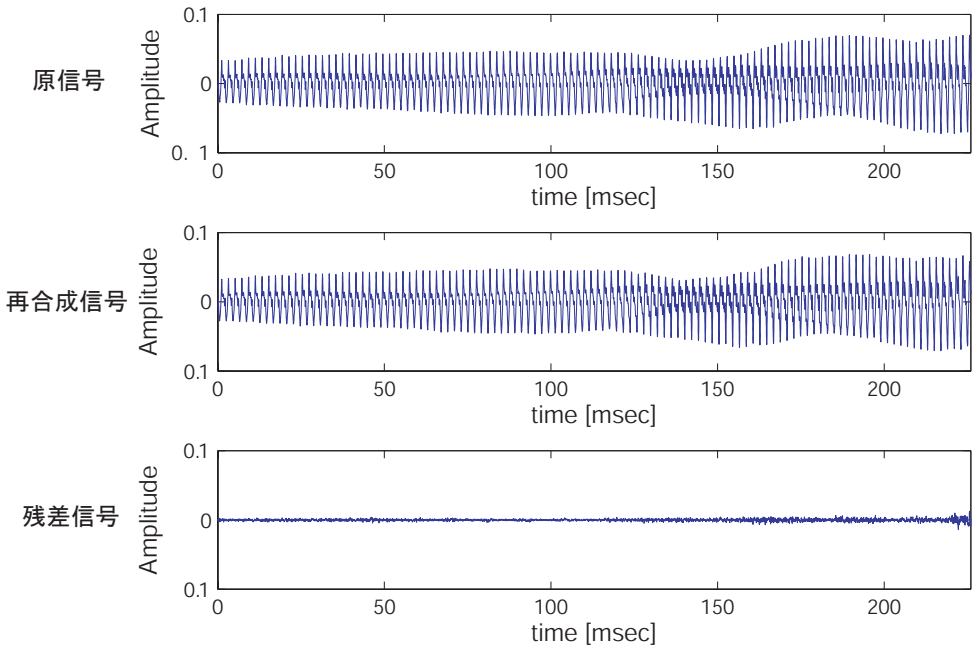


図 5.2: ABS 法による信号解析結果

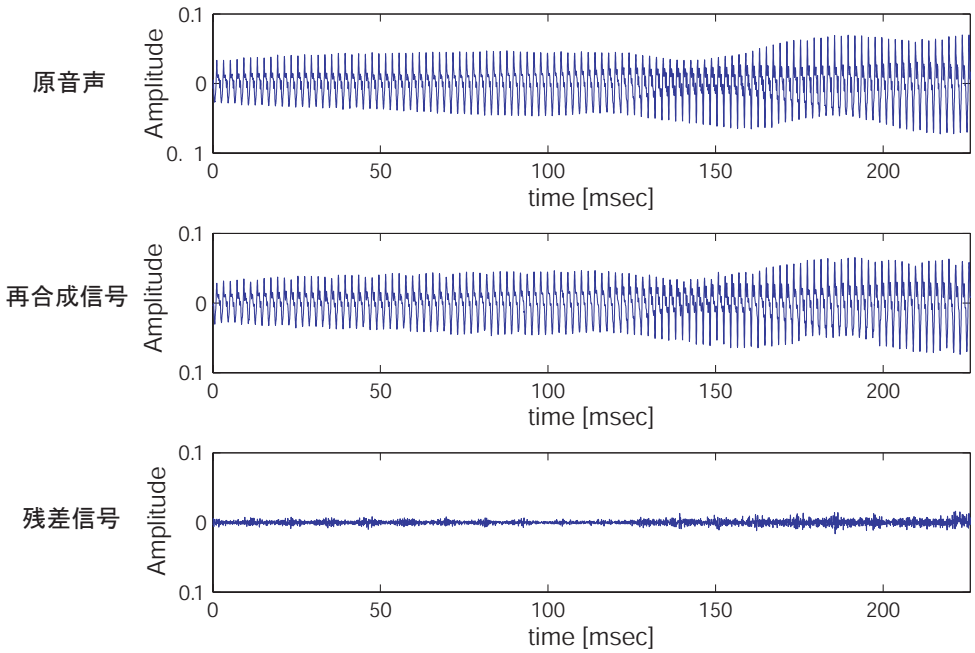


図 5.3: 平田らの手法による信号解析結果

第6章　結論

6.1　まとめ

本論文では、一般化調和解析の特徴と従来手法の問題点を述べ、一般化調和解析のコスト演算をFPGAを用いて高速化する手法を提案した。また、ニュートン法を使用し3つのパラメータ同時推定によってパラメータ推定精度を向上させる手法について述べた。ハードウェアをパイプライン化、並列化し、スループットを向上させることを目的として16並列、動作周波数48MHzの演算器をFPGA上に構築した。PCを用いたソフトウェア実行に比べ、解析窓長512サンプルの場合、約10倍の速度向上がを達成した。また従来手法と解析速度、推定精度の比較を行い、その優位性を示した。

6.2　今後の課題

本論文で構築したハードウェアの特徴として、コスト値演算器の並列化により、長い窓長ほど計算効率が向上することが挙げられる。一般化調和解析の特長である短い窓長の解析で、コスト値計算器の並列化による計算速度向上の効果が削減される。これを解消する方法として、

- パイプラインに空きが生じないコスト値計算器の構築
- 複数パラメータのコスト値演算を同時に行うパラメータレベルでの並列化

が挙げられる。

謝辞

本研究を行うにあたりご指導、御鞭撻を頂いた北陸先端科学技術大学院大学情報科学研究科 田中清史　助教授に深く感謝するとともに、ここに御礼申し上げます。

適切な御意見、御助言を頂きました本学の日比野　靖教授、井口　寧助教授に深く感謝致します。

疑問質問に対して親切なアドバイスをして頂いた田中研究室の請園智玲氏、今井俊晴氏、高橋礼彦氏、上田憲吾氏、島田信行氏、森田充氏田中研究室 OB の吉兼寛氏に厚くお礼申し上げます。

参考文献

[1] N.Wiener "The Fourier Integral and Certain of Its Applications" 1958 Dover Publication Inc.

[2] E.B. George and M.J.Smith "Analysis-by-synthesis/overlap-add sinusoidal modeling applied to the analysis and synthesis of musical tones" J.Audio ENG.Soc.40 pp497-515 1992

[3] E.B. George and M.J.Smith "Speech analysis/synthesis and modification using an analysis-by-synthesis/overlap-add sinusoidal model" IEEE Trans Speech Audio Process 5 pp389-406 1997

[4] 牛山 聡、東山三樹夫、飯塚昌弘、平田能睦 "一般調和解析による波形分析 "信学技報 EA pp38-103 1994

[5] T.Terada, H. Nakajima, M.Tohyama and Y.Hirata "Non-stationary waveform analysis and synthesis using generalized harmmonic analysis" IEEE-SP,Int.Symp.TF/TS Analysis pp429-432 1994

[6] VirtexⅡ Platform FPGA User Guide UG002(v1.9) 5 August 2004

[7] http://xilinx.com