# ログステップマルチキャリア波による<br>超音波ドップラーイメージングの理論と実装

前田 泰成

博士（情報学）

総合研究大学院大学
複合科学研究科
情報学専攻

平成 25 年度 (2013)

2014 年 3 月

本論文は総合研究大学院大学複合科学研究科情報学専攻に
博士（情報学）授与の要件として提出した博士論文である.

博士論文審査委員会

| | |
|---|---|
| 橋爪 宏達 (主査) | 総合研究大学院大学/国立情報学研究所 |
| 後藤田 洋伸 | 総合研究大学院大学/国立情報学研究所 |
| 小野 順貴 | 総合研究大学院大学/国立情報学研究所 |
| 杉本 雅則 | 北海道大学 |
| 高須 淳宏 | 総合研究大学院大学/国立情報学研究所 |

# Theory and Implementation of Ultrasonic Doppler Imaging Using Log-step Multicarrier Signals

Yasushige Maeda

Doctor of
Philosophy

Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)

March 31st, 2014

A dissertation submitted to
the Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Advisory Committee

Hiromichi Hashizume (Chair) National Institute of Informatics/
The Graduate University for Advanced Studies
Hironobu Gotoda National Institute of Informatics/
The Graduate University for Advanced Studies
Nobutaka Ono National Institute of Informatics/
The Graduate University for Advanced Studies
Masanori Sugimoto Hokkaido University
Atsuhiro Takasu National Institute of Informatics/
The Graduate University for Advanced Studies

# 概要

本論文は、新たに考案した広帯域信号であるログステップマルチキャリア信号を提案し、その理論的記述を行う。また同波形を用いた 3D ドップラーイメージングと反射体速度推定のための新アルゴリズムを提案し、同アルゴリズムを用いたシミュレーションおよび空中超音波環境における実験結果を示すものである。

本研究の主要成果は、送信波にログステップマルチキャリア波を使い、ドップラーシフトを受けた受信波との間に相互相関信号処理を行うことによって達成される。

空中超音波を用いたイメージングは、暗闇や暗渠など光を発することが困難な状況下での測定、霧やスモッグなどの光非透過環境での測定において、光学的手法を用いたイメージングに対する優位性を持つ。また、光速に対して動体反射体との相対速度比の変化が得られやすい音速の特性を利用し、物体速度を測定できることにも、空中超音波を用いる利点がある。超音波イメージングでは、光学的手法におけるレンズとシャッターの信号処理による代替手段の開発が基本的課題である。この機能を持つ信号処理技法はビームフォーマーとマッチドフィルターとして知られている。

近年 MEMs や cMUTs といった半導体技術を応用して製造された広帯域の超音波を送受信できる素子が注目を集めている。本研究ではイメージングのために、これらの素子の利用を想定し、広帯域信号の信号処理を行う方法論を扱う。広帯域信号として従来からよく用いられてきたものにチャープ信号が挙げられるが、最近では通信分野で多用される直交周波数多重化方式（OFDM）で用いられる信号を用いる方法が考案された。OFDM 信号はキャリア周波数が等間隔で並んだ信号であるため、等ステップマルチキャリア信号と呼ぶ。

ドップラー効果は反射波の周波数に対して乗法的に作用することに注目する。等ステップマルチキャリア波は、ドップラー効果により等間隔で並んだキャリアは高周波数ほど間隔の広いスペクトル分布を持つようになる。このことにより各キャリアの直交性が失われるばかりでなく、マッチドフィルターの演算における同一波形の検出性能が損なわれる。このため広範囲のドップラー量の測定には、想定されるドップラー

シフト範囲に対応した多大な数の相関演算を必要とする。

この欠点を改善するのが、ログステップマルチキャリア信号である。ログステップマルチキャリア信号は、キャリアの対数間隔が等しくなるように配置されているため隣接キャリアの比はドップラー不変性を保有する。また同時にログステップマルチキャリア信号のマッチドフィルター感応性のドップラーシフトによる変化によりドップラー量の微細変化の検知能力を高める。

本稿では、ログステップマルチキャリア波の各キャリア初期位相に整列位相の設定を施した場合において、この波が他の広帯域信号に見られない良い性質を持つことを示す。ログステップマルチキャリア波のこの性質を用いたドップラーイメージング及びドップラー速度推定のためのアルゴリズムの提案を行う。

この速度推定アルゴリズムは、従来医用超音波で循環器系の血流検査のために用いられてきたカラードップラー法に対して、検出可能な推定域が広く、推定精度においても約 20 倍の性能を持つことが、シミュレーションにより示される。

本稿は次の構成をとる。

最初に第 1 章で、本研究の大きなテーマである広帯域の連続波を用いた音響イメージングについて紹介し、位置と速度の同時推定を行うドップラーイメージングの研究背景について解説する。ここでは、自然界における超音波測位を参考にしながら研究の概要について記述する。本研究で参考にした先行研究を、空中超音波のみならず、医用超音波、非破壊検査、合成開口レーダー等の関連分野を参照しながら紹介する。

第 2 章では、ドップラーイメージングに用いられる各種手法について、その基礎となる技術を中心に定式化を行う。特に超音波を用いた 3D イメージングと到来角、到来時間推定を可能にする基礎技術についての検討を行う。本研究で用いるビームフォーマー、マッチドフィルター、3D イメージングおよびドップラー速度推定についての基礎概念を説明する。この章ではまた、医用超音波で用いられる各種イメージング手法の紹介を行う。特に本稿で比較検討の対象としたカラードップラー法について詳しく解説する。加えて同手法においてドップラー推定を行うために考案した包絡線位相からドップラー速度を再構成する方法について述べる。

第 3 章では、提案手法で用いるログステップマルチキャリア信号について定義を行い、その各種理論的性質について証明を行う。また同信号について他の広帯域信号と比較しながら解明を行う。本研究の主要概念である、(1) ドップラー相互相関関数の持つ到来時刻の系統偏移および (2) 相互相関プロファイルの微細構造について、その理論的性質について解明する。

第 4 章では、これらを利用した 3D イメージングについて定式化し計算アルゴリズムを構築する。また実際にイメージング実験を行った結果を紹介する。

第５章では、既存手法及び提案手法を用いたドップラー推定精度についての比較を、シミュレーションにより実施する。

第６章では、提案手法の耐ノイズ性を改善する方法として、ログステップマルチキャリア波特有の性質を用いた速度推定手法の改善方法についての提案を行う。また同手法についてシミュレーションにより有用性を検討する。

第７章では、提案手法による実験を行うために製作した超音波イメージング装置についての紹介を行う。特に多チャンネルからの高速同時入力を可能にする回路技術について、本研究で行った発案と工夫に重点を置きながら、当該分野で必要とされる超音波アレイ受信装置について解説する。

最後に第８章において、本論文の総括を行うと同時に、研究の今後の展開について述べる。

# Abstract

This paper describes a series of newly developed wide-band signals called log-step multicarrier signals, and shows theoretical properties of the signals. In application of those signals, we develop algorithms for 3-dimensional Doppler imaging, and the target velocity estimation. Results of computer simulations using the algorithm and results of physical ultrasonic experiments in air environment are to be shown.

The principal achievement of this study is established by using the log-step multicarrier signal as a transmission wave, and cross-correlating to the received one affected by a Doppler shift.

In comparison with imaging technologies using optical method, those using airborne ultrasound have an advantage of applications in difficult situations such in darkness or underdrain where emitting light is prohibited, and in fog or smog where lightning is disturbed. Also by using airborne ultrasound, we have an advantage of measuring the velocity of a reflector object since the relative velocity ratio is easier to obtain by using ultrasound than light or radio-wave. Fundamental problems in ultrasonic imaging are to develop the alternative methods that simulate the lens and an optical condensing device. Those functionalities of signal-processing are known as the names of beamformer and matched-filter.

Recently newly developed devices called micro electro mechanical systems (MEMs) and capacitive micro-machined ultrasonic transducers (cMUTs) by fabrication of semiconductor technology have considerably attention since they are capable of receive and transmit wide-band ultrasonic sounds. We suppose facilitating those devices and construct a methodology for signal-processing of wide-band signals. Although in tradition, chirp signals are preferably employed in such applications using wide-band signals, an imaging method which utilizes the signals of orthogonal frequency-division multiplexing (OFDM) in the field of

digital communication has been developed. An OFDM signal consists of multiple carriers positioned in frequency domain with an equal step, hence we call an in-step multicarrier signal.

We consider that the Doppler effect acts on the reflected sound in a multiplicative manner. The reflected sound of in-step multicarrier signal affected by a Doppler shift turns to have carrier frequencies with in-equal steps. This effect distorts not only the orthogonal properties of the carriers, but also the detection performance of the identical signal by the matched-filter operation. To measure the Doppler velocity with a wide estimation rage, it is required to execute a large number of cross-correlations calculation for supposed numbers of the Doppler effect.

The log-step multicarrier signals overcome this disadvantage of the in-step multicarrier signals Since in the proposed signal, the frequencies of carriers are designed with logarithmically equal, the frequency ratio between each adjacent carriers attains the property of Doppler invariance. The signals also have a sharp sensitivity for a Doppler shift in matched-filter operation, therefore a high performance in measuring the amount of Doppler shift is attained by using the signals.

In this paper, a series of the log-step multicarrier signals is carefully designed by specifying the initial phases of the carriers by a single parameter, and call them ordered-phase log-step multicarrier signals. The characteristic feature of that series which does not appears in other wide-band signals is shown. Then we propose a new algorithm for Doppler imaging, which constructs a 3D image of stationary and moving objects, and the algorithm to estimate the velocity of the objects.

Comparison of the proposed velocity estimation algorithm and the color Doppler algorithm which has been developed in medical application to detect circulatory blood flow is executed by a computer simulation. The result shows that the proposed algorithm has a wider estimation range than the color Doppler method, and attains around 20 times estimation accuracy.

This paper has the following structure.

Firstly in Chapter 1, I introduce the main issue of this work, acoustic imaging methods using wide-band continuous waves, and explain the background of the study on the Doppler imaging, which measures the position and the velocity of the target simultaneously. In this chapter, I take examples of animals' imaging in nature. Former studies in this field of airborne ultrasound with referring related

fields such as ultrasonography, non-destructive inspection, and synthetic aperture radar.

In Chapter 2, I formalize fundamental methods and techniques used in the Doppler imaging. Consideration of the fundamental methods is done in the field of signal-processing of wide-band signals with arrayed sensor, and explains the fundamental concepts in this paper including beamformer, matched-filter, 3D imaging and Doppler velocity estimation. In this chapter, several imaging methods used in medical applications are also introduced. I focus on the color Doppler method, which is frequently used in practical medical ultrasound applications, to be employed for airborne ultrasonic imaging. I show an ad-hoc algorithm on the color Doppler method to estimate the Doppler velocity from the obtained phase of the envelope function.

In Chapter 3, the definition of log-step multicarrier signals is given, and some theoretical properties of the signals are proved. We also give comparisons with other wide-band signals and clarify the difference. In this chapter, the main concepts of this study (1) the systematic temporal shift on the Doppler cross-correlation function, and (2) the fine structure of the cross-correlation profile, are explained from the theoretical point of view. We construct the algorithms for 3D imaging and Doppler velocity estimation.

In Chapter 4, the performance comparison of the color Doppler method and the proposed method is done by a computer simulation. I also show the results of imaging in physical environment using the implemented experimental apparatus.

In Chapter 5, I propose an algorithm to improve the performance of the Doppler velocity estimation using a proper feature of the log-step multicarrier signals in noisy conditions .

In Chapter 6, we confirm by a computer simulation that the improved algorithm shows an expected performance.

In Chapter 7, I show the experimental apparatus for ultrasonic imaging that I have designed and implemented. Circuit design criteria, that enable fast capturing from multiple channels, are explained with focusing especially in the original idea and techniques in this study.

Finally in Chapter 8, I conclude this study, and give the future perspective.

# 目次

# 図目次

# 表目次

# 第 1 章

# 序論

## 1.1 本研究の背景と目的

空中超音波を用いたイメージングは、暗闇や暗渠など光を発することが困難な状況下での測定、霧やスモッグなどの光非透過環境での測定において、光学的手法を用いたイメージングに対する優位性を持つ。また、光速に対して動体反射体との相対速度比の変化が得られやすい音速の特性を利用し、物体速度を測定できることにも、空中超音波を用いる利点がある。超音波イメージングでは、光学的手法におけるレンズとシャッターの信号処理による代替手段の開発が基本的課題である。この機能を持つ信号処理技法はビームフォーマーとマッチドフィルター [Asa11][Aki10] として知られている。

音波を用いたイメージングが真価を発揮するのはむしろ水中や生体への応用においてである。空中に比べ水中における音波の伝搬速度は大きく減衰も小さいため、高性能なエコーロケーションが可能である。医用超音波においては、送信波にパルス波を用いて生体組織の画像化と血流などの測定を行うカラードップラー法 [KNKO85] が早くから開発され実用に供されてきた。

ドップラー効果を利用して動体速度の検出を動体位置と同時に検出して画像化する技術は、ドップラーイメージングと呼ばれ研究が行われている。カラードップラー法もその一種である。ドップラーイメージングでは、センサーアレイに対する到来角及び距離方向の反射体の 3 次元位置情報に加え、距離方向の動体速度の検出が可能であり、単体のセンサーアレイを用いた場合に計 4 次元分の情報を検出することが可能である。

近年 MEMS (Micro Electro Mechanical system) や CMUTs (Capacitive Micromachined Ultrasound Transducers) [LNB$^+$11] といった半導体技術を応用して製造

された広帯域の超音波を送受信できる素子が注目を集めている。本研究ではイメージングのために、これらの素子の利用を想定し、広帯域信号の信号処理を行う方法論を扱う。広帯域信号として従来からよく用いられてきたものにチャープ信号が挙げられるが、最近では通信分野で多用される直交周波数多重化方式 (OFDM)[Ita05] で用いられる信号を用いる方法が考案された [ISH11]。OFDM 信号はキャリア周波数が等間隔で並んだ信号であるため、等ステップマルチキャリア信号と呼ばれる。

ドップラー効果は反射波の周波数に対して乗法的に作用することに着目すると、等ステップマルチキャリア波は、ドップラー効果により等間隔で並んだキャリアは高周波数ほど間隔の広いスペクトル分布を持つようになる。このことにより各キャリアの直交性が失われるばかりでなく、マッチドフィルターの計算における同一波形の検出性能が損なわれる。このため広範囲のドップラー量の測定には、想定されるドップラーシフト範囲に対応した多大な数の相関演算を必要とする。

これらの問題を解決するため、本論文では新たに考案した広帯域連続波を提案する。この連続波は、従来の等間隔で並んだキャリアに代えて指数間隔で並んだ複数のキャリアから構成され、ドップラーシフト作用に対する相似変換性を持つものである。本論文ではこの連続波をログステップマルチキャリア波と呼び、その性質を解明する。特に各キャリアの位相が本論文で整列位相と呼ぶ特別な関係にあるとき、ログステップマルチキャリア波は高いドップラー感応性とドップラー不変性を併せ持つようになる。

本論文では、超音波イメージングの課題の中でも、特にドップラー速度推定法の提案と推定精度の改善をテーマに議論する。この目的のために、整列位相ログステップマルチキャリア波の性質を説明するいくつかの命題を挙げてそれらを証明する。その性質には相互相関における到来時刻系統偏移と相互相関プロファイル微細構造が含まれる。またこれらの性質を用いたドップラーイメージング及び速度推定アルゴリズムの提案を行う。ドップラーイメージングアルゴリズムの有効性は本論文中で実機を用いた実験により示される。速度推定アルゴリズムの性能は、シミュレーションにより特にカラードップラー法との比較を行うことによって示される。

整列位相ログステップマルチキャリア波の性質はまた、過酷な低い SNR 条件下においても高いドップラー速度推定精度を保つアルゴリズムの設計に役立てる事ができる。到来時刻系統偏移を利用して整列位相ログステップマルチキャリア波 2 波を連結した送信波を用いるノイズ耐性改善アルゴリズムを提案し、シミュレーションにより有効性を検証する。

歴史的経緯から言うと、パルス波に代えてインパルス再構成のために相互相関計算を用いる方法は、高い解像度が得られエネルギー効率も良いことからその有効性が主

張され [Col91][Wil93a][Wil93b] 近年広く認められる手法になった。

広帯域連続波に相互相関計算を使用してドップラー解析を行う手法は、広帯域曖昧度関数 [Alt73] や連続ウェーブレット解析 [You97] の名前で知られ研究されてきている。

カラードップラー法もドップラー速度を検出するために相関計算を用いるが、この場合は受信波の遅延を伴う自己相関計算であり、干渉法の一種だと捉える事ができる。その意味で連続波からインパルスを再構成するために送信波と受信波の相互相関計算を用いる方法とは区別される。パルス波に対する干渉法の適用にはパルスの包絡線に対するものとパルスのキャリアに対するものが考えられ、前者を用いた伝統的なカラードップラー法に対し、後者を用いたカラードップラー光学干渉法 [RYBI02][YRI03] が考案されその推定精度における優位性が主張されている。

イメージングの性能や推定精度を改善するために広帯域連続波を用いて相関計算によりインパルスを再構成する方法としては、伝統的にチャープ波を用いる方法が研究の対象になっていた [ITMM02][HKK08]。

また本研究に類似の研究として、音楽情報処理における波形の乗法相似性を用いたスペクマート法の研究 [TNS03][KSNS04][SKOS06a] を挙げることができる。スペクマート法は特に豊かな倍音構成を持つ楽音の分析に有効性が認められている。

## 1.2 本論文の構成

本論文の以降の部分を次の構成に従って記述する。

第 2 章では、ドップラーイメージングに用いられる各種手法について、その基礎となる技術を中心に定式化を行う。特に超音波を用いた 3D イメージングと到来角、到来時間推定を可能にする基礎技術についての検討を行う。本研究で用いるビームフォーマー、マッチドフィルター、3D イメージングおよびドップラー速度推定についての基礎概念を説明する。この章ではまた、医用超音波で用いられる各種イメージング手法の紹介を行う。特に本稿で比較検討の対象としたカラードップラー法について詳しく解説する。加えて同手法においてドップラー推定を行うために考案した包絡線位相からドップラー速度を再構成する方法について述べる。

第 3 章では、提案手法で用いるログステップマルチキャリア信号について定義を行い、その各種理論的性質について証明を行うと同時に、他の広帯域信号と比較しながら同信号の性質を解明する。ここでは、本研究の主要概念である、ドップラー相互相関関数の持つ到来時刻の系統偏移および、相互相関プロファイルの微細構造について、その理論的性質について解明する。

第 4 章では、ログステップマルチキャリア波の性質を利用したドップラーイメージングについて定式化し、計算アルゴリズムを構築する。また実際にイメージング実験を行った結果を紹介する。第 5 章では、ログステップマルチキャリア波を用いたドップラー速度推定アルゴリズムを提案する。また、既存手法であるカラードップラー法および提案手法を用いた推定精度の比較を、シミュレーションにより実施する。第 6 章では、ログステップマルチキャリア波の性質である相互相関における到来時刻系統偏移を利用したドップラー速度推定法の耐ノイズ性改善の提案を行う。ここでは偏移方向の異なる波形を 2 波連結した送信波を用いて、SNR が 0dB を下回るような悪条件下においても、ドップラー速度推定アルゴリズムが有効であることを、シミュレーションにより検証する。第 7 章では、提案手法による実験を行うために製作した超音波イメージング装置についての紹介を行う。特に多チャンネルからの高速同時入力を可能にする回路技術について、本研究で行った発案と工夫に重点を置きながら、当該分野で必要とされる超音波アレイ受信装置について解説する。

最後に第 8 章において、本論文の総括を行うと同時に、研究の今後の展開について述べる。

# 第 2 章

# ドップラーイメージング

前章では、超音波イメージングについて広く敷衍して関連分野についての紹介を行った。本章では、特に本論文の前提となる各技法について、特に各技法の計算法について特定の議論を行う。音響イメージングにおけるドップラーイメージングとは、通常のイメージング技法に加えて、ドップラー効果による影響を何らかの方法で測定することによって、動反射体の位置と速度を同時に測定し図示する方法のことをいう。

医用超音波の分野ではドップラーイメージングの方法が早くから確立し実用化されてきた。医用超音波では連続波ドップラー法、パルスドップラー法、カラードップラー法、パワードップラー法が広く用いられている。本論文ではこの中でもアルゴリズムの同定ができ、イメージング可能なドップラー範囲と推定精度のバランスのとれたカラードップラー法を比較対象とする。

本章では、まず速度の検出を伴わない通常のイメージング手法について概観し、その後で速度検出を伴うドップラーイメージングについて先行研究と共に紹介する。医用超音波で用いられる手法についてもカラードップラー法を中心に紹介する。

## 2.1 音響イメージング

音響イメージングとは一般に、光の代わりに音を用いて世界の状態を可視化する方法のことをいう。カメラのような光を用いる光学イメージング装置は、自然界から発せられる光をレンズなどの手段によって方向成分に分解し、フィルムや CCD 素子などを一定時間露光して受動的に光のエネルギーを捉える。これに対して音響イメージング装置は、能動的に特定の帯域にエネルギーを持つ音波を発信し、対象物で反射又は散乱した音波をマイクロフォンで捉える事によって音のエネルギーを捉える。

音響イメージングには、暗闇や暗渠など光を発することが困難な状況下での測定、

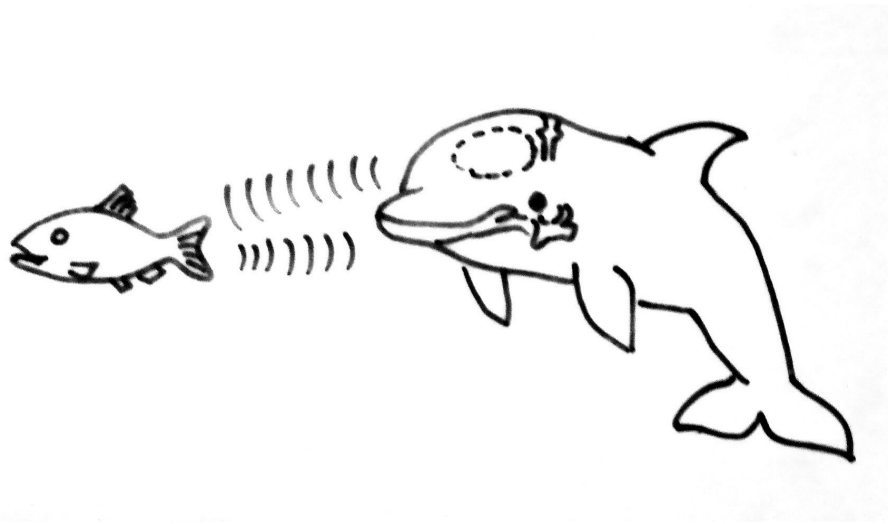

図 2.1　イルカの補食活動

霧やスモッグなどの環境での測定において、光学的手法を用いたイメージングに対する優位性がある。また、光速よりも音速が遅いことを利用し、ドップラー効果により物体速度を測定できることにも利点がある。音響イメージングで用いられる方法は、主にアクティブセンシングと呼ばれる方法である。自然界が発する波動を受動的に捉えるパッシブセンシングとは違い、アクティブセンシングでは発信器によって任意の波形を発生することで反射波や散乱波を捉える。この方法では発生する波動をデザインすることが重要な課題となる。

自然界の動物ではイルカやコウモリが超音波によるエコーロケーションの能力を持つとされている。エコーロケーションは反射体で跳ね返って来た音波を複数の場所で受信し、到来時間により反射体の位置を特定する仕組みである。イルカは水中で 2 種類以上の音波を発する。そのうちの一つはクリックルと呼ばれる高い周波数のパルス波であり、もう一つは、ホイッスルと呼ばれる低い周波数の連続波である。クリックルは位置測定のために用いられ、ホイッスルはイルカ間のコミュニケーションに用いられていると考えられている [SNM02]。

イルカはまた、超音波を発する事で獲物の聴覚を狂わせて補食活動の助けとすることも知られている。この場合に超音波は鼻腔から発せられるが、この際に鼻腔と皮膚の間にあるメロン器官と呼ばれる脂肪のかたまりが凸レンズの役割を果たして対象物に向けて超音波ビームを集中する (図 2.1)。これはいわゆる送信側ビームフォーマーであり、音響イメージングの生体による実現形態であると言える。

いっぽうコウモリは空中超音波を発して対象物で反射する音波を拾うことにより反射体の定位を行う [SFnk11]。コウモリは獲物の捕捉のために非常に短いパルス状の

図 2.2　コウモリの補食活動

チャープ波を発生することが知られている。パルス間隔は対象物に近づくほど狭くなるため、コウモリは対象物の運動も含めたエコーロケーションを行っているものと考えられている (図 2.2)。

音響イメージングはエコーロケーションの考え方をさらに押し進め、実空間に配置された複数の対象物を反射波によって平面又は空間上にマッピングし、画像化までを行おうというものである。音響イメージングは複数の対象物の位置を同時にマッピングするために、次に述べるビームフォーマーが用いられる。

## 2.2　ビームフォーマーによる方向成分の抽出

ビームフォーマーは特定方向からの成分を抽出するレンズの役割を果たす。ビームフォーマーの適用には複数のセンサーを配置して複数の異なる位置での受信信号が得られることが必要不可欠になる。音波は媒体上を、光波と比べて比較的遅い速さで伝わるため、空間上の異なる位置にあるセンサーの受信信号は同一時刻においてもそれぞれ異なる。ビームフォーマーは、同一方向からの到来波が同一時刻にすべてのセンサーにおいて一致するように、各センサーから得られた信号に遅延や位相回転を適用して足し合わせる。この計算がすべての方向について繰り返されてセンサーアレイの次元分の到来方向成分が得られる。センサーアレイは 2 次元平面上に配置された場合には、到来方向成分全体の次元は 2 次元となる。

ビームフォーマーはセンサーアレイにより受信した信号を遅延や位相回転を伴う加

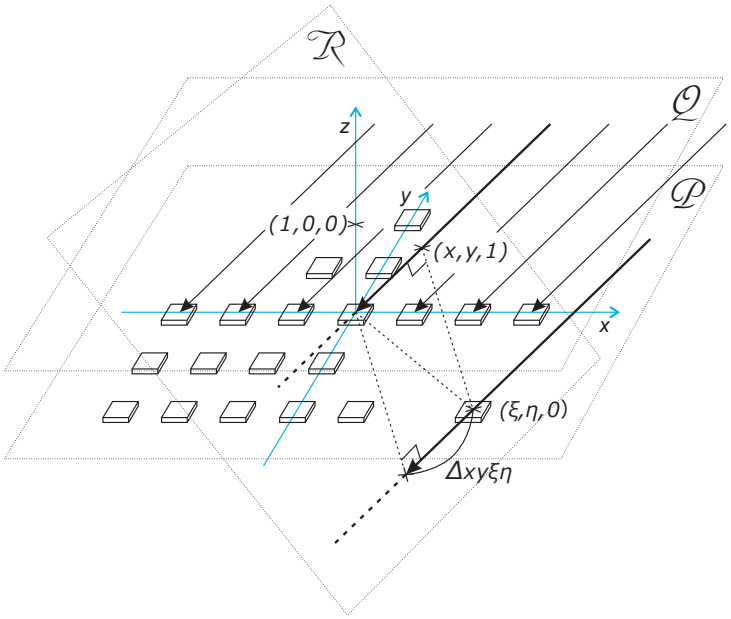


図 2.3　2D ビームフォーマーにおける遅延時間 $\Delta_{xy\xi\eta}$ の計算

重平均を行うことによって、音波の特定の到来方向成分を抽出するものである。センサー毎の信号に波面の到来時刻が同一になるような遅延を加えるものを遅延和ビームフォーマーと呼び、特定周波数の位相が揃うような位相回転を加えるものを位相回転ビームフォーマーと呼ぶ。

遅延和ビームフォーマーはどの周波数に対しても同一の到来方向成分が得られるため広帯域信号の処理に向いている。いっぽう位相回転ビームフォーマーはセンサ信号の加重和係数として位相回転を表現する複素数を選べば良いため、表記が簡潔で計算が効率的に行えることが特徴である。本稿では広帯域信号を扱う必要から、遅延和ビームフォーマーを扱う。

ビームフォーマーの適用には、センサーアレイの使用が不可欠である。センサーアレイは目的に応じて、1 次元または 2 次元上に配置される。2 次元センサーアレイは一般に、どのような曲面上に配置することも理論的に可能であるが、製作上の容易さから平面上に配置されることが多い。

平面状のセンサーアレイからの信号を処理する 2D ビームフォーマーは到来方向の表現に 2 つのパラメータを必要とする。ビームフォーマーの適用の結果、到来時間時間ごとの 2 次元画像が得られるが、到来方向を示す 2 次元座標の表現にいくつかの方法が用いられている。代表的なものに極座標を用いた方法と、経度緯度座標を用いたものがあるが、本稿では表現の簡潔さと計算の容易さの観点から、単位球面の中心射影に基づく方法 [KJM98] に基づく方法を使用する。

遅延時間の最も正確な定式化は、送信機から反射体を経て各々のセンサーに至る経路の長さを音速で除算した結果として得られる。この定式化では、単一の送信機と単一の受信機を用いた場合に同一到来時間の反射体の位置の為す曲面は回転楕円面となり、この計算をそれぞれのセンサーと到来時間について行うことは計算コストを要することになる。平面波近似は、反射体と送信機、受信機との距離が十分長い場合に良好な近似をもたらす。ビームフォーマーにおける平面波近似は、センサー毎の遅延時間の計算をセンサー面の鉛直方向ベクトルと波数ベクトルのなす角度を内積演算に帰着させることができるため、計算コストの点で有利である。

平面波近似を行った場合のビームフォーマーの計算方法を次に示す。

3 次元空間が $(x, y, z)$ で座標づけられているものとする。センサーアレイは座標原点 $(0, 0, 0)$ を中心として、$z = 0$ 平面に置かれていると仮定する。するとアレイを構成するセンサーの座標は $(\xi, \eta, 0)$ と表現される。この座標系において、反射体の位置は一般的に $(x_1, y_1, z_1)$ と表現できるが、これを $z = 1$ 平面の上にベクトルの終点がくるように正規化する。すなわち $x = x_1/z_1$、$y = y_1/z_1$ とし、$z_1 = 1$ と変換する。するとセンサー座標 $(\xi, \eta, 0)$ の到来方向ベクトル $(x, y, 1)$ への正射影は、センサーの座標原点に対する経路差となる (図 2.3)。すなわち、次式により経路差 $\Delta_{xy\xi\eta}$ を求めることができる。

$$\Delta_{xy\xi\eta} = \frac{(\xi, \eta, 0) \cdot (x, y, 1)}{|(x, y, 1)|} = \frac{\xi x + \eta y}{\sqrt{x^2 + y^2 + 1}} \tag{2.1}$$

この経路差を正負を考慮して音速 $c$ で除算して遅延時間が得られる。つまり、次式により遅延時間が得られる。

$$\tau_{xy\xi\eta} = -\frac{1}{c}\Delta_{xy\xi\eta} = -\frac{\xi\ x + \eta\ y}{c\sqrt{x^2 + y^2 + 1}} \tag{2.2}$$

遅延和ビームフォーマーはこの遅延時間を用いてセンサー毎の受信信号を足し合わせることによって方向成分 $B_{xy}(t)$ を抽出する。

$$B_{xy}(t) = \int_{-X}^{X} \int_{-Y}^{Y} s_{\xi\eta}(t - \tau_{xy\xi\eta})\, d\eta\, d\xi \tag{2.3}$$

到来角を $(x, y, 1)$ により指定するこの方法は、中心射影と呼ばれ、地球を半径 1 の球体だと見立てて、北極点の接平面に座標系を設定したものである (図 2.4)。極座標系 $(\theta, \phi)$ から接平面の座標系 $(x, y, 1)$ への変換は、

$$\begin{cases} x = \tan\phi \cos\theta \\ y = \tan\phi \sin\theta \end{cases} \tag{2.4}$$

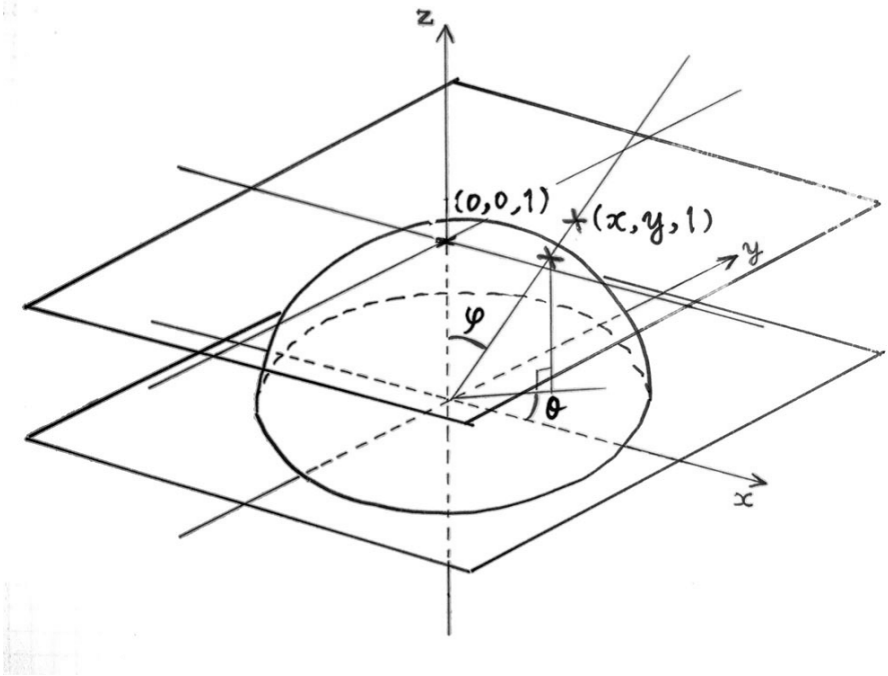


図 2.4　中心射影による接平面への座標変換

として得られる。

ここで紹介した方法を含め、同一到来時刻における反射体像を同一平面にマッピングして得られた画像は、医用超音波で C モード画像と呼ばれているものである。なお、医用超音波の F モード画像はこれとは異なり、信号処理によって任意の曲面上にある反射体像を平面上にマッピングしたものであり、本論文で扱う画像とは異なる。本論文では F モード画像は扱わない。

## 2.3　マッチドフィルターによるインパルスの再構成

マッチドフィルターは、一定時間露光を行うシャッターの役割を果たす。ここでの光学的手法との相違点は、人工的に信号を発生して反射や散乱により跳ね返ってきた音波を受信することであって、理想的な条件での受信波が予想できることである。これを積極的に用いる方法がマッチドフィルターである。

光学的手段におけるシャッターに比べて、マッチドフィルターははるかに高速に動作する。シャッターが一定の露光を決めるために一度だけ開閉するのに対して、マッチドフィルターは送信波の波形の上下にあわせてめまぐるしく開閉を繰り返して、受信波の中から送信波と同一の特徴を持った波形を選び出す働きを持っている。この働きを機械的手段により実現することはできないため、信号処理による積分演算によっ

て行われることになる。

結果としてマッチドフィルターは、センサーアレイによって捉えられた音場の情報のうち、波の到来時刻に関する情報を抽出する。受信波に送信波と同一の特徴が認められると、マッチドフィルターは到来時刻に鋭いピークを持つ波形を出力する。ビームフォーマーとマッチドフィルターをあわせて用いることにより、2 次元分の到来方向成分と 1 次元分の到来時間成分が分解でき、計 3 次元分のイメージングを行うことができる。

ここで一般に用いられるマッチドフィルターの計算に次の 2 種類のものがあることに注意しておく。

マッチドフィルタは、2 つの信号列の類似度を測る方法である。一般的にマッチドフィルタは 2 つの関数の相互相関を用いて定義されている。すなわち、送信信号 $g(t)$ と受信信号 $h(t)$ との時刻 $t$ における相互相関は、次式で定義される。

$$\langle g|h\rangle(t) = \int_{-\infty}^{\infty} \overline{g}(\tau - t)\, h(\tau) d\tau \tag{2.5}$$

$f(t)$ と $g(t)$ が実数値関数のときには、エルミート積は通常の積として計算できる。この場合にも相互相関関数が得られるが、$f$ と $g$ の周波数に対応した鋭いピークが得られる反面、包絡線の検出が難しくなる。

相互相関定理によれば、2 つの関数 $g(t)$ と $h(t)$ の相互相関は、周波数スペクトル積の逆フーリエ変換に等しい。

$$\langle g|h\rangle(t) = \mathfrak{F}^{-1}(\overline{\mathfrak{F}g} \cdot \mathfrak{F}h)(t) \tag{2.6}$$

2 つの実数値関数 $g(t)$ と $h(t)$ の 相互相関も同様に計算できる。

パルス圧縮 (pulse compression) フィルタは、マッチドフィルタの変種であり、送信波 $g(t)$ と受信波 $h(t)$ のフーリエ変換から得られる複素スペクトルの商を用いて定義される [SH06]。

$$\langle h/g\rangle(t) = \mathfrak{F}^{-1}(\mathfrak{F}h/\mathfrak{F}g)(t) \tag{2.7}$$

一般に複素数 $z \neq 0$ に関して、逆数 $z^{-1}$ と、共役複素数 $\overline{z}$ の間に、$z^{-1} = \overline{z}/|z|^2$ なる関係が成立している。送信波 $g$ のフーリエ変換に関して、$|\mathfrak{F}g(\xi)|^2 = 1$ なる関係が成り立つとき、(5) 式と (6) 式は等価となる。なぜなら先の関係式より $(\mathfrak{F}g)^{-1} = \overline{\mathfrak{F}g}$ であり、ただちに、$\mathfrak{F}h/\mathfrak{F}g = \mathfrak{F}h \cdot \overline{\mathfrak{F}g}$ が成り立つからである。

つまりパルス圧縮フィルタ (2.7) は、あらゆる周波数成分の大きさが 1 に等しい場合に、スペクトル積整合フィルタ (2.6) の代用物として機能する。また、多少の周波数成分の大きさの比がある場合においても、それぞれの大きさを 1 としたときの整合フィルタ (2.6) の挙動を模倣していると言える。

しかしながら、関数 $g$ として用いる実際の送信波形は、特定の周波数帯域にのみ信号成分が含まれ、それ以外の帯域は周波数成分が極めて 0 に近くなるのが普通である。そのため、応用においてはバンドパスフィルターを併用して信号帯域にのみこの演算を適用する等の工夫が必要になる。

マッチドフィルターの計算に複素除算を用いる方法は、チャープ波などの連続スペクトルを持つ信号を送信波とする場合に向いている。本論文では、離散スペクトルのマルチキャリア波を扱う必要性から、もっぱらエルミート積によるマッチドフィルターを用いる事になる。

ここでビームフォーマーとマッチドフィルターの適用順について注意しておく。マッチドフィルターの前過程または後過程としてビームフォーマーを適用することによって、センサーアレイの受信した信号を方向成分に分解することができる。

2D ビームフォーマー $B$ 及び、マッチドフィルター $M$ はそれぞれ、コンパクト領域上の自乗可積分関数に対する演算として定義されているため、積分順序は交換可能である。もっと強く、デジタル信号処理におけるビームフォーマーおよびマッチドフィルターの計算は有限和であり、いつでも両者は交換可能である。

すなわち次の可換図式が成立する。

$$\begin{array}{ccccc} & & \overset{B}{} & & \\ & \Xi \times H \times T & \longrightarrow & X \times Y \times T & \\ M & \downarrow & & \downarrow & M \\ & \Xi \times H \times T & \longrightarrow & X \times Y \times T & \\ & & \underset{B}{} & & \end{array} \tag{2.8}$$

このため、ビームフォーマーとマッチドフィルターの計算順序は、計算量の観点から効率の良いものが選択される。

## 2.4　ヒルベルト変換による複素信号の再構成

通常我々が、電気信号を音波に変えて利用する信号は、音圧を基とした実レベル信号である。すなわちセンサーにより受信される信号は、局所的瞬間的な媒体の膨張と収縮を表した量であり、これを電気信号との対応をつけることによって電圧に変換して利用する。これに対して電気信号を表す数式は、簡潔さや計算上の観点から、複素数による表記が好まれて用いられる。

信号処理によって複素数を扱うことは、複素偏角を用いる事によって位相の検出が容易になるなどの利点がある反面、実数で表現された信号系列から複素数を構成することの計算コストが伴う。実数信号から複素数信号を求めるための方法として、従来

は直交検波とローパスフィルターを併用することが普通であったが、デジタル信号処理手法の発展により、ヒルベルト変換を用いる例が増えている。本論文でも、この例に従い、実数信号から複素数信号を構成するための方法にヒルベルト変換を用いる。次にその導入を行う。

複素数で表現された量は、電場に対する磁場といったような、実数に対応する虚数を表現する物理量が存在しない限りそのまま送信することは難しい。実際的装置においては、実数部だけを取り出してそれを音圧などに変えて送信を行う。

受信側においては、観測データとして得られる実数数列から送信側で表現されていた複素数を再構成する必要性がしばしば生じる。虚数部は実数部とはすべての帯域にわたり位相だけが異なり実数部とは直交している必要がある。すなわち解析関数の再構成が課題である。これを行うのがヒルベルト変換 [Mor76][Tak06] である。ヒルベルト変換は、次の形の積分変換であり、その極限値を Cauchy 主値として取り出す操作といえる。

$$\mathfrak{H}x(t) = \frac{-j}{\pi} \int_{-\infty}^{\infty} x(t-\tau)\, \frac{1}{\tau}\, d\tau \tag{2.9}$$

ヒルベルト変換は正の周波数に関しては位相を $\pi/2$ だけ遅らせ、負の周波数に関しては位相を $\pi/2$ だけ進めるフィルタとして定義できる。またそのデジタルフィルタとしての実装も可能であり [Mik05]、階段関数を逆フーリエ変換して得られる係数列を FIR フィルタの係数として用いることによって実現する。この方法によって、実信号列 $x(t)$ に $z(t) = x(t) + j\mathfrak{H}x(t)$ を対応させ、我々は解析信号列 $z(t)$ を得る。

実信号列に代えて解析信号列を用いることによって、信号処理において次の点が有利になると考えられる。(1) 絶対値を計算することによって任意の時点における瞬間振幅が求められる。(2) 実数部と虚数部の比を計算することによって任意の時点における位相と瞬間周波数が求められる。(3) 解析信号に対して複素数体上の算術が適用でき周波数変換等の処理が容易になる。

近年、デジタル信号処理の進展によりヒルベルト変換を用いて複素数で信号処理を行う研究が増えており、ヒルベルト変換を周波数解析に有効に用いた例 [NH06] も報告されている。我々は、受信信号からヒルベルト変換を用いて複素信号を構成し、これを超音波による距離測定に用いる実験を行いヒルベルト変換の有効性を検証した [MSH10]。

## 2.5　ドップラーイメージング

ドップラーイメージングとは音響イメージングの一種であり、位置と同時に反射体の速度を測定し図示する手法のことをいう。

ドップラー効果は、波動のスペクトルに作用して、受信波の周波数を変化させる。送信機と受信機を向かい合わせに配置すると、動く送信機から発せられる音波の周波数は、送信機の速度の影響を受けて変化する。送信機と受信機を固定して動く反射体の反射波を受信する場合には、音波の周波数は動く送信機の直接波を受信する場合の約 2 倍の変化を受ける。この変化した周波数から反射体の速度を推定することをドップラー速度推定と呼ぶ。

一般に、ドップラー効果の影響は受信波の周波数に対して乗法的に作用する。送信器と受信器を固定し、運動する反射体の速度を計測する場合、反射体の運動速度を $v$、音速を $c$、運動方向と音波の到来方向がなす角度を $\theta$ とすると、送信波の周波数 $f$ と受信波の周波数を $f'$ の間に次の関係が成り立つ。

$$f' = \rho f = \frac{c + v\cos\theta}{c - v\cos\theta} f \tag{2.10}$$

この式に現れる乗数 $\rho$ でドップラー量を表し、$\rho$ で決まるドップラー効果を $\mathfrak{D}_\rho$ で表わすことにする。

ドップラー効果は、受信波の到来方向に働くため、センサー平面に平行に動く物体の速度は検知できない。これを検知するには、センサーを増やして複数の位置に受信する必要がある。

本研究では、次の方法によりドップラーシフトを定式化して扱う。ドップラー効果は受信波を時間原点を中心に信号全体を $1/\rho$ だけ伸縮させ、また、ドップラー効果の前後で信号の二乗積分が保たれるという前提の下でドップラー効果による信号長の変化を補うため振幅が $\sqrt{\rho}$ 倍に変化すると考えると都合が良い。そこでドップラー効果 $\mathfrak{D}_\rho$ は、送信波 $g(t)$ に次のように作用して受信波 $h(t)$ となるものとする。

$$h(t) = \mathfrak{D}_\rho\, g(t) = \sqrt{\rho}\, g(\rho t) \tag{2.11}$$

この式の中で、ドップラー量 $\rho$ は時刻 $t$ に対して乗数として働き、信号の周波数を乗法的に変化させていることに注意する。その結果、ドップラー効果によって、もともと低い周波数の信号はあまり変化せず、もともと高い周波数の信号はより多く変化することになる。

ドップラーシフトを伴った受信波の解析は、広帯域あいまい度関数 [Alt73] [You97] によって早くから行われてきた。近年は連続ウェーブレット解析によって再定式化された方法によって分析がなされている [NSG98]。

現実的な環境において伝搬する信号には必ず減衰と遅延が伴うため、減衰率を $\alpha$ とし、遅延時間を $\tau$ としてドップラー効果を伴う伝搬遅延を次式の形で定式化することもある。

$$h(t) = \alpha \mathfrak{D}_\rho \, g(t-\tau) = \alpha\sqrt{\rho}\, g(\rho\,(t-\tau)) \tag{2.12}$$

ドップラーイメージングでは受信波の周波数変化を検知し、その量を図示する。一般に周波数変化の検知は単一周波数の連続波の場合に一番易しく、複雑なスペクトルを持つ連続波の場合に一番難しい。単一周波数の連続波の場合の周波数測定は、受信波をフーリエ変換して得られるスペクトルのグラフからピークを検出してその周波数を調べることによって得られる。

周波数変化を調べる別の手段として、一定の間隔をおいて送出されたパルスの間隔変化を調べる方法もある。この場合、送信波として用いられるのはいわゆるパルス波であり、連続波ではない。

複雑なスペクトルを持つ連続波の場合、スペクトル解析によって周波数変化を捉えるのはより難しい課題になる。まず最初に試みられるのは、複雑なスペクトルの中に特徴を見つけ、その特徴がドップラーシフトによってどのように変化するかを調べることである。例えばスペクトルが上限と下限で切り立っている場合には、スペクトルの上限値と下限値の平均を求めることが、最も簡便な周波数検査法となるが、この方法を一般化してスペクトルの形が様々である場合に適用することは困難である。

上述の困難を克服するために、いくつかのアプローチが試みられている。本章の最後でそれを紹介する。

## 2.6 医用超音波におけるドップラーイメージング

医用超音波の研究領域では、生体内の循環器系の流速を求めるために早くからドップラーイメージングの手法が確立した。

循環器系の流速の時間的変化を求めるためには連続波ドップラー法が良く用いられる。この方法は、一定の周波数の超音波を連続的に送出することにより、受信側でスペクトル解析により周波数領域におけるピークを検出することによってドップラー量を検出するものである。次に連続波ドップラー法では得ることのできない距離方向の解像度を得るために、パルスドップラー法が考案された。これは、連続波をパルスに分割し、一定のパルス間隔をおいて送出することによってパルス間隔に相当する距離

方向の解像度を得ようというものである。

この領域におけるドップラーイメージングの代表例として挙げられるカラードップラー法は、これらのアプローチとは異なり、受信されたパルスを時間領域で自己相関計算を行うことにより、流速を計測しようというものである。カラードップラー法については、そのアルゴリズムの定式化が早くから与えられ [KNKO85]、能力と限界についての研究が行われてきた [Mit90]。

本論文ではカラードップラー法を取り上げ後に提案手法との比較検討を行う。カラードップラー法によるドップラーイメージングでは、反射体の速度は連続するパルス波の自己相関計算を行うことにより計算される。この手法では、自己相関計算を複素領域で行い、その偏角の値を色づけすることにより速度画像を得る。

カラードップラー法は一定の間隔でパルス波を連続的に発生させる。パルス波とはキャリアを包絡線関数により振幅変調して得られる波形であり、$z(t)$ を複素包絡線関数として次式により定義される。

$$g(t) = z(t)e^{j\omega_0 t} \tag{2.13}$$

この参照波の実部を間隔 $T_1$ を持って連続送出し、送信波とする。

受信波からは包絡線関数 $z(t)$ を再構成する。包絡線関数はドップラーシフトの影響を受けてもとの $z(t)$ とは異なるものとなっているため、これを $z'(t)$ と表記することにする。包絡線を再構成するために、伝統的に用いられてきた方法は直交検波とローパスフィルターによって行う方法であるが、デジタル信号処理ではより正確なヒルベルト変換を用いる方法が好まれる。すなわち受信波をヒルベルト変換することによってまず複素信号を再構成し、その絶対値を求める事で包絡線を得る。この包絡線を再度ヒルベルト変換することによって複素包絡線を構成する。

こうして受信波から複素包絡線 $z'(t)$ が得られると、カラードップラー法ではこの関数に送信時のパルス間隔 $T_1$ を遅延時間として干渉法を適用する。

$$R_1(t) = \int_{t-T_1/2}^{t+T_1/2} \overline{z'}(\tau - T_1)\, z'(\tau) d\tau \tag{2.14}$$

ドップラーシフトが作用してパルス間隔が短くなると、積分の結果得られる複素位相は 0 よりも進行することになる。逆にパルス間隔が長くなると、複素位相は遅れて 0 よりも小さくなる。こうして得られた $R_1(t)$ の実部と虚部の比を求めることによりドップラー位相が計算できる。得られたドップラー位相の符号に従って、赤、青の輝度を与えイメージングを行う。伝統的に緑の輝度は分散を表現するために用いられる。

カラードップラー法の利点は、自己相関のみを用いた簡単な計算によって高速処理が可能である点である。一方で、包絡線位相の範囲が理論上 $[-\pi/2, \pi/2]$ に限定され

ることから、ドップラー量の推定精度とドップラー量の推定範囲にトレードオフが生じることが欠点となる。この問題については後に、提案手法との比較の中で触れる。

近年、カラードップラー法に光学干渉法をあわせて適用し、距離分解能を上げる提案もなされている [RYBI02] [YRI03]。この方法はカラードップラー光学干渉法 (CD-OCT) と呼ばれる。カラードップラー光学干渉法は、ドップラー速度の検出のためにキャリア位相を用いる方法であり、同目的のために包絡線位相を用いるカラードップラー法とは区別される。この場合も、キャリア位相が $[-\pi, \pi]$ に限定されることから $2\pi$-曖昧性と呼ばれる現象が発生し、曖昧性の解消が問題になる。

## 2.7　ドップラーイメージングにおける精度向上の試み

ドップラー効果の持つ特性は、受信波がドップラーシフトの影響を受けないことを前提に設計された音響イメージングの各種の方法に大きな影響を及ぼす。

ドップラー不変量を用いてドップラーイメージングを実現する方法としては、古くから広帯域相互あいまい関数による方法が知られ、ルジャンドル多項式にこれを適用する例が提案されている [Alt73]。広帯域相互あいまい関数は後に連続ウェーブレット解析に一般化され [You97]、短時間フーリエ変換に対する優位性が示されている [UG04]。

ドップラー効果のマッチドフィルター処理に与える影響は古くから研究され、参照波のドップラー不変性に関しての一般的記述がなされている [Alt73]。中でもチャープ波のドップラー寛容性についての先駆的な研究 [YS06] においては、ドップラー量と共に変化するパルス圧縮についての検討があり、双曲周波数変調チャープ波のドップラー寛容性における優位性が示されている。

マルチキャリア信号は複数のキャリア信号を加算して得られる信号で、特に高速デジタル通信に有利な直交周波数多重分割方式 (OFDM) で用いられる OFDM 波が最近注目を集めている。直交周波数分割多重方式は、無線 LAN や次世代携帯電話で用いられる通信方式であり、通信効率の高さと処理の高速性により注目されている。比較的簡単な信号処理で高い伝送効率が得られるため、無線 LAN や Bluetooth などの分野に応用が進んでいる。最近ではこれを携帯電話の通信に用いる LTE の技術が広まり、高速デジタル通信の標準的な通信方式となりつつある。

超音波イメージングに OFDM 波を用いることにより、複数の音源から同時に超音波を送信して開口合成を行い、複数の送信波のチューニングを施すことによって高い精度での反射体イメージングを実現できることが示されている [ISH10] [ISH11]。OFDM 波は近年合成開口レーダーによる地形探査などへの応用も提案され、懸案で

あったアーティファクトの発生をキャリア周波数分割により抑制する方法が提案されている。[RMP12]

このようにこんにち広範に用いられる OFDM 波であるが、対象物の移動がドップラー効果をもたらし波形が歪むため、高速で動く物体から送受信を行う場合には応用が難しい。ドップラー効果は反射波の周波数に対して乗法的に作用することに注目する。ドップラー効果により OFDM 波の等間隔で並んだキャリアは高周波数ほど間隔の広いスペクトル分布を持つようになる。このことにより各キャリアの直交性が損なわれる [RK99] ばかりでなく、マッチドフィルターの演算における同一波形の検出性能が損なわれる。このため広範囲のドップラー量の測定には、想定されるドップラーシフト範囲に対応した多大な数の相関演算を必要とする。近年、チャープ波との混合により測定装置の移動速度に対する耐性を得るための提案が行われているが [LZqBD12]、得られる効果についての定量的な解析は課題として残る。

チャープ波とは通常、時間と共に周波数の変化する FM チャープ波のことを指す。チャープ波は広帯域超音波を用いたイメージングのための参照波として最も多く利用されている。波形生成の容易さと、理論的性質の解明が進んでいることがその理由として挙げられる。OFDM 波同様にチャープ波は連続波であるから、これをドップラーイメージングに用いて時間解像度を得るためにはパルスの再構成を必要とする。これについては、従来用いられてきたインパルスに代えてチャープ波を用いることの有効性が示され [Col91]、同波を用いてドップラー速度推定を行うための理論的基礎が示されていた [Wil93a][Wil93b]。

近年になって、送信波にアップチャープ波及びダウンチャープ波を用いることによる推定精度改善の試みがなされている [ITMM02]。チャープ波は時間と共に周波数の変化する信号であるため、ドップラーシフトの影響を受けて開始周波数と終了周波数が変化する。このためマッチドフィルターによるピーク再構成においてピークの現れる時刻に系統的な偏移が表れる。この偏移を除去して正確なドップラー量を検出するためには、同一の偏移傾向を持つ 2 波を送出し、マッチドフィルターによって得られるピーク間隔を検出すれば良い。この考え方に基づいて、チャープ波 2 波を用いたドップラー速度計測が提案され有効性が検証されている [HKK08][HKK09][SKOK10]。

ドップラーイメージングに関連性の高い研究として、音楽情報処理におけるスペクマート法の研究が挙げられる [TNS03][KSNS04][STKN04][SKNS05][SKOS06a]。スペクマート法は、周波数領域でのスペクトルの対数周波数分布に着目し、乗法相似性を持つ波形を対数変換を行ったスペクトルの畳み込み和として検出する方法である。スペクマート法は、周波数が乗法的に変化し、豊かな調音構造を持つ楽音の分析に有効な分析である [SKOS06c][SKOS06d][SKOS06b]。本論文で提案する乗法相似性を持

つ送信波の設計手法は、スペクマート法における乗法相似波形の検出手法と類似性が高く、アルゴリズムの構成の上で参考とされる。

# 第 3 章

# ログステップマルチキャリア波

前章で述べたように、ドップラー効果は反射波の周波数に対して乗法的に働きスペクトルを変化させる。すなわち、高い周波数のキャリアほど大きく周波数が変化し、低い周波数のキャリアはあまり変化しない。このことは、送信波と受信波の相関演算による信号の検出に深刻な困難をもたらす。たとえば OFDM 波の場合には、送信波において等間隔で並んでいたキャリア周波数は、ドップラーシフトにより変化して受信波においてはもはや等間隔ではなく、キャリア間の直交性が失われる。また、ドップラーシフトが大きくなると、相互相関による信号の検出そのものができなくなる。

本章では、この問題を解決するため、ドップラーシフトに強いマルチキャリア波を提案する。我々はドップラーシフトに対する２つの要請を満たす必要がある。１つめは、ドップラーシフトが大きくなっても、相互相関による信号の検出を可能とすることであり、２つめは、ドップラーシフト量を精密に測定できる手段を与えることである。提案する広帯域信号は、ドップラーシフトの乗法性に対応して乗法相似性を持つものである。乗法相似性を持つ波形に、既知のものではログスイープチャープ波が挙げられるが、更に良い性質を持つ波形を考案しその性質を解明することが、本章の目的である。

表現の簡潔性のために、信号は複素化された形で記述する。但し位相表現の目的のために図版は実数によるものを提示する場合があるが、実数計算による場合と複素計算による場合で差異が生じるときには、その都度注記することにする。

## 3.1　一般位相ログステップマルチキャリア波

キャリアが等間隔で並んだ等ステップマルチキャリア波に対し、キャリアを指数間隔で並べた波形が考えられる。これを、キャリア周波数の対数が等間隔になることか

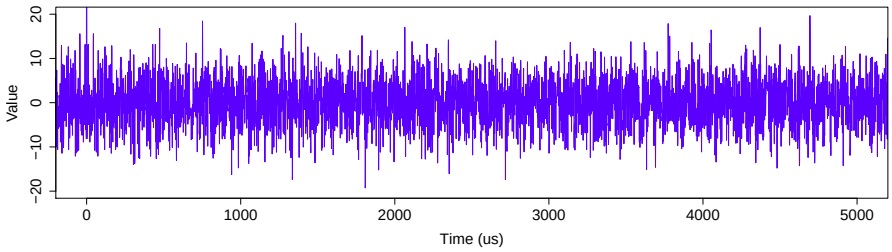


図 3.1　一般位相ログステップマルチキャリア波

らログステップマルチキャリア波と呼ぶ。このような波形の最も身近な例は音楽に見られる。平均律の音階は、半音音程が $2^{(1/12)}$ の周波数比を持つ音階であって、伝統的音階を良く近似することから電子楽器の調律などに用いられている。この平均律の音程の構成音をすべて同時に鳴らせた合成音がログステップマルチキャリア波である。

ログステップマルチキャリア波の特性を決める重要なパラメータに、帯域幅とキャリア数がある。帯域幅 $p$ は最低音と最高音の周波数比であり、1 オクターブを 2、2 オクターブを 4 と表す。キャリア数はその名の通り合成波形に含まれるキャリアの数を表し、平均律の場合では 1 オクターブを構成する 12 の音がこれに相当する。

ドップラー効果が、反射音の音程を乗法的にシフトする、すなわち移調することを考慮すると、あるキャリアを移調した先の音程がすでに送信波に含まれていることが、ドップラー効果を測定する上で重要な役割を果たすことになる。

信号処理を行う上で重要なパラメータに、各キャリアの位相がある。各キャリアの位相が一斉に一致すると合成波のその時点に鋭いピークが現れるため、通常避けられるが、理論的記述のためには位相を自由に選べるようにしておく方が好ましい。

ログステップマルチキャリア波とは複数のキャリアからなる信号で、隣接するキャリア周波数が定数比を持つものである。次式によって一般形を与える。

$$g_{p,q}(t) = \sum_{k=0}^{q-1} a_k \exp\left[2\pi p^{k/q} f_0 t + \phi_k\right] \tag{3.1}$$

ここで $p > 0$ は帯域を決める実数であり、整数 $q$ によりキャリア数を定める。また $a_k$ は各キャリアの振幅、 $f_0$ は基本周波数、$\phi_k$ は各キャリアの初期位相である。一般的なログステップマルチキャリア波では、振幅 $a_k$ 及び初期位相 $\phi_k$ を自由に決定することができる。特に初期位相 $\phi_k$ を乱数に選んだ場合、波形は帯域制限されたホワイトノイズに近いものとなる (図 3.1)。

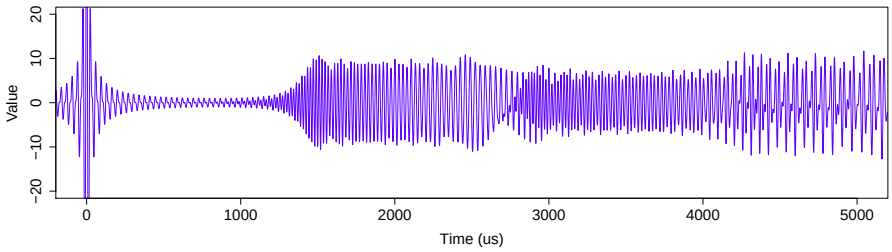


図 3.2　整列位相ログステップマルチキャリア波

## 3.2　整列位相ログステップマルチキャリア波

初期位相 $\phi_k$ は時間原点における各キャリアの位相を決定する。すべてのキャリアで $\phi_k = 0$ とすると、時間原点におけるキャリアの位相が 0 に揃えられ、時間原点に鋭いピークが発生する。ピークの周りでは、各キャリアの位相は周波数に応じた速さで分散してゆくため、波形は急峻に減衰して無音に近い状態となる。更に時間が経過すると、位相の重なり合うキャリアが増えてゆくため波形は徐々に乱雑さを増してゆく (図 3.2)。

整列位相ログステップマルチキャリア波の基本的なアイデアは、この乱雑さが増えてゆく状態のログステップマルチキャリア波を、キャリア位相の統一的な扱いが可能であるぎりぎりのところで用いて、イメージングを始めとする各種応用に役立てようとするものである。

我々は、単一の定数 $T_0$ によって各キャリアの位相を次式により決定する。

$$g_{p,q}(t, T_0) = \sum_{k=0}^{q-1} \exp\left[2\pi p^{k/q} f_0 (t + T_0)\right] \tag{3.2}$$

すなわち $\phi_k = 2\pi p^{k/q} f_0 T_0$ とする。これを整列位相ログステップマルチキャリア波と呼ぶ。簡単のため各キャリアの振幅は $a_k = 1$ に固定して考える。このように各キャリアの位相を、単一の定数 $T_0$ により決定すると、各キャリアの位相が、$\phi_k = 2\pi p^{k/q} f_0 T_0$ と揃えられ、ドップラーシフトの検出に都合良く整列する。これが整列位相と呼ぶ所以である。

残るパラメータのうち $p$ と $f_0$ は扱う超音波の帯域により決定する。ちょうど 1 オクターブの帯域を使用する場合、$p = 2$ であり、$f_0$ は最低音の周波数を決める。我々が調節できるのは、キャリア数 $q$ と初期時刻 $T_0$ であり、それぞれログステップマルチキャリア波に独特の特徴をもたらす。

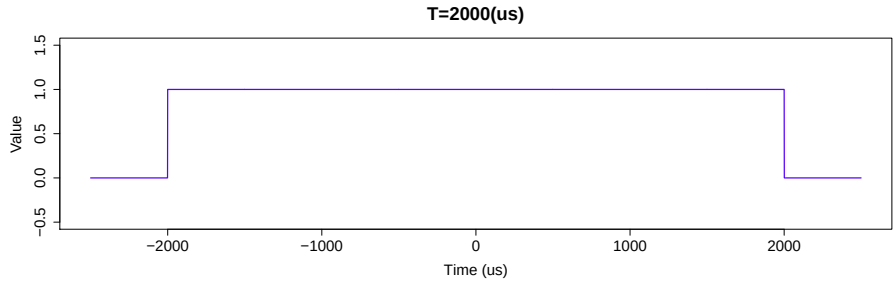


図 3.3　送信波の構成に用いる矩形窓関数

ここで $T_0 = 0$ と定めたものは整列位相ログステップマルチキャリア波の中でも特別の意味を持っている。式 (3.2) を見るとわかるように、$T_0 = 0$ の場合に式は簡単になり、生成される波形は時間原点に鋭いピークを持つものとなる。このような波形はエネルギー効率が悪いため送信波として用いるには都合が悪いが、ログステップマルチキャリア波の定性的な理解のために役立つものとなる。$T_0$ を大きくすると、時間原点に現れていたピークは遠ざかり、乱雑な波形が現れるようになる。我々の応用のためには、程よい大きさに設定された $T_0$ が好ましい。

## 3.3　信号長と窓関数

ログステップマルチキャリア波は連続波であり、全体を送信するには無限の時間を必要とする。OFDM 波とは異なり、キャリアの周波数比は一般的に無理数となるため、ログステップマルチキャリア波は周期のない関数であり、このうちのどの区間を切り出して用いるかということが重要な意味を持つ。送信区間を形式的に取り扱うため、本論文では送信区間において値 1 をとり、それ以外で 0 となる窓関数 $r_{[-T,T]}(t)$ を導入し (図 3.3)、次式によりこれを定める。

$$r_{[-T,T]} = \begin{cases} 0 \text{ for } t < -T \\ 1 \text{ for } -T \leq t \leq T \\ 0 \text{ for } T < t \end{cases} \tag{3.3}$$

この窓関数によってイメージングの際に用いる送信波を次の方法により構成することができる。すなわち送信波 $g(t)$ を、この窓関数により切り取られるログステップマルチキャリア波とする。

$$g(t) = r_{[-T,T]}(t) \cdot g_{p,q}(t, T_0) \tag{3.4}$$

以後、特にことわりのないときには、送信波はいつでもこの形で定められるものとし、$T$ を半信号長、$2T$ を信号長と呼ぶことにする。特に初期時刻 $T_0$ と半信号長 $T$ との

関係が、後のログステップマルチキャリア波の特徴を求める上で重要となる。

初期時刻 $T_0$ を半信号長 $T$ よりも大きく選ぶと、信号区間内に鋭いピークのない波形が得られる。送信波に適切なのは鋭いピークのないものであり、本研究でも実際に実験に用いるのはこの波形である。

図 3.4 は $T_0 = 0$ の場合に整列位相ログステップマルチキャリア波をキャリア数 $q$ を変えながら列挙したものである。なお、基本周波数 $f_0 = 28600$ (Hz) とした。図 3.5 は、$T_0 = 3000$ (us) の場合に得られるログステップマルチキャリア波をキャリア数 $q$ を変えながら列挙したものである。半信号長 $T = 2000$ (us) に対し、$T_0 = 3000$ (us) と設定して $T < T_0$ が成り立つようにしている。なお、基本周波数は同じく $f_0 = 28600$ (Hz) である。

前述のように初期時刻 $T_0$ の選び方はログステップマルチキャリア波の波形概形を大きく変化させる。いっぽうスペクトルも変化するが、その変化は主にスペクトルの位相に現れ、スペクトル絶対値はあまり変化しない。$T_0 = 3000$ (us) の場合に得られるスペクトルを図 3.6 にキャリア数 $q$ を変えながら列挙する。キャリアの周波数に応じて高いスペクトルが見られるのが確認できるが、キャリアの密度が高くなると、隣り合うキャリア同士が干渉して、やがてスペクトルは連続するようになる。本論文では隣接キャリアが干渉を始める寸前のところにキャリア数 $q$ を固定して用いることになる。

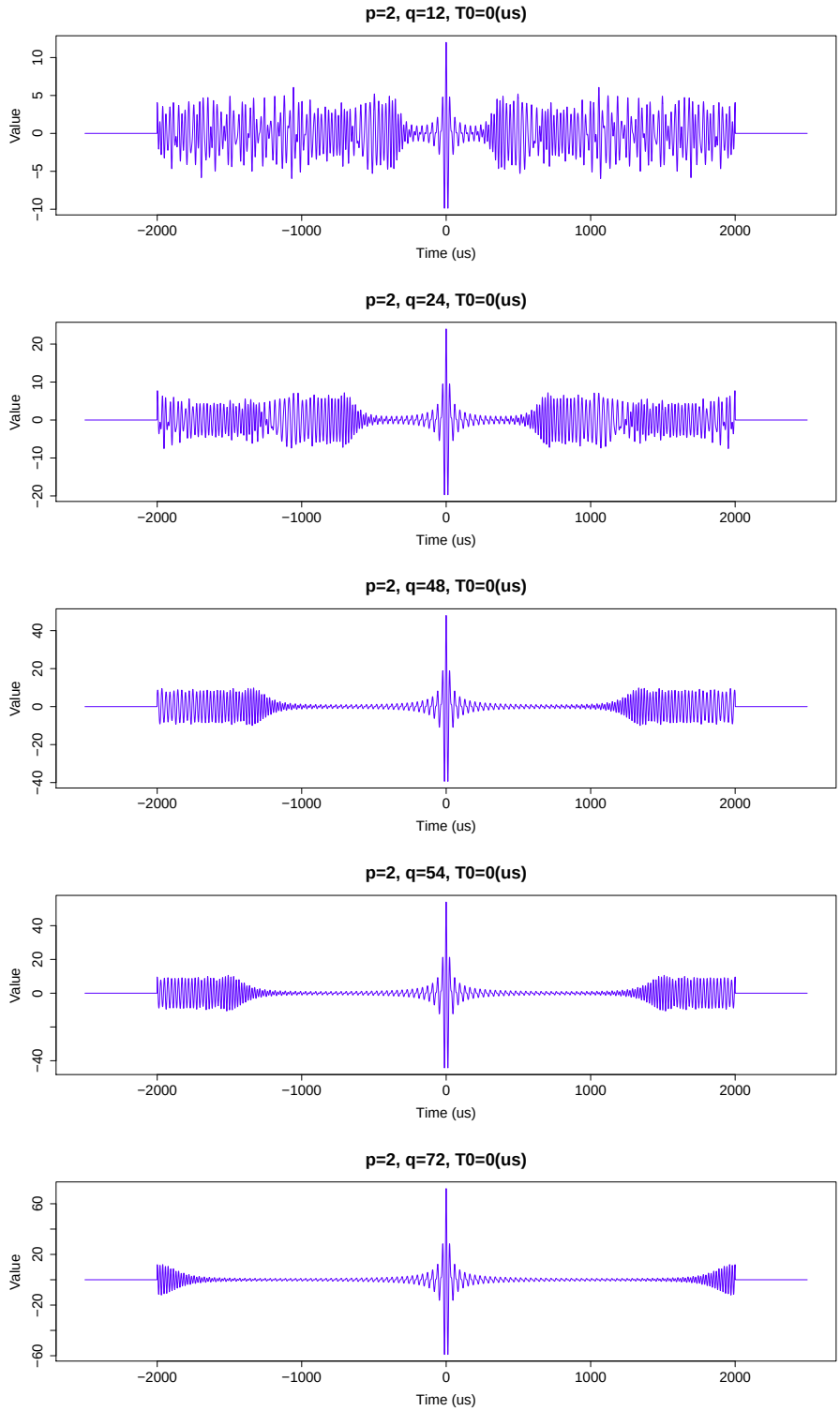


図 3.4　初期時刻 $T_0 = 0$ でキャリア数 $q$ を変化させたときの整列位相ログステップマルチキャリア波の波形

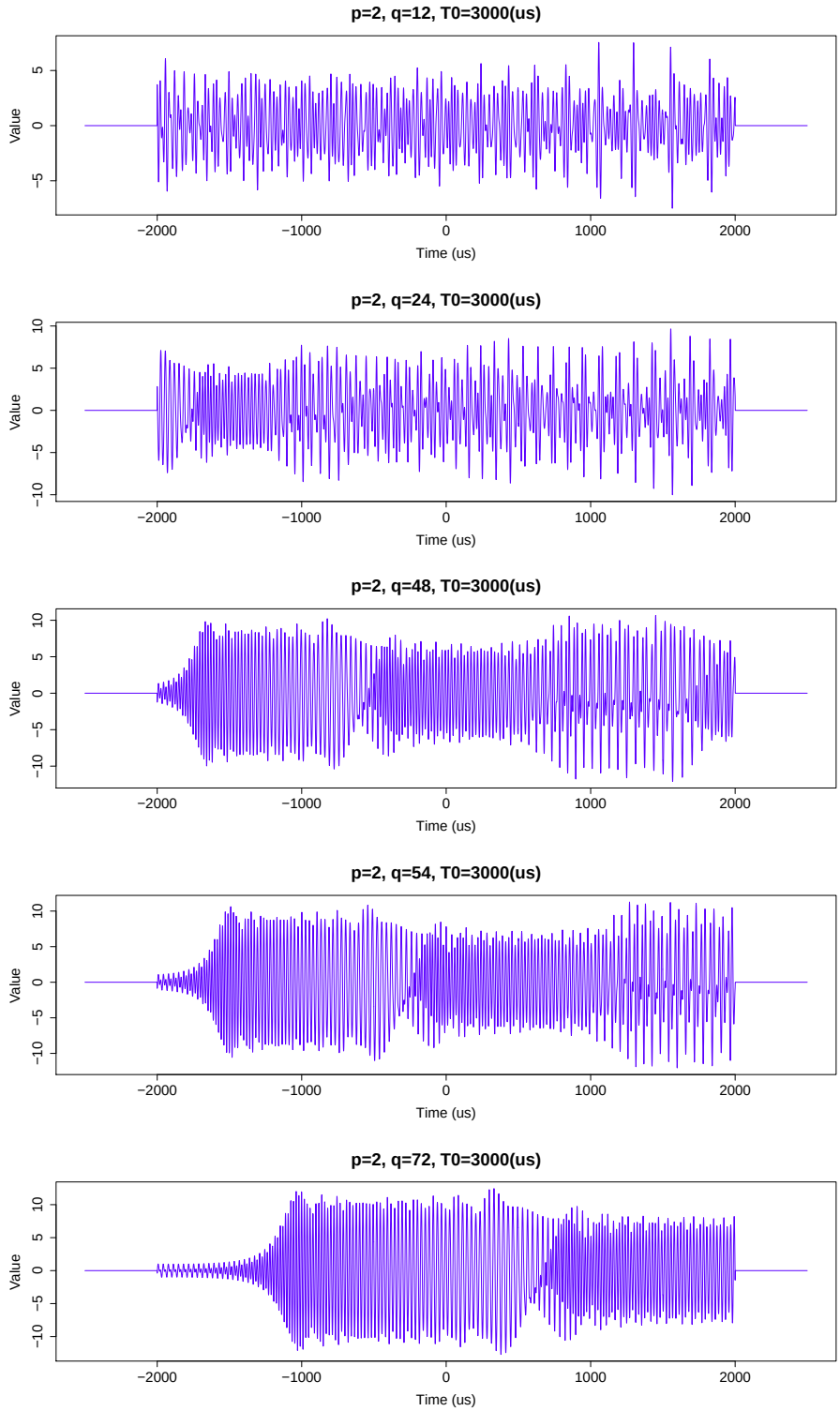


図 3.5　初期時刻 $T_0 = 3000$ (us) でキャリア数 $q$ を変化させたときの整列位相ログステップマルチキャリア波の波形

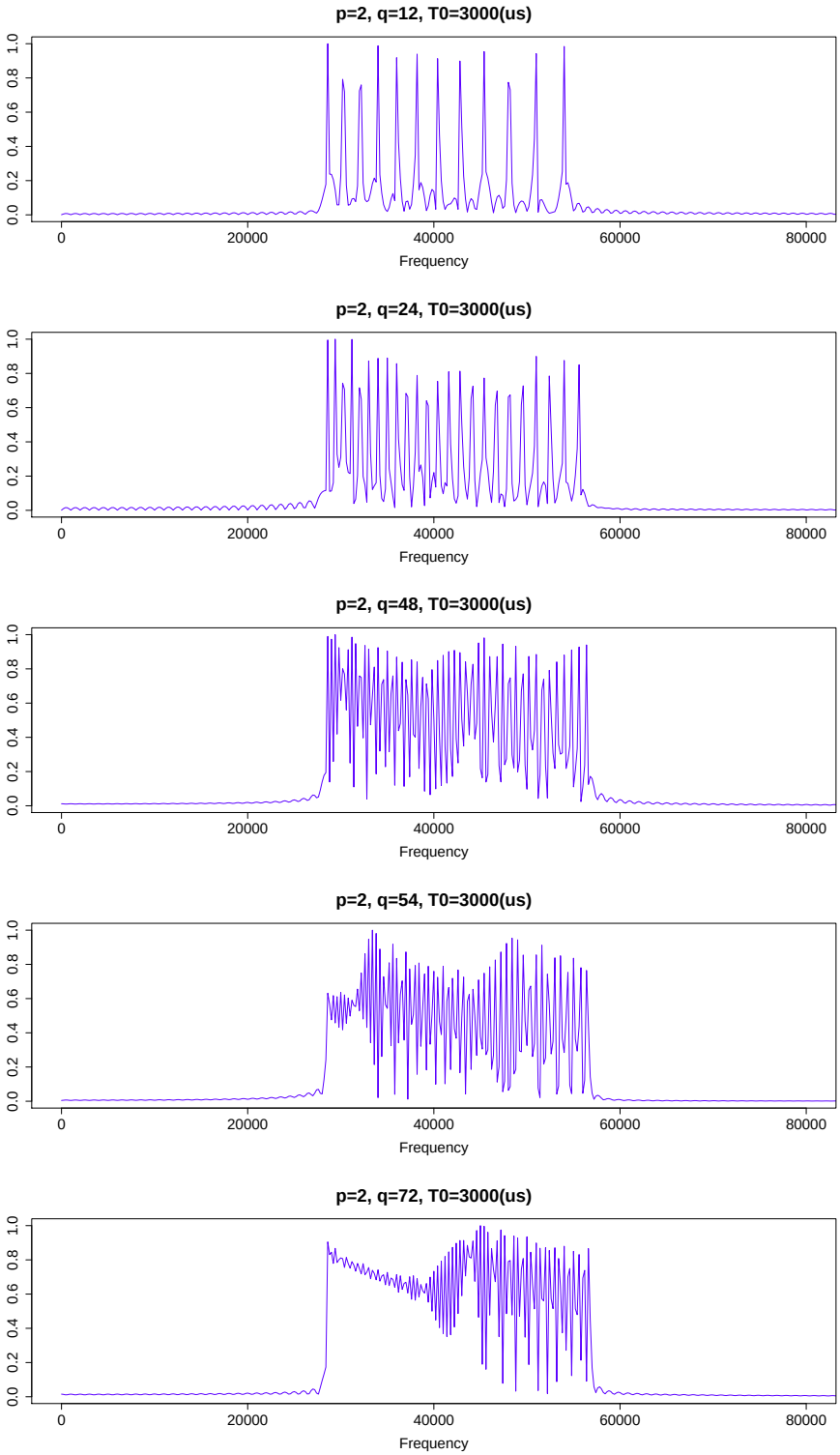


図 3.6　初期時刻 $T_0 = 3000$ (us) でキャリア数 $q$ を変化させたときの整列位相ログステップマルチキャリア波のスペクトル

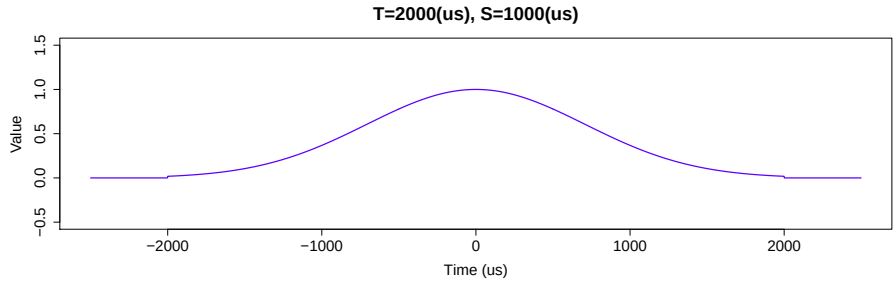


図 3.7　自己相関波形を改良する目的で用いられるガウス窓関数

## 3.4　ログステップマルチキャリア波の自己相関

イメージングにおいては反射波の処理の結果、反射体位置に鋭いピークが得られ、それ以外は 0 に近い値を出す方法が好ましいとされる。

送信波が全帯域で全反射し、ドップラーシフトの影響も見られずノイズもないときには、マッチドフィルターによる送信波と受信波の相互相関は送信波の自己相関と等しくなり、単に反射体までの距離に対応した遅延が見られるに過ぎなくなる。すなわち送信波の自己相関の結果得られる波形は理想的条件下で得られるイメージング画像を表現する。このため送信波のデザインでは自己相関の結果得られるピークの形について考慮し、イメージング性能を評価しておくことが望まれる。

送信波を $g(t)$ とすると、その自己相関 $A_g(t)$ は次の積分計算で求められる。

$$A_g(t) = \langle g|g \rangle (t) = \int_{-\infty}^{\infty} \overline{g}(\tau - t) g(\tau) d\tau \tag{3.5}$$

図 3.8 及び 図 3.9 に整列位相ログステップマルチキャリア波の自己相関波形を示す。図 3.8 は初期時刻 $T_0 = 0$ の場合にキャリア数 $q$ を変化させて列挙したもので、図 3.9 は初期時刻 $T_0 = 3000$ (us) の場合である。これらの図により、自己相関波形はキャリア密度に大きく依存していることが確認できる。キャリア密度が高くなれば、その分時間原点に得られるピークは高くなり、その周りでキャリアの位相が $[-\pi, \pi]$ 区間にまんべんなく分配される傾向が強まるため、サイドローブは低く抑えられる。いっぽうキャリア密度が低いときには、時間原点に得られるメインローブに対してサイドローブの比率が高くなる。

送信波がログステップマルチキャリア波と窓関数の積によって構成されることを考えると、窓関数の形を変更することによって自己相関波形をより良いものにすることができる。この目的のために良く用いられるものはガウス窓等の両端で減衰を伴う窓

関数である。図 3.10 に初期時刻 $T_0 = 3000$ (us) の整列位相ログステップマルチキャリア波に窓関数としてガウス窓を適用した場合の自己相関を示す。ログステップマルチキャリア波自己相関のサイドローブは、メインローブからは比較的離れた位置に現れることが多く、キャリア密度が高いほどその傾向は強まるが、ガウス窓の適用はこのようなサイドローブを抑制する効果が高く、組み合わせとして有効に機能していることが確認できる。

ただし本研究では自己相関波形の改良は主たる目的ではない。理論的な扱いを簡単にするために、以降の議論では窓関数として矩形窓を使用したものを対象とする。

## 3.5　相互相関と系統偏移

送信波と受信波の相互相関を計算することによって、到来時間のピークを再構成する。この計算において理想的な状態は、受信波が送信波と等しくなる場合に得られ、相互相関は自己相関と等しくなる。現実には、受信波は時間をかけて媒体中を伝搬し、伝搬経路において減衰し、ノイズが混入し、動体からの反射波にはドップラーシフトの影響も見られる。このため、相互相関計算に対するノイズやドップラーシフトの影響を調べることが重要な課題となる。

相互相関は、ある波形 $g(t)$ と別の波形 $h(t)$ の類似度を調べるために行われる計算であり、次の積分により定義が与えられる。

$$C_{g,h}(t) = \langle g|h\rangle\,(t) = \int_{-\infty}^{\infty} \overline{g}(\tau - t)h(\tau)d\tau \tag{3.6}$$

ここでは、波形 $g(t)$ と波形 $h(t)$ が何らかの関係を持つときに現れる性質を調べる事が課題である。特にドップラーイメージングのためには、$g(t)$ を送信波とし、$h(t)$ を受信波とするときに、$g(t)$ がドップラーシフトの影響を受けて $h(t)$ へと変化した場合に相互相関の値がどうなるかを調べておくことが必要になる。

そこで、ドップラー量を $\rho$ とし、伝搬遅延時間を $T_1$、減衰率を $\alpha$ とする。ノイズのない環境下では、ドップラーシフトの結果得られた受信波 $h(t)$ は次の式で表現される。

$$h(t) = \alpha\sqrt{\rho}\, g(\rho\,(t - T_1)) \tag{3.7}$$

ここで伝搬遅延時間 $T_1 = 0$ とし、減衰率 $\alpha = 1$ とすると、純粋にドップラーシフトの相互相関に与える影響を調べる事が出来る。この場合に式 (3.7) は $h(t) = \sqrt{\rho}\, g(\rho\, t)$ と簡単になる。この式はドップラー量 $\rho$ によるドップラー作用素を $\mathfrak{D}_\rho$ として $h(t) = \mathfrak{D}_\rho g(t)$ とも表現される。

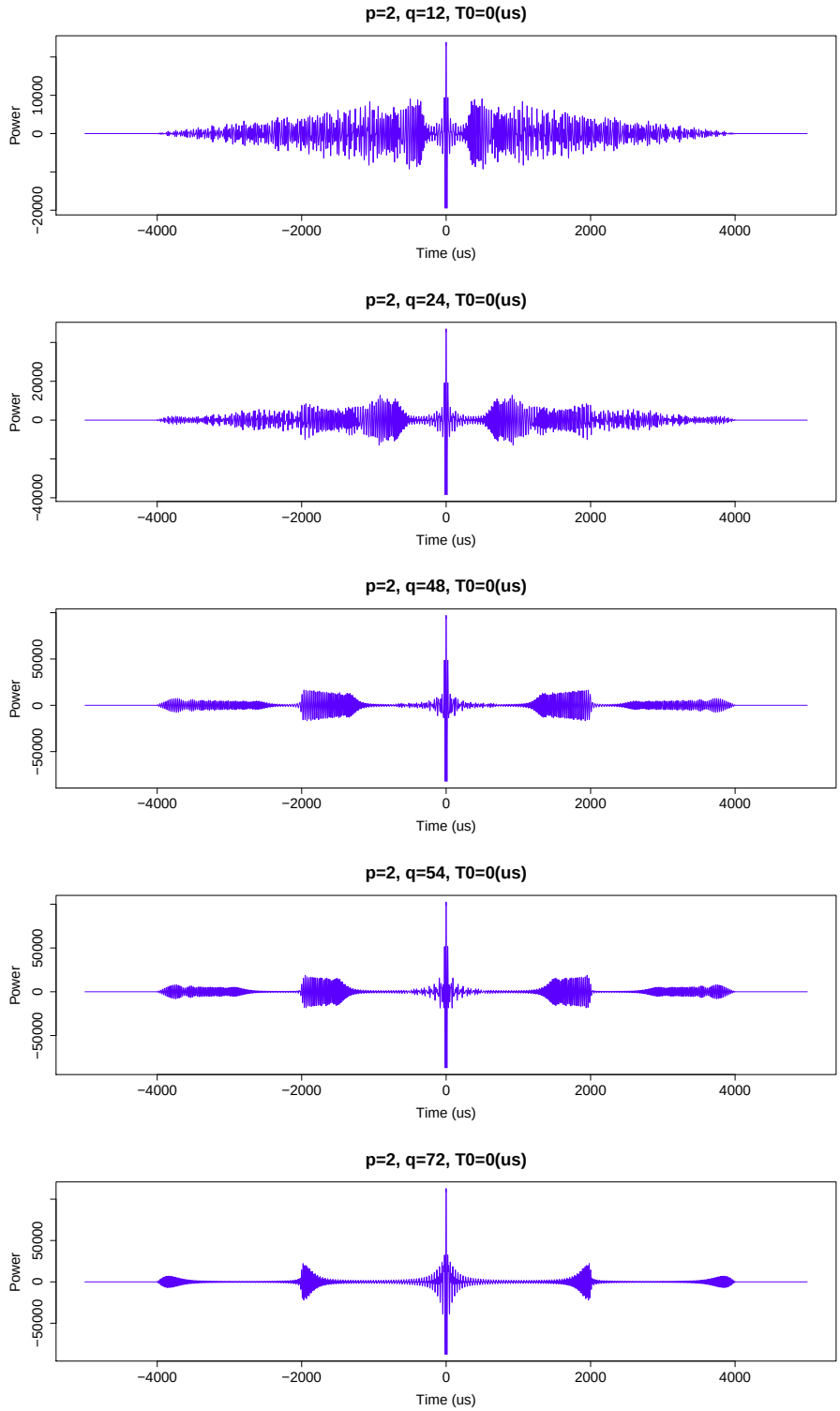


図 3.8　ログステップマルチキャリア波の自己相関の系列 $(T_0 = 0)$

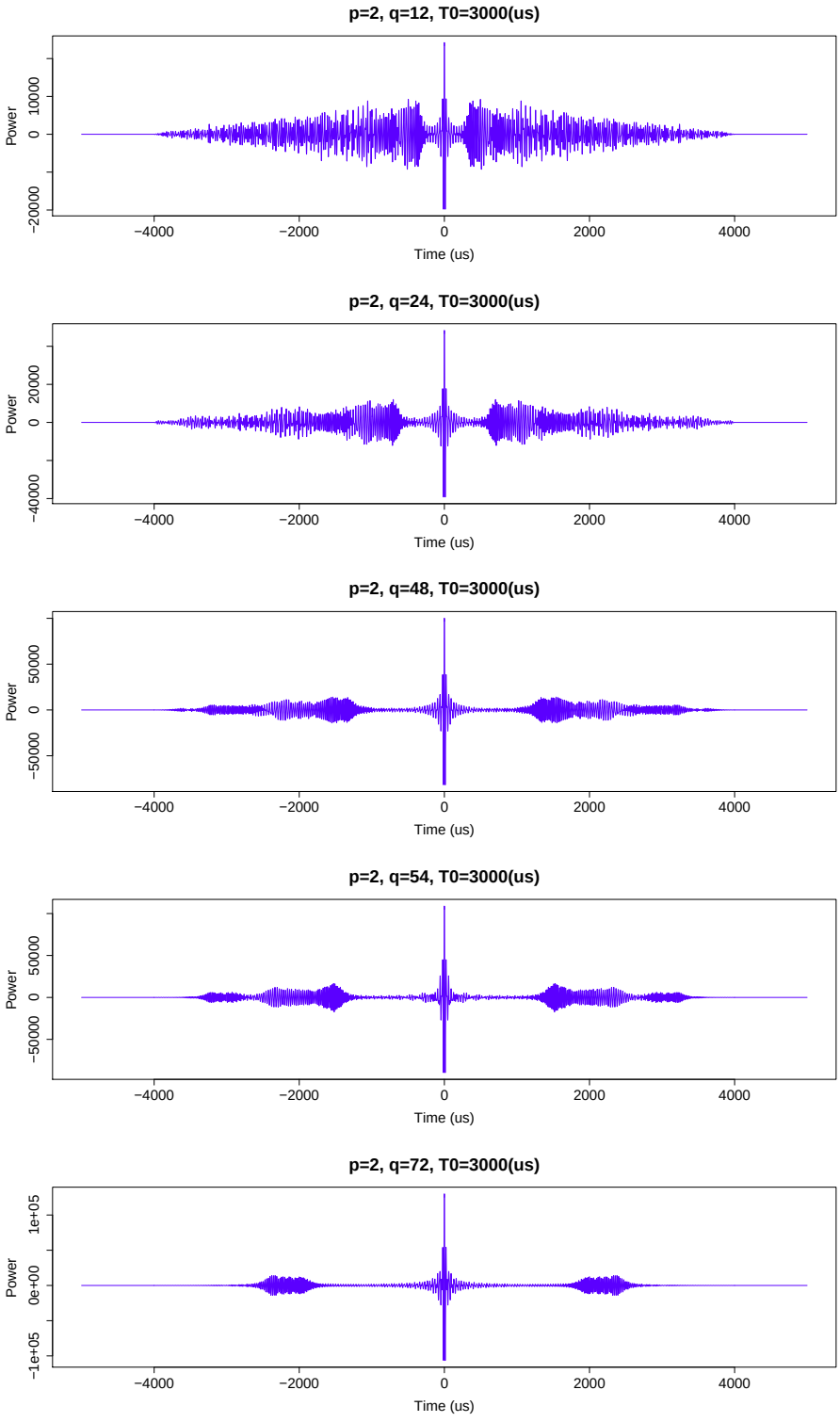


図 3.9　ログステップマルチキャリア波の自己相関の系列 ($T_0 = 3000$)

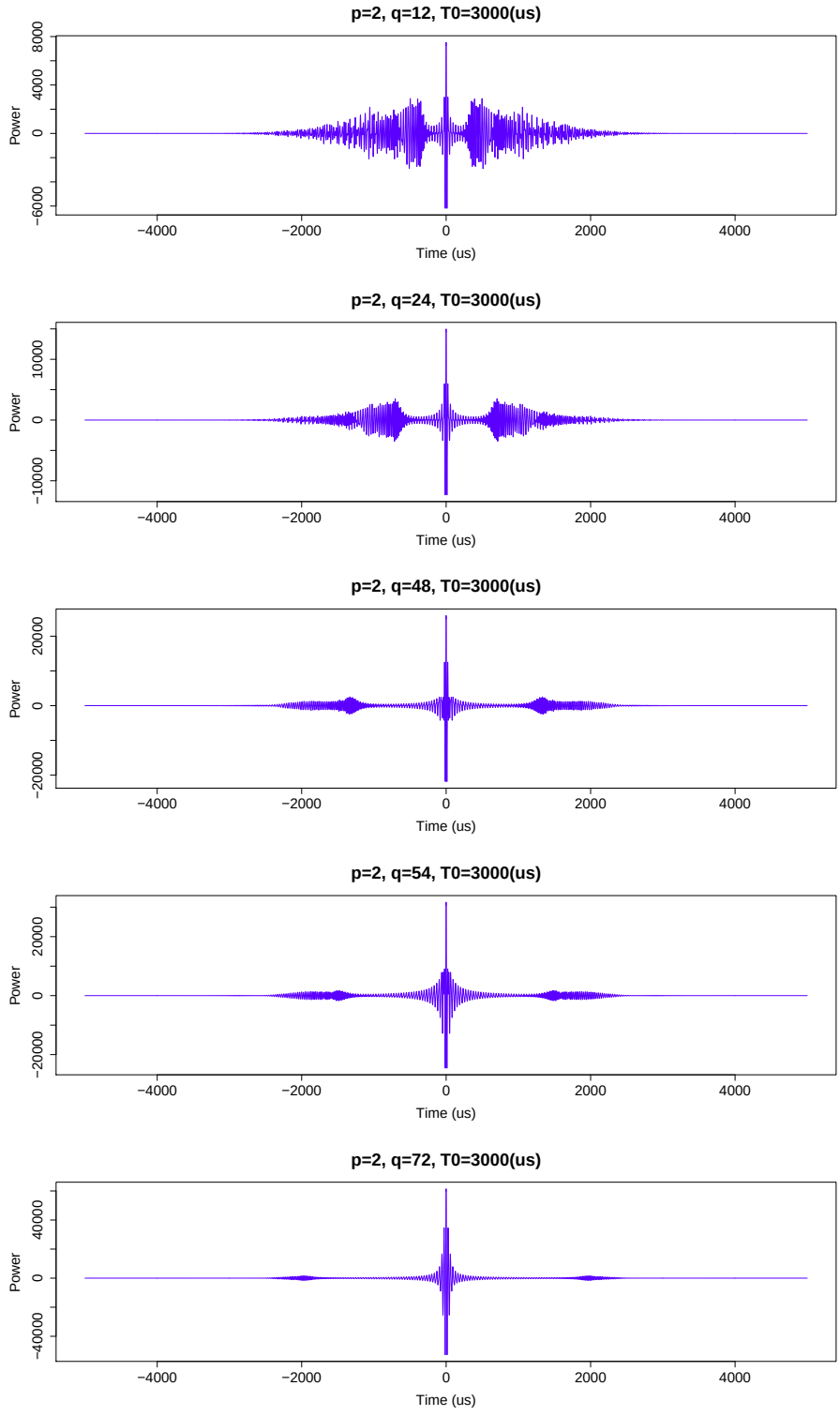


図 3.10　ガウス窓関数を適用したログステップマルチキャリア波の自己相関の系列 ($T_0 = 3000$)

この場合の相互相関の形を調べる事が本章の残りの部分の大きな課題である。すなわち次式の性質を調べる事になる。

$$C_{g,\mathfrak{D}_\rho g}(t) = \langle g|\mathfrak{D}_\rho g\rangle\,(t) = \sqrt{\rho}\int_{-\infty}^{\infty}\overline{g}(\tau - t)g(\rho\,\tau)d\tau \tag{3.8}$$

この式は波形 $g(t)$ に対応して決まる、実変数 $t$ と実変数 $\rho$ に関する 2 変数複素数値関数 $W_g(t,\rho) = C_{g,\mathfrak{D}_\rho g}(t)$ である。この関数 $W_g(t,\rho)$ は伝統的に広帯域曖昧度関数 [Alt73] と呼ばれてきたものとパラメーター変換の意味で同一の概念である。また、更にパラメーター変換を行って連続ウェーブレット解析 [You97] と呼ばれているが、混乱を避けるために、本論文の以降の部分では $W_g(t,\rho)$ を単に波形 $g(t)$ のドップラー解析と呼ぶことにする。

ホワイトノイズにドップラー解析を適用すると、$t = 0, \rho = 1$ の位置にピークが現れ、それ以外では値は 0 となる。これに対して、時系列的な特徴を持った波形では、時間 $t$ の方向にサイドローブが現れる。また、乗法相似性を持った波形には、ドップラーシフト方向に再び相関が見られることになる。相互相関値をドップラー量検出に用いる目的においては、遅延時間方向にはサイドローブが低く、ドップラーシフト方向に繰り返し現れるピークを持つ波形を設計したい。

整列位相ログステップマルチキャリア波の初期時刻を $T_0 = 0$ とすると、ドップラーシフトに強い波形が得られることが想像できる。送信波には、あるキャリアの周波数がドップラーシフトを受けて別の周波数に変化した状態のキャリアが、すべて足し合わせて準備されており、しかも時間原点においてすべてのキャリアの位相は一致しているからである。ところが $T_0 = 0$ として得られる波形は、時間原点に鋭いピークを持つため、送信波としては扱いづらい。我々は送信波にピークが現れないように $T_0$ を設定したい。

初期時刻 $T_0$ の絶対値を、半信号長 $T$ よりも大きく設定すると、時間原点にあった鋭いピークは窓関数の外に追いやられるため送信波には現れなくなる。あまり $T_0$ の絶対値を大きく設定すると、今度は各キャリアの位相がランダムになって扱いづらい波形が得られるが、程よい大きさの $T_0$ を選択して、$T_0 = 0$ のときの信号の性質をできるだけ保ったままで鋭いピークのない信号を設計することが可能である。

整列位相ログステップマルチキャリア波のドップラー解析を、$T_0 = 0$ と、$T_0 = 3000$ (us) に設定して行った場合について（図 3.11）に示す。これらは相関計算に実数を用いた場合である。相関計算に複素数を用いると、結果は少し違ったものになる（図 3.12）。この図を見ると、特に時間方向の濃淡に差異が生じていることが確認できる。複素数で計算した場合の図の濃淡には複素数絶対値を対応させているが、解析関数の複素数絶対値には時間方向の包絡線を検出する機能があるためである。それに対してドップ

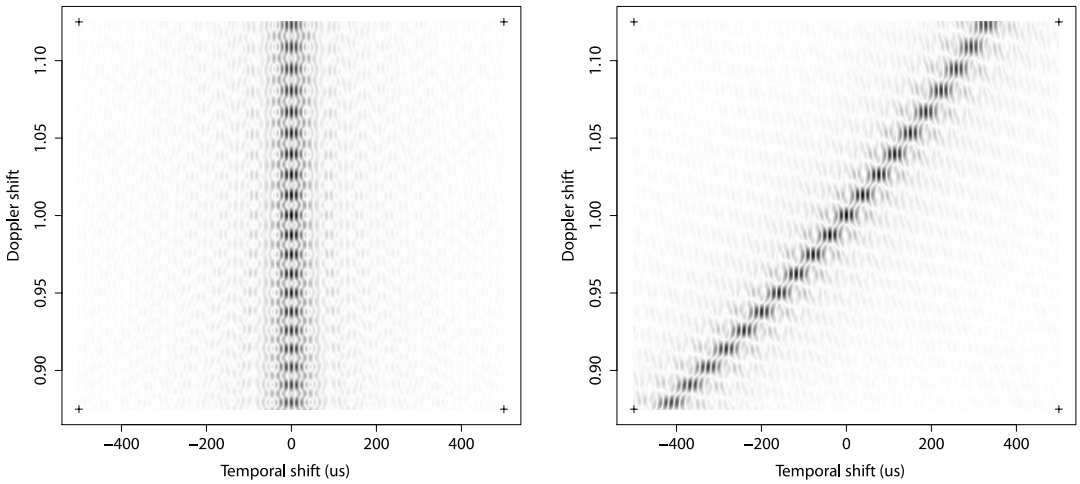


図 3.11　実数で計算した整列位相ログステップマルチキャリア波のドップラー解析 (左: $T_0 = 0$、右: $T_0 = 3000$ (us))

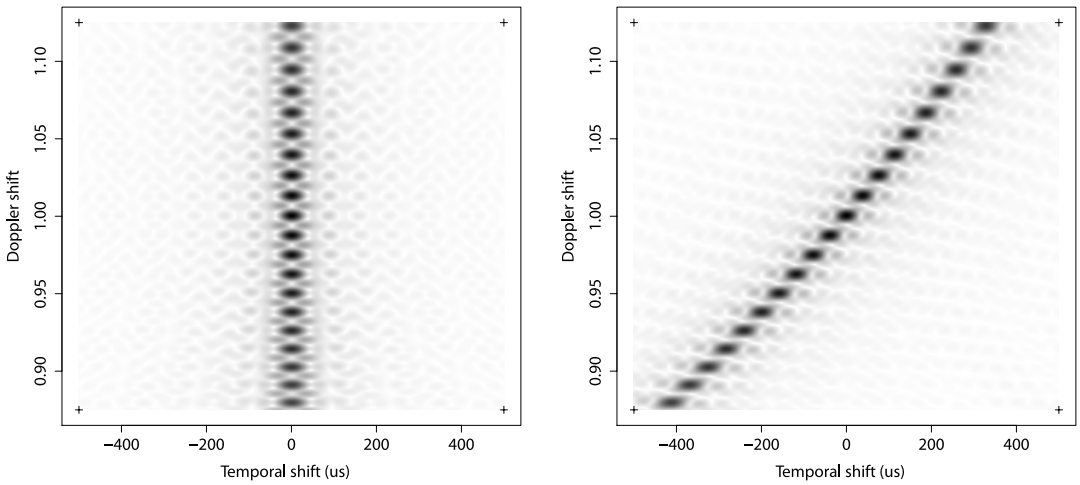


図 3.12　複素数で計算した整列位相ログステップマルチキャリア波のドップラー解析 (左: $T_0 = 0$、右: $T_0 = 3000$ (us))

ラー量方向の濃淡には実数計算の場合と複素数計算の場合で比較してあまり差異が見られない。これはドップラー量方向の変化が、信号の複素化とは別の原理に基づいていることを示している。その原理については後で詳しく検討する。

図 3.11 及び図 3.12 を初期時刻 $T_0$ の違いで比較すると、ドップラー量とピーク時刻の相関関係に典型的な違いがあることが確認できる。$T_0 = 0$ の場合にはピーク時刻は

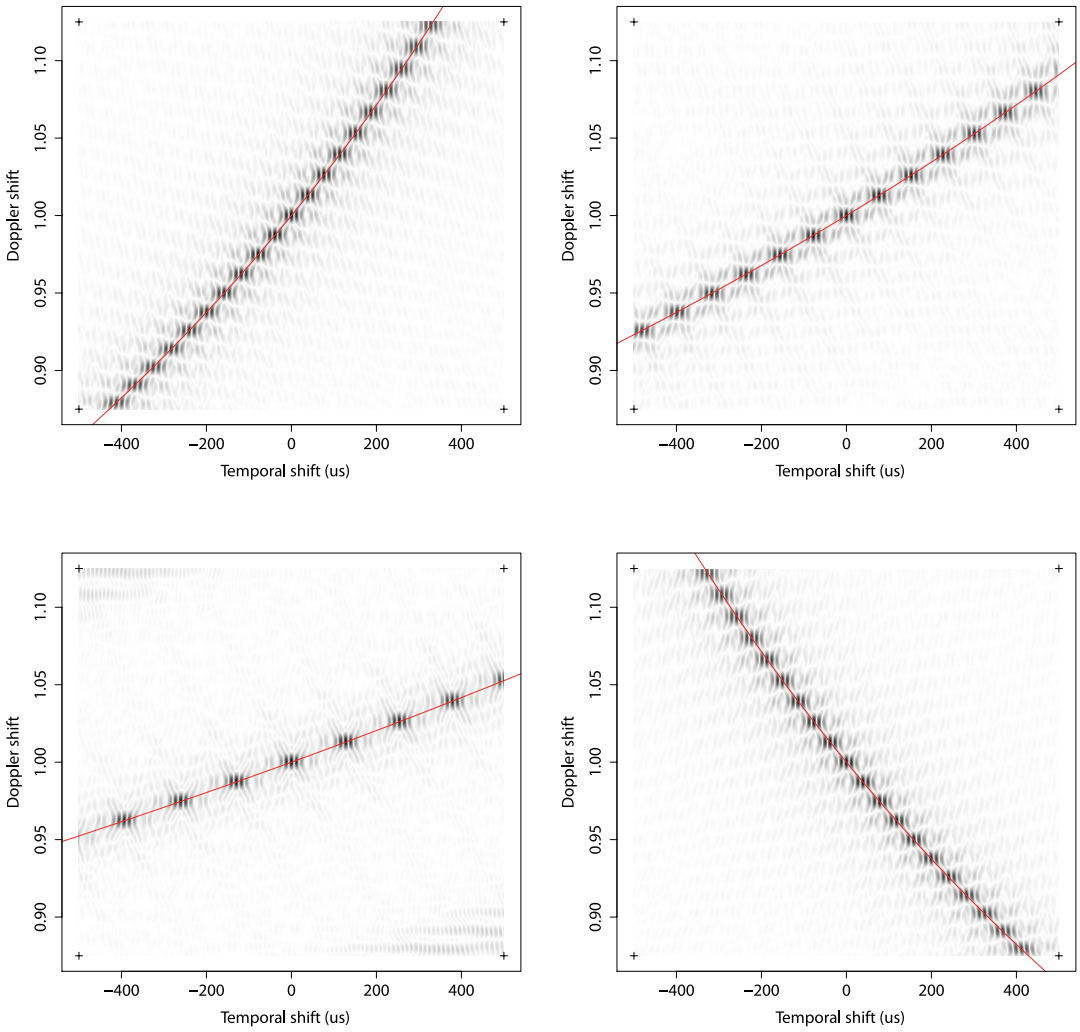


図 3.13　相互相関に現れたピークと系統偏移 (左上:$T_0 = 3000$、右上:$T_0 = 6000$、左上:$T_0 = 10000$、右下:$T_0 = -3000$)

ドップラー量に寄らず一定でありいつでも時間原点にピークが得られるが、$T_0 = 3000$ (us) の場合にはドップラー量に対してピーク時刻は正の相関を示している。つまりドップラー量が 1 よりも大きくなって受信波の周波数が高くなるとピーク時刻はより遅く現れるようになり、ドップラー量が 1 よりも小さくなって受信波の周波数が低くなるとピーク時刻はより早く現れるようになる。

図 6.2 に初期時刻 $T_0$ を様々に変えた場合のドップラー解析結果を示す。$T_0$ が大きくなるほどピーク時刻は右に傾き、いっぽう $T_0$ が負の値になるとピーク時刻は左に傾いていることが確認できる。ピークを与える点を結ぶと赤い曲線が得られるが、これらの赤い曲線はこれから示す計算式に基づいて算出される。

まずピーク時刻の概念を定式的に述べる。ドップラー解析におけるピーク時刻とは、$\rho$ をドップラー量、$g(t)$ を送信波としたときに次式で定義される時刻 $t_0$ のことである。

$$t_0 = \underset{-\infty<t<\infty}{\arg\max} \left\{ \langle g | \mathfrak{D}_\rho g \rangle (t) \right\} \tag{3.9}$$

送信波を整列位相ログステップマルチキャリア波 $g(t) = r_{[-T,T]}(t) g_{p,q}(t, T_0)$ とすると、ドップラー量を変化させたときのピーク時刻に関して次の定理が成り立つ。

**[定理] 1** ドップラー量を $\rho$ としたとき、整列位相ログステップマルチキャリア波 $g(t)$ の相互相関 $\langle g | \mathfrak{D}_\rho g \rangle (t)$ は、次の時刻 $t_0$ にピークを持つ。

$$t_0 = T_0 \left(1 - 1/\rho\right) \tag{3.10}$$

証明

まず送信波 $g(t)$ と受信波 $\mathfrak{D}_\rho g(t)$ の相互相関計算で、$t_0 = T_0(1 - 1/\rho)$ において周波数の一致する各キャリアの位相が一致することを示す。

送信波の $k$ 番目のキャリアは、

$$\begin{aligned} c_k(\tau) = \cos \omega = \cos \Big[ 2\pi p^{k/q} f_0 (\tau - t_0) + \phi_k \Big] \\ \text{但し } \phi_k = 2\pi p^{k/q} f_0 T_0 \end{aligned} \tag{3.11}$$

と表される。いっぽうドップラー量が $\rho = p^{\kappa/q}$ であるとき、受信波の $k - \kappa$ 番目のキャリア $c'_{k-\kappa}$ の周波数が $c_k$ の周波数に一致して、

$$\begin{aligned} c'_{k-\kappa}(\tau) = \sqrt{\rho} \cos \omega' = \sqrt{\rho} \cos \Big[ 2\pi p^{(k-\kappa)/q} f_0 \rho \tau + \phi_{k-\kappa} \Big] \\ \text{但し } \phi_{k-\kappa} = 2\pi p^{(k-\kappa)/q} f_0 T_0 \end{aligned} \tag{3.12}$$

と表される。従って問題は $\omega \equiv \omega'$ を示すことに帰着される。

ところが $t_0 = T_0(1 - 1/\rho)$、$\rho = p^{\kappa/q}$ であるから、これらを代入して計算すると、

$$\begin{aligned} \omega &= 2\pi p^{k/q} - 2\pi p^{k/q} f_0 T_0 (1 - 1/\rho) + 2\pi p^{k/q} f_0 T_0 \\ &= 2\pi p^{k/q} f_0 \tau - 2\pi p^{k/q} f_0 T_0 \left(1 - 1/p^{\kappa/q}\right) + 2\pi p^{k/q} f_0 T_0 \\ &= 2\pi p^{k/q} f_0 \tau - 2\pi p^{k/q} f_0 T_0 + 2\pi p^{(k-\kappa)/q} f_0 T_0 + 2\pi p^{k/q} f_0 T_0 \\ &= 2\pi p^{k/q} f_0 T_0 + 2\pi p^{(k-\kappa)/q} f_0 T_0 \end{aligned} \tag{3.13}$$

$$\begin{aligned} \omega' &= 2\pi p^{(k-\kappa)/q} f_0 \rho \tau + 2\pi p^{(k-\kappa)/q} f_0 T_0 \\ &= 2\pi p^{(k-\kappa)/q} f_0 p^{\kappa/q} \tau + 2\pi p^{(k-\kappa)/q} f_0 T_0 \\ &= 2\pi p^{k/q} f_0 \tau + 2\pi p^{(k-\kappa)/q} f_0 T_0 \end{aligned} \tag{3.14}$$

従って $\omega = \omega'$ となり、時刻 $t_0$ においてキャリア $c_k(\tau)$ と キャリア $c'_{k-\kappa}(\tau)$ の位相が一致することが示された。

次に位相一致点における相関値の極大性を示す。すなわち、$\rho = p^{k/q}$ とし、$\rho'$ を $\rho$ の近傍に属する点とし、$t_0$ を位相一致時間とすると、$t_0$ の近傍に属する時間 $t$ に対し、近似的に、

$$|\langle g|\mathfrak{D}_{\rho'}\, g\rangle\,(t)| \leq |\langle g|\mathfrak{D}_{\rho} g\rangle\,(t_0)| \tag{3.15}$$

が成立することを示す。但し記法の簡潔のために、代表例として $\rho = 1$ の場合について近似不等式が成立する事を示す。

$c_k(t)$ をログステップマルチキャリア波の各キャリアとすれば、定義によって、$\rho = 1$ の近傍及び $t = t_0$ の近傍で

$$\begin{aligned}
|\langle g|\mathfrak{D}_{\rho}\, g\rangle\,(t)| &= \left|\sqrt{\rho}\int_{-\infty}^{\infty}\left(\sum_{k=0}^{q-1}\overline{c_k}(\tau - t)\right)\left(\sum_{m=0}^{q-1} c_k(\rho\tau)\right)\, d\tau\right| \\
&\simeq \left|\sqrt{\rho}\sum_{k=0}^{q-1}\int_{-\infty}^{\infty}\overline{c_k}(\tau - t)\, c_k(\rho\tau)\, d\tau\right| \\
&\leq \sqrt{\rho}\sum_{k=0}^{q-1}\int_{-\infty}^{\infty}|\overline{c_k}(\tau - t)\, c_k(\rho\tau)|\, d\tau \\
&\simeq \sqrt{\rho}\sum_{k=0}^{q-1}\int_{-\infty}^{\infty}|\overline{c_k}(\tau - t_0)\, c_k(\rho\tau)|\, d\tau
\end{aligned} \tag{3.16}$$

となるためである。なお等号は $\rho = 1$ 及び $t = t_0$ において成立する。

その他のピーク $\rho = p^{k/q}, k \neq 0$ の場合も同様の近似不等式が成立する。ここで次のことに注意しておくべきである。すなわち、周波数の近似するキャリア以外のすべてのキャリア間の干渉が全体の相互相関に影響を与えるため、この不等式は近似不等式となるということである。

□

証明中に示されたように、整列位相ログステップマルチキャリア波の到来時刻系統偏移の発生は、送信波と受信波でキャリア位相の一致する時刻がドップラーシフトに従って変化し、その時刻で周波数の一致するすべてのキャリアが同時に同じ位相を持つことに由来している。パラメーターを調整して送信波としてピークのない適切なものを構成すると、この系統偏移が必ず現れ、それがあらかじめ予測可能な理論式に基づいているということが本節における重要な結論である。

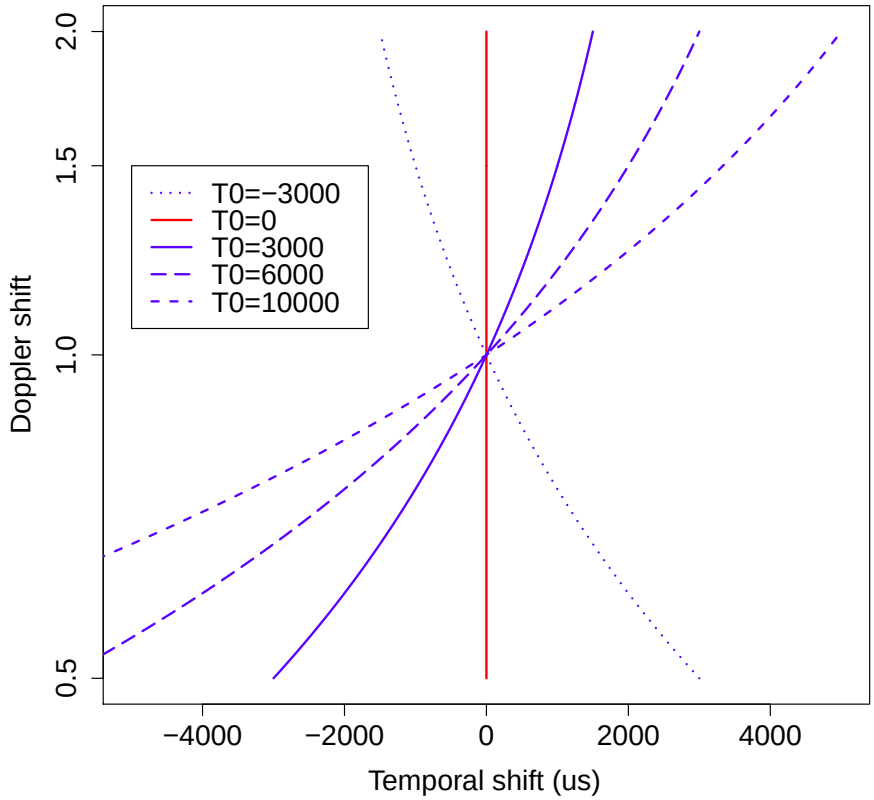


図 3.14　$T_0$ の変化による系統偏移の変化

## 3.6　相互相関プロファイル

整列位相ログステップマルチキャリア波のドップラー解析の結果、ピーク時刻に系統偏移が現れることはすでに述べた。ところがこれは、相互相関の最大値を与える時刻についての考察であって、相互相関の最大値そのものがドップラーシフトに従ってどのように変化するのかという疑問には答えていない。本節では、相互相関プロファイルの概念を導入し、ドップラーシフトが作用したときの相互相関の最大値の振る舞いについての解答を与える。

相互相関プロファイルは本研究で導入した概念である。相互相関プロファイルは、言わばドップラー解析図の背骨構造であり、ドップラーシフトごとに得られる相互相関ピークを結んだ曲線に沿って相関値をプロットしたものである。

形式的に述べると、波形 $g(t)$ の相互相関プロファイルとは、次式により定義される

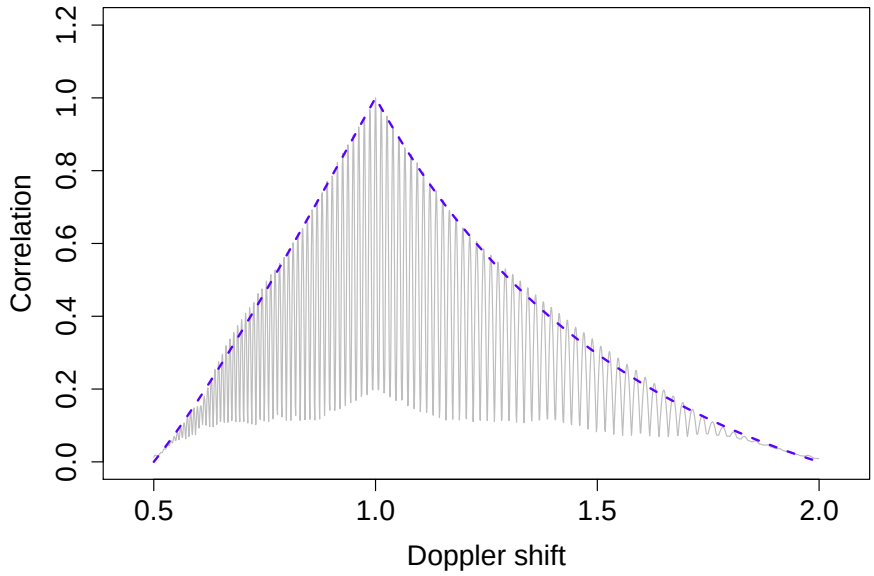


図 3.15　プロファイル概形構造と微細構造

関数 $\lambda_g(\rho)$ である。

$$\lambda_g(\rho) = \max_{-\infty<t<\infty} \{\|\langle g|\mathfrak{D}_\rho g\rangle(t)\|\} \tag{3.17}$$

本節ではこの相互相関プロファイルの構造について、概略構造と微細構造の 2 つの観点から議論する（図 3.15）。

プロファイル概略構造は、プロファイル包絡線によりほぼその形が決定する。プロファイル包絡線とは、プロファイルの極大値を結んで得られる曲線であり、極大値においては値が一致し、それ以外の点でより大きな値をとる関数 $\Lambda_g(\rho)$ として定義される。すなわち、$\Lambda_g(\rho)$ は次式を満たす。

$$\lambda_g(\rho) \leq \Lambda_g(\rho) \text{ for any } \rho \tag{3.18}$$

このような関数 $\Lambda_g(\rho)$ は一意ではなく、また本論文の記載では偶成的に生じた一部のピークで不等号の向きが逆転することもあり得るが、より扱いやすく簡単な包絡線関数を発見してその性質を調べる事は大切な課題である。

プロファイル微細構造は、極大値の近傍でのプロファイルのふるまいを近似する数式を与えることによって考察する。すなわち極大値を与える特定のドップラー量 $\rho_1$ について、小さな数 $\epsilon$ に対し次式を満たす関数 $\mu_g(\rho)$ を発見することが課題である。

$$\mu_g(\rho) \simeq \lambda_g(\rho) \text{ for any } \rho \text{ s. t. } |\rho_1 - \rho| < \epsilon \tag{3.19}$$

表 3.1 整列位相ログステップマルチキャリア波の相互相関プロファイル比較図の設定

| **図番号** | **種別** | **初期時刻** $T_0$ | **帯域幅** $p$ | **キャリア数** $q$ |
|---|---|---|---|---|
| 図 3.16 | 比較図 A | 0 | 1.5, 1.75, 2.0, 2.5, 3.0 | 54 |
| 図 3.17 | 比較図 B | 0 | 2.0 | 12, 24, 48, 54, 72 |
| 図 3.18 | 比較図 A | 3000 | 1.5, 1.75, 2.0, 2.5, 3.0 | 54 |
| 図 3.19 | 比較図 B | 3000 | 2.0 | 12, 24, 48, 54, 72 |
| 図 3.20 | 比較図 A | 6000 | 1.5, 1.75, 2.0, 2.5, 3.0 | 54 |
| 図 3.21 | 比較図 B | 6000 | 2.0 | 12, 24, 48, 54, 72 |
| 図 3.22 | 比較図 A | 10000 | 1.5, 1.75, 2.0, 2.5, 3.0 | 54 |
| 図 3.23 | 比較図 B | 10000 | 2.0 | 12, 24, 48, 54, 72 |

対象がマルチキャリア波の場合には、$\lambda_g(\rho)$ はすべてのキャリア間干渉を計算することによって厳密式を与える事ができる。しかしその方法だけでは数値計算によってグラフの形を決める以外の方法では、あらかじめ $\lambda_g(\rho)$ の形を知ることができず不便である。本論文ではキャリア間干渉の計算を、隣接するキャリア間に限定することによってより簡単な $\mu_g(\rho)$ の形を決定する。

具体的な議論に入る前に、数値計算によって得られた整列位相ログステップマルチキャリア波の相互相関プロファイルを観察しておくことにする。

基本周波数 $f_0$ と信号長 $2T$ を固定して考えると、整列位相ログステップマルチキャリア波は帯域幅 $p$ と、キャリア数 $q$ と、初期時刻 $T_0$ によって決定される。そのためプロファイルの形を考察する上で、これらのパラメーターとの関係を調べる事が重要である。方針として、初期時刻 $T_0 = 0, 3000, 6000, 10000$ (us) の順に次の 2 種類の比較図を作成した。比較図 A は、キャリア数を $q = 54$ に固定して、帯域幅を $p = 1.5, 1.75, 2, 2.5, 3.0$ と変化させたものである。比較図 B は、帯域幅を $p = 2.0$ に固定して、キャリア数を $q = 12, 24, 48, 54, 72$ と順に変化させたものである。

図 3.16、図 3.17、図 3.18、図 3.19、図 3.20、図 3.21、図 3.22、図 3.23 に比較結果を示す。表 3.1 は、各比較図の設定をまとめたものである。

これらの図から、プロファイルの概形構造には、初期時刻 $T_0$ と帯域幅 $p$ が複合的に関係し、$T_0$ が大きくなると概形構造は鋭くなり、帯域幅 $p$ が増加すると概形構造は緩やかになることが確認できる。またキャリア数 $q$ はプロファイルの櫛形の櫛の粗さに関係し、プロファイル概形構造にはあまり関与していないことが観察できる。

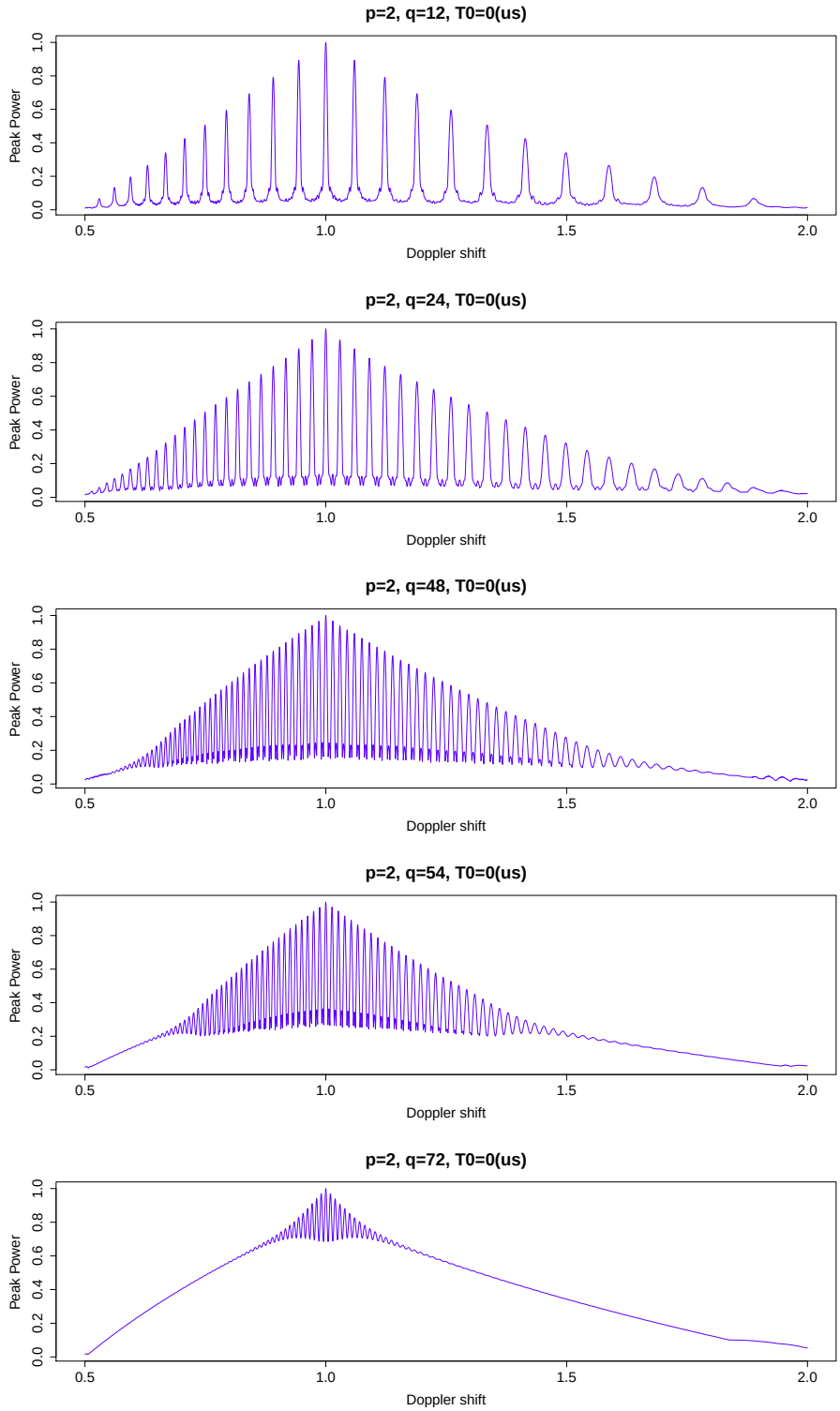


図 3.16　ログステップマルチキャリア波のキャリア数によるプロファイルの変化 ($T_0 = 0$)

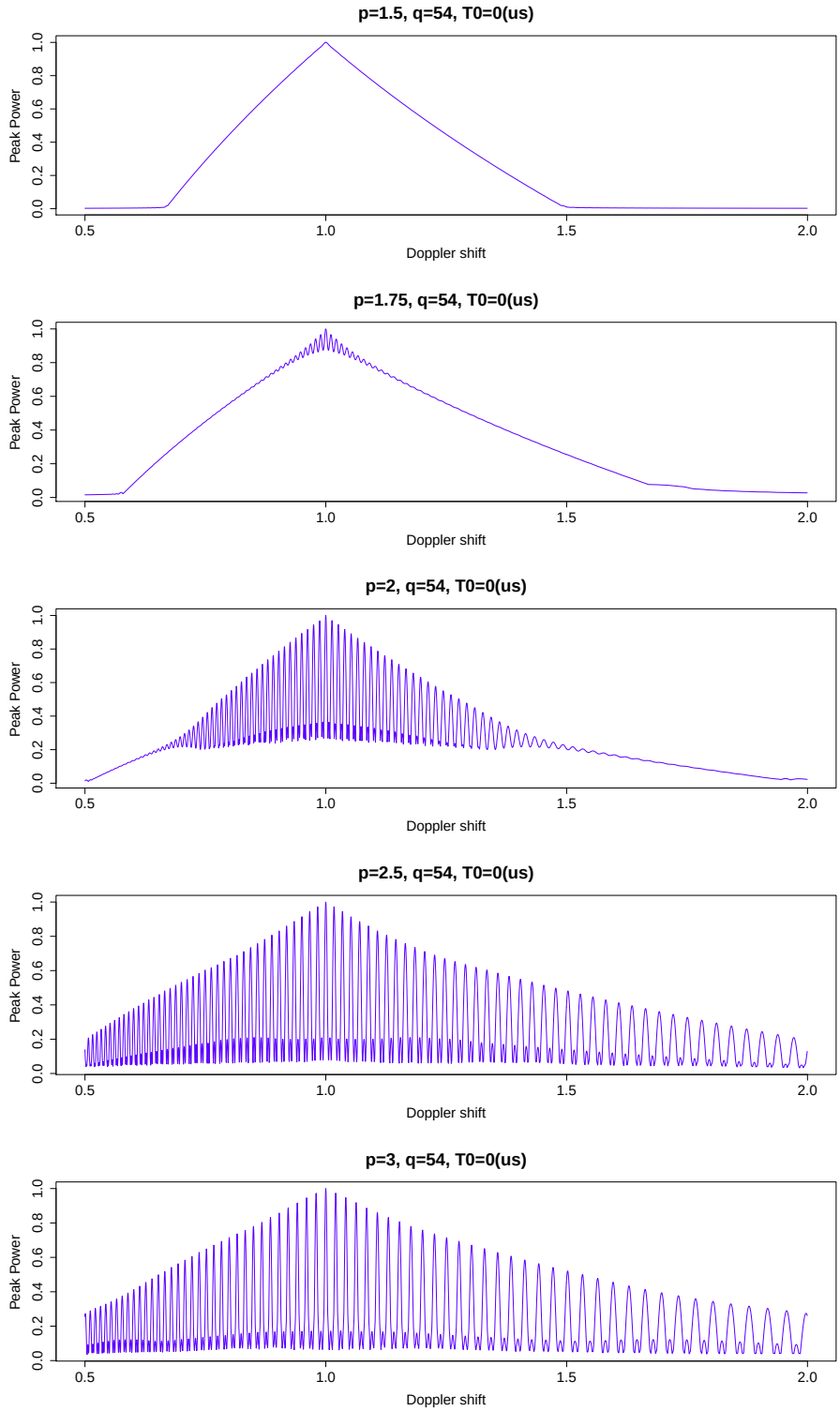


図 3.17　ログステップマルチキャリア波の帯域幅によるプロファイルの変化 ($T_0 = 0$)

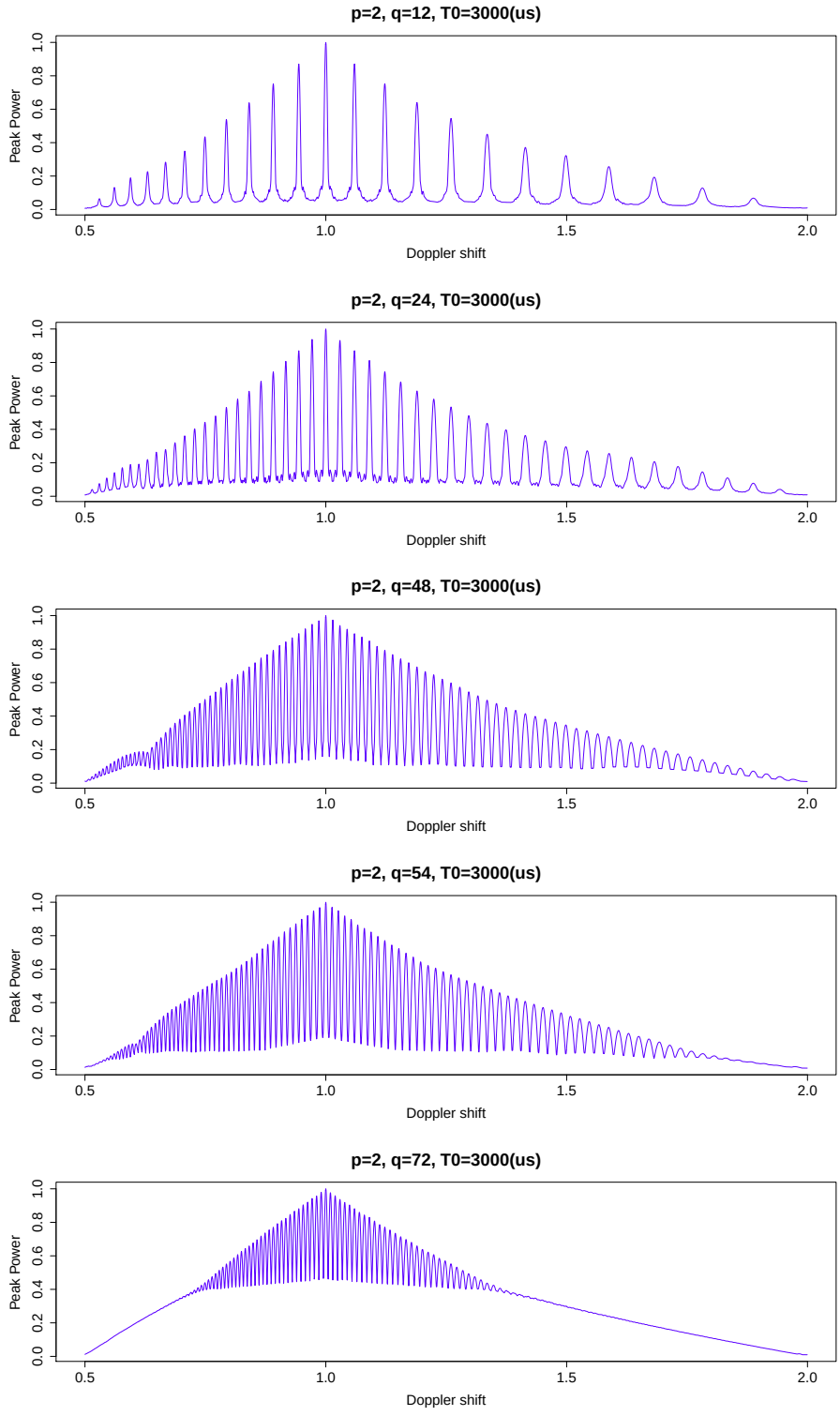


図 3.18　ログステップマルチキャリア波のキャリア数によるプロファイルの変化 ($T_0 = 3000$)

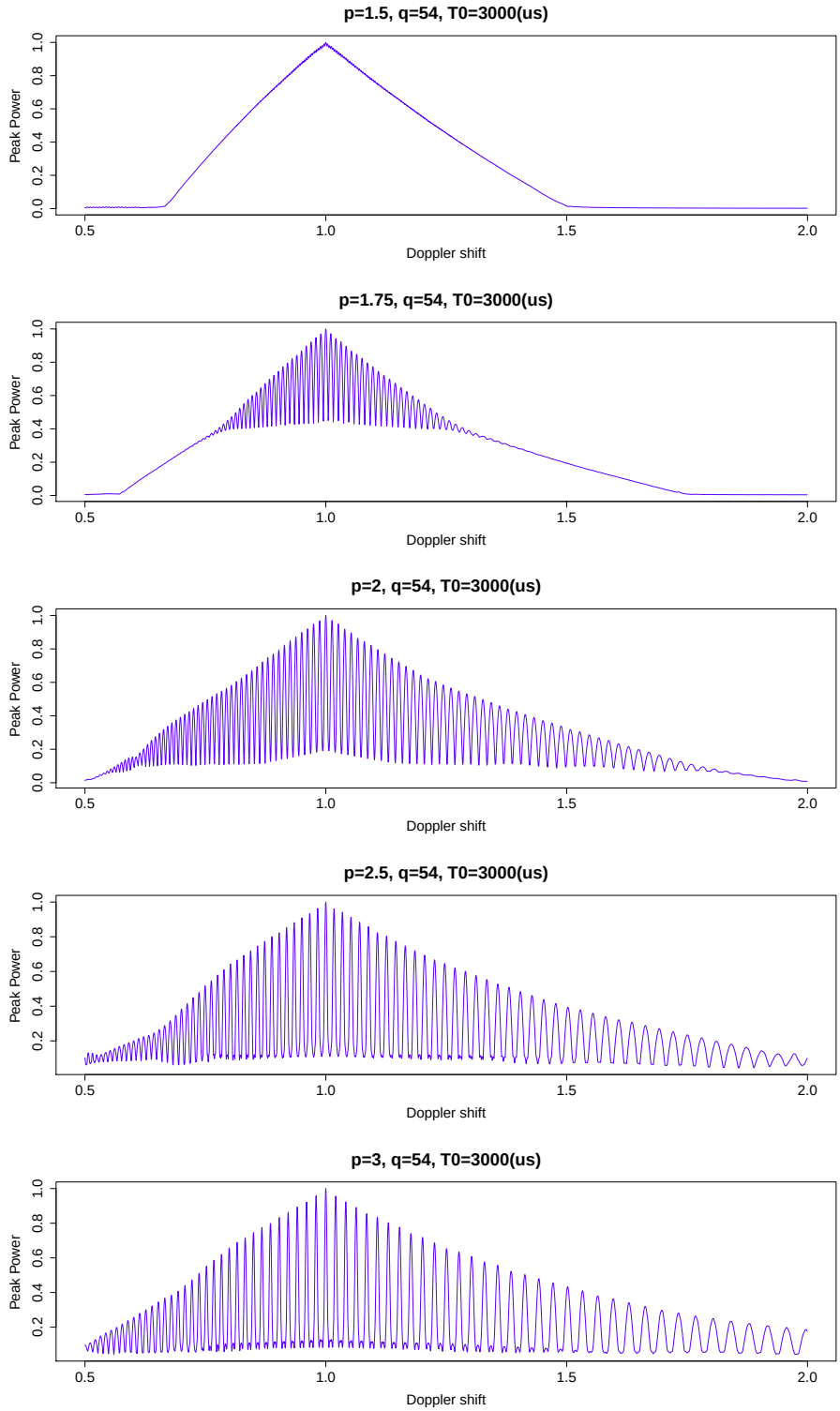


図 3.19　ログステップマルチキャリア波の帯域幅によるプロファイルの変化 ($T_0 = 3000$)

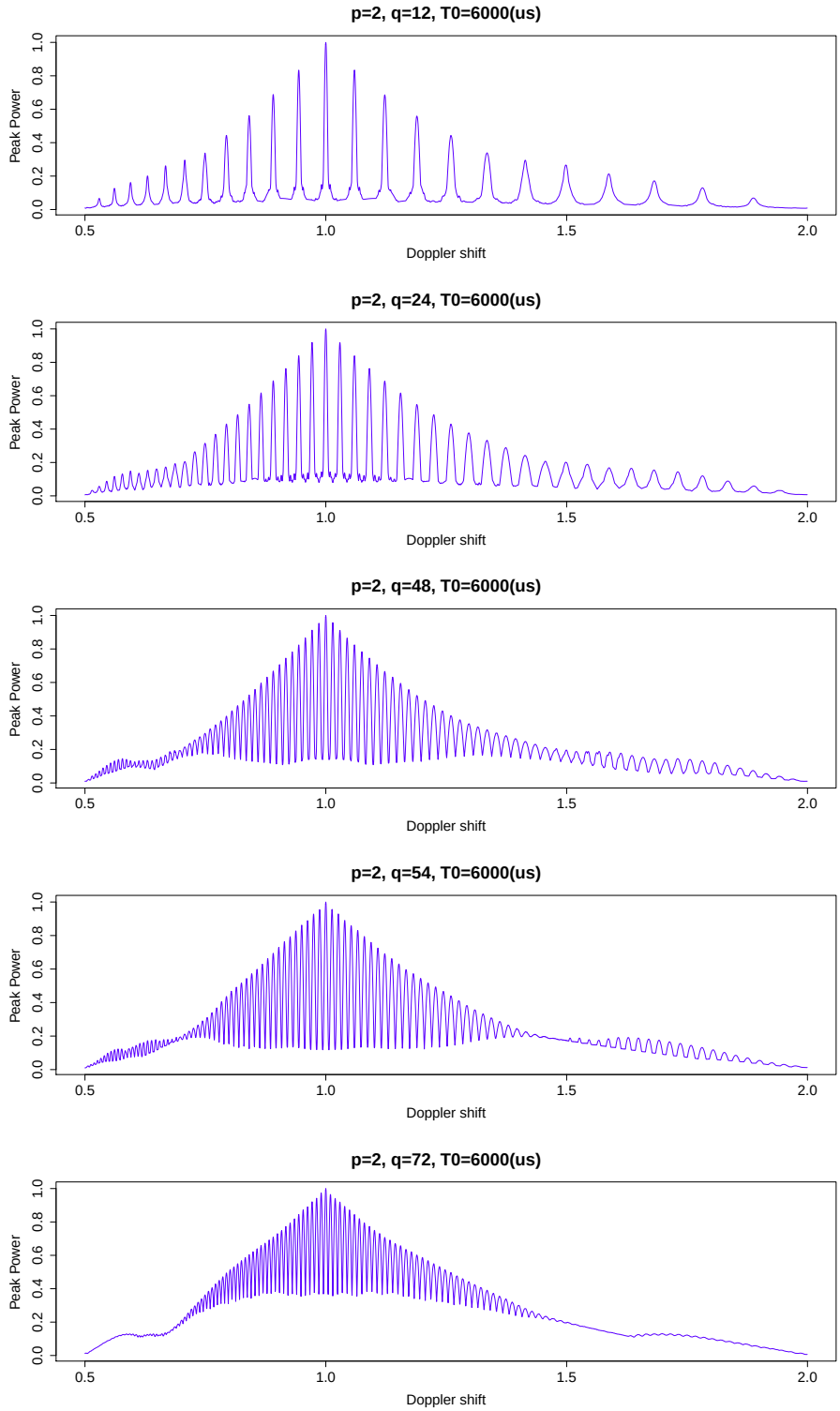


図 3.20　ログステップマルチキャリア波のキャリア数によるプロファイルの変化 ($T_0 = 6000$)

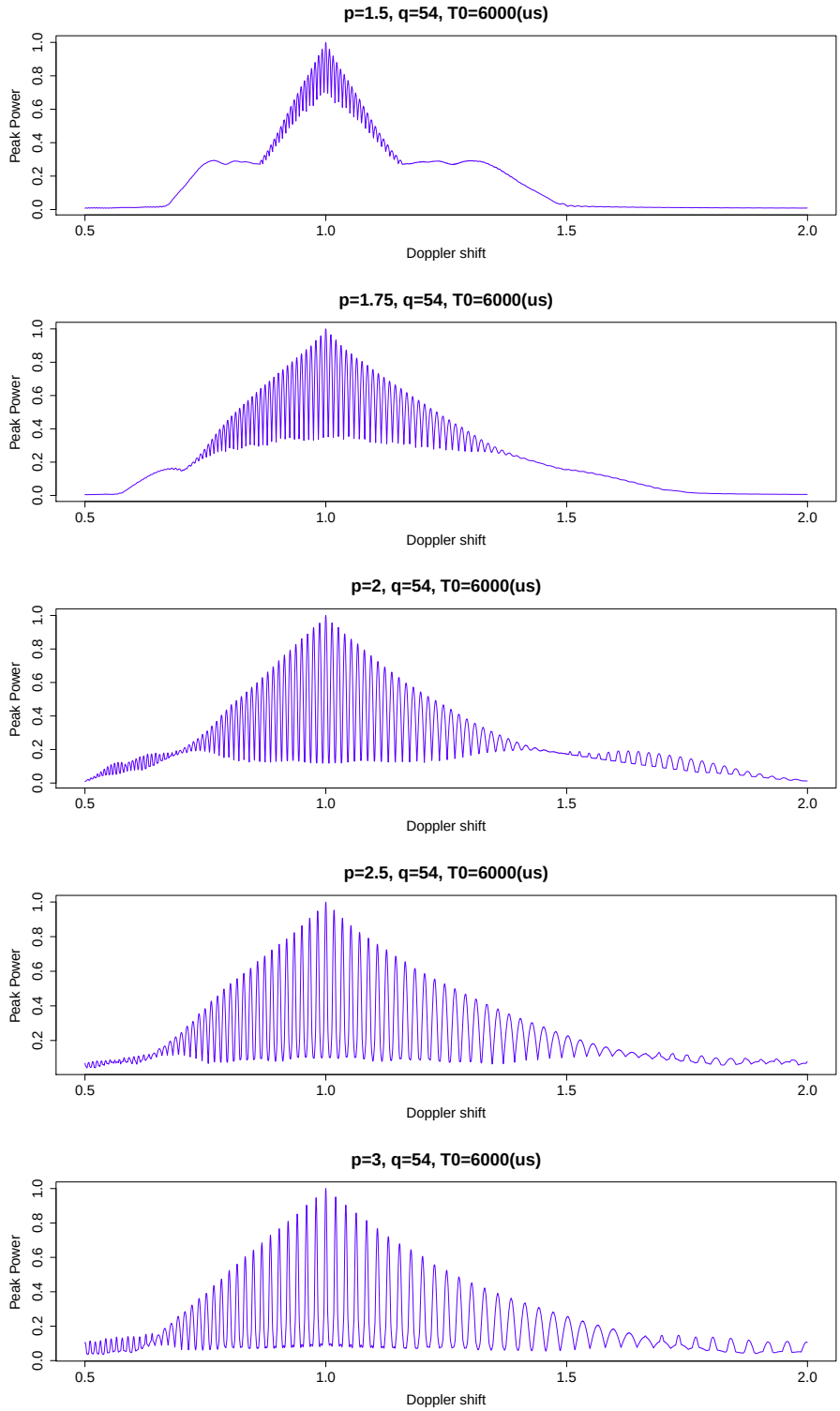


図 3.21　ログステップマルチキャリア波の帯域幅によるプロファイルの変化 ($T_0 = 6000$)

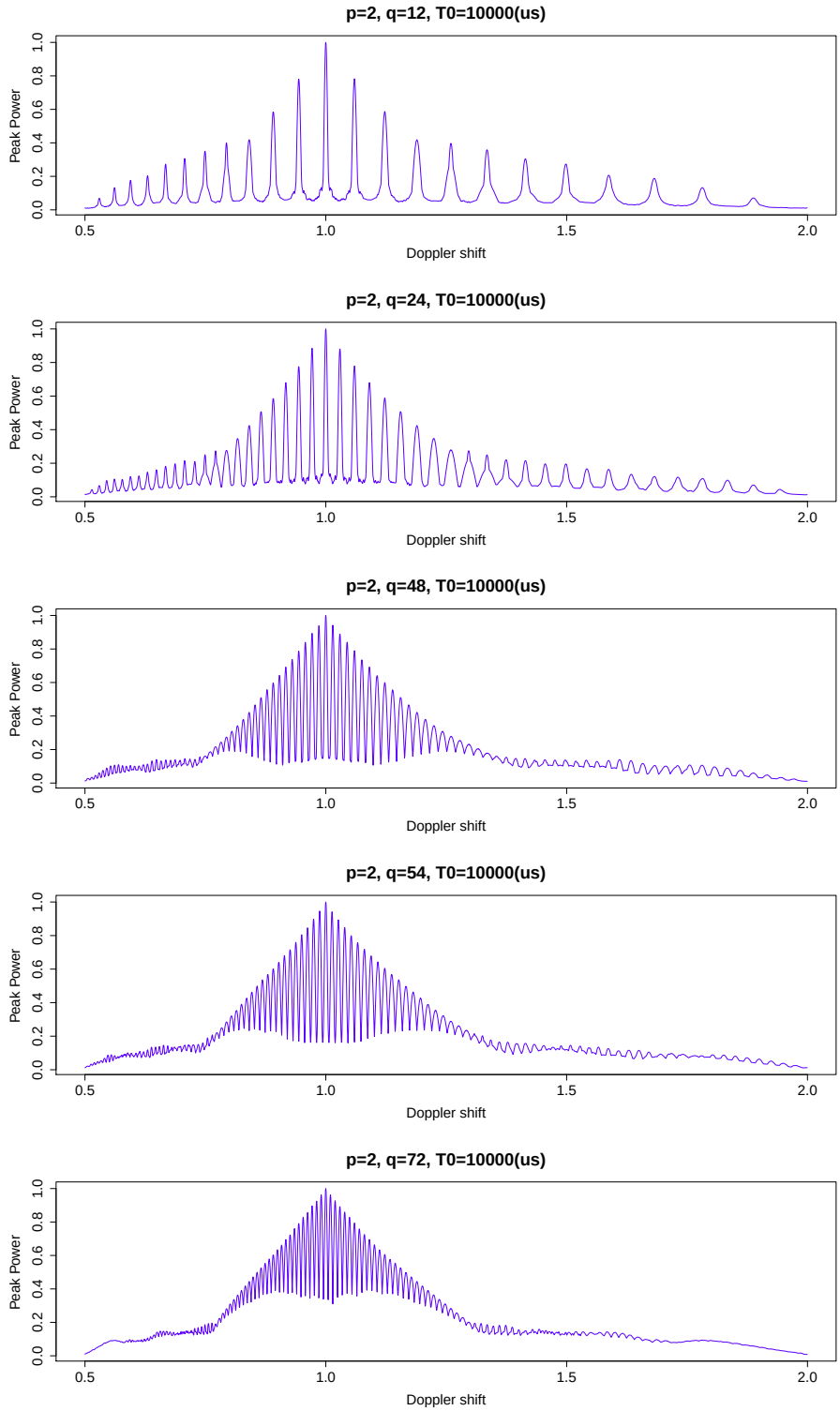


図 3.22　ログステップマルチキャリア波のキャリア数によるプロファイルの変化 ($T_0 = 10000$)

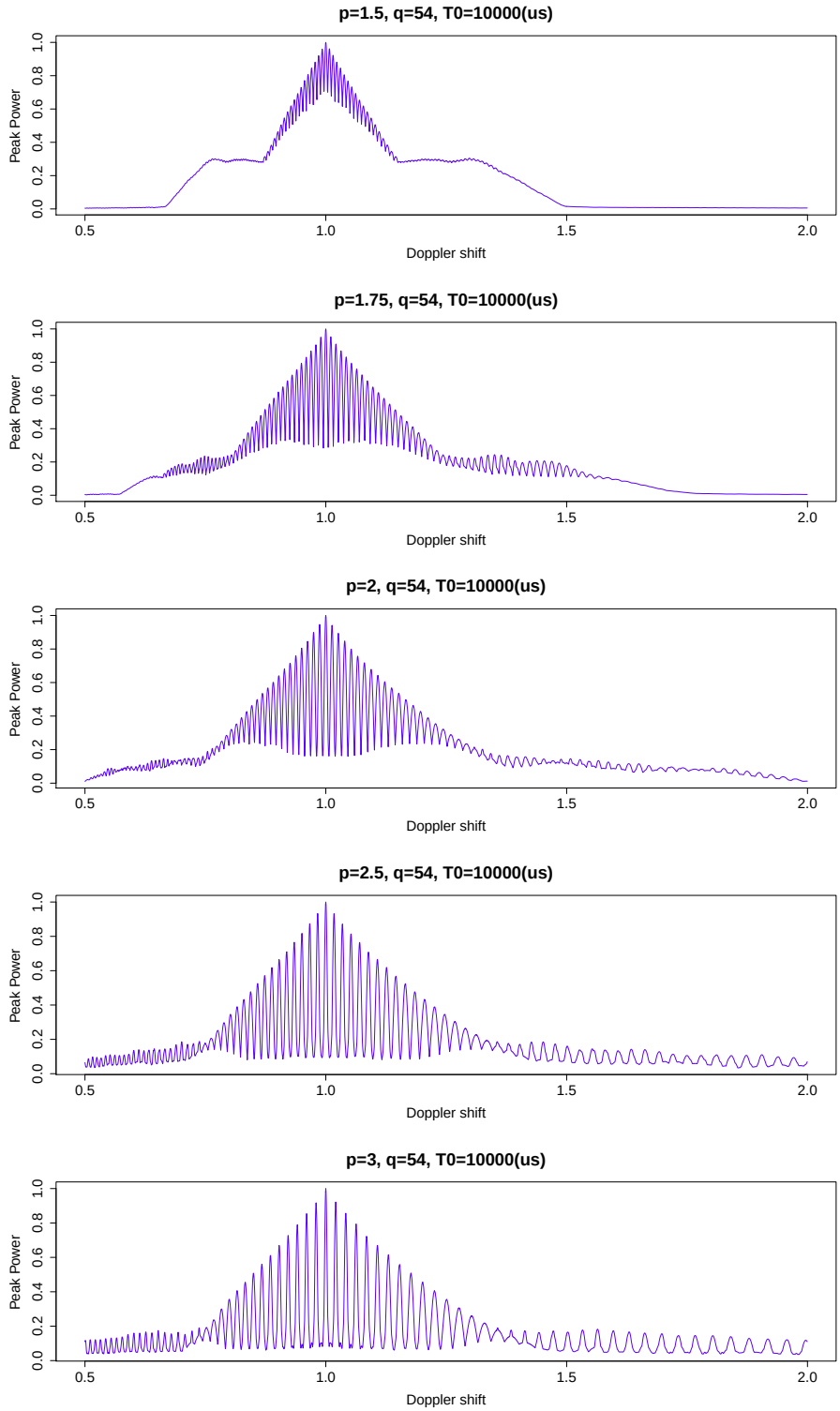


図 3.23　ログステップマルチキャリア波の帯域幅によるプロファイルの変化 ($T_0 = 10000$)

## 3.7　プロファイル概形構造

本節ではプロファイル概形構造についての考察を行う。

一般の波形に対してそのプロファイル包絡線を与えることは難しい課題である。しかし、整列位相ログステップマルチキャリア波に関しては前述の [定理 1] により、ピークを与える時刻があらかじめ分かっているため具体的な式を与える事が可能である。

マルチキャリア波の相互相関値の計算では、送信波を構成する複数のキャリアすべてについて、受信波を構成するすべてのキャリアとの干渉を計算する。周波数が異なる場合のキャリア間の干渉の計算にはフーリエ変換による解析を必要とするが、周波数が一致するキャリア間の干渉は単に信号区間の重複の問題に帰着するので計算が簡単になる。

整列位相ログステップマルチキャリア波の場合、プロファイルの極大値はキャリア周波数が一致する点において与えられるので、極大値の計算は、周波数の一致するキャリア数と、信号重複区間の積の計算によって行うことができる。これが本節で述べる結果をもたらす重要なアイデアである。

プロファイル包絡線は、極大値を結んで得られる曲線であるから、離散的に現れる極大値を連続的に表現する方法を見いだしてこれを定式化することにより概形構造の記述を行う。既に観察したように、初期時刻 $T_0$ の大小がプロファイル包絡線の概形に関して大きな影響がある。一般的に $|T_0|$ が小さな方がプロファイル包絡線はなだらかであり、$|T_0|$ を大きくするとプロファイル包絡線は急峻になる。そればかりか、半信号長を $T$ として $|T_0| \leq T$ の場合と $|T_0| > T$ の場合で送信波と受信波で重複区間の現れ方が異なるため、記述式は別々のものになる。

まず $|T_0| \leq T$ の場合について検討する。ドップラーシフトによる信号長の伸張と短縮は、各キャリアの位相が 0 に揃う点（ゼロ位相点）を中心に発生するので、この場合は $|T_0| \leq T$ の条件によりゼロ位相点が信号区間内に含まれているということになる。そのため、送信波と受信波と間の重複区間は、どちらか短い方の信号区間と一致するため、重複区間についての計算は簡単なものになる。

$|T_0| \leq T$ の場合には、整列位相ログステップマルチキャリア波のプロファイル包絡線に関して、次の定理が成り立つ。

**［定理］ 2** 整列位相ログステップマルチキャリア波 $g(t) = r_{[-T,T]}(t)\, g_{p,q}(t, T_0)$ は、$|T_0| \leq T$ の場合に、プロファイル $\lambda_g(\rho)$ が次の関数 $\Lambda_g^0(\rho)$ により近似的に包絡さ

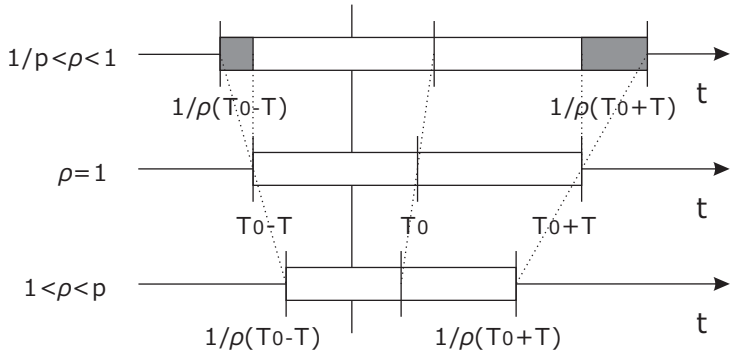


図 3.24　ドップラーシフトによる重複区間の変化 ($|T_0| \leq T$)

れる。

$$\Lambda_g^0(\rho) = \frac{2T}{\sqrt{\rho}} \left\{ q + q\frac{\log \rho}{\log p} \right\} (\text{for } 1/p \leq \rho \leq 1)$$
$$\Lambda_g^0(\rho) = 2\rho T \left\{ q - q\frac{\log \rho}{\log p} \right\} (\text{for } 1 \leq \rho \leq p)$$
$$\Lambda_g^0(\rho) = 0 \ (\text{otherwise}) \tag{3.20}$$

証明

まず、プロファイルのピークは離散的に現れることに注意する。すなわち $k$ を整数として、ドップラー量 $\rho = p^{k/q}$ において、相互相関値は局所的なピークを持つ。これらのピークにおいて、プロファイルの値が与式の値を持つことを示す。

周波数の異なるキャリア間の相関は 0 であり無視できるとする。すると相互相関値は、周波数の一致するキャリア数と送受信波で位相の一致する重複区間の積により決定される。

まず周波数の一致するキャリア数について考察する。ドップラーシフトがない場合には、送受信波のすべてのキャリアの周波数が一致し、その数は $q$ である。ドップラー効果が密の方向に作用する場合 ($\rho \geq 1$) は、$\rho = 1$ のとき一致するキャリア数は $q$ 個であり、$\rho = p$ のとき周波数の一致するキャリア数が 0 となり、$1 \leq \rho < p$ の間は一律に減少する。キャリア数はシフト量の対数となるため、一致するキャリアの数は、

$$N = q - q\,(\log \rho / \log p) \tag{3.21}$$

で与えられる。但し $\rho \geq p$ のとき $N = 0$ となる。

一方ドップラー効果が疎の方向に作用する場合 ($\rho < 1$) には、同様の考察により、一致するキャリアの数は、

$$N = q + q\,(\log \rho / \log p) \tag{3.22}$$

となる。但し $\rho \leq 1/p$ のとき $N = 0$ となる。

次に送受信波で位相の一致する重複区間について考察する。まず整列位相ログステップマルチキャリア波において、$t = -T_0$ を中心に受信波の位相が伸縮することを鑑みて、エポックを $-T_0$ だけずらして考えることにする（図 3.24)。

ドップラー量が $\rho \geq 1$ の場合と $\rho \leq 1$ の場合に場合分けする。$\rho \geq 1$ の 場合には、位相一致区間の長さは、受信波の信号長と等しく $L = 2T/\rho$ となる。$\rho \leq 1$ の場合には、受信波の信号長は長くなるが、位相一致区間の長さは、送信波の信号長と等しく $L = 2T$ となる。

最後に相互相関値に受信波の係数 $\sqrt{\rho}$ が乗ぜられ、最終的に相互相関値 $\sqrt{\rho}\,NL$ が得られる。

□

いっぽう $|T_0| > T$ の場合には、ゼロ位相点が信号内に含まれないため、ドップラーシフトによる受信波信号長の伸縮の結果、受信波の信号区間は送信波の信号区間の左右どちらか片方にのみはみ出すことになる。その結果、重複区間の長さは $|T_0| \leq T$ の場合に比べて短くなり、重複区間の長さに関する計算式も多少複雑なものになる。$|T_0| > T$ の場合には、整列位相ログステップマルチキャリア波のプロファイル包絡に関して、次の定理が成り立つ。

**［定理］ 3** 整列位相ログステップマルチキャリア波 $g(t) = r_{[-T,T]}(t)\, g_{p,q}(t, T_0)$ は、$|T_0| > T$ の場合に、プロファイル $\lambda_g(\rho)$ が次の関数 $\Lambda_g^1(\rho)$ により近似的に包絡される。

$$\Lambda_g^1(\rho) = \sqrt{\rho}\left\{\frac{2T}{\rho} - \left(\frac{1}{\rho} - 1\right)(|T_0| + T)\right\}\left\{q + q\frac{\log\rho}{\log p}\right\}\left(\text{for } \frac{1}{p}, \frac{|T_0| - T}{|T_0| + T} \leq \rho \leq 1\right)$$

$$\Lambda_g^1(\rho) = \sqrt{\rho}\left\{\frac{2T}{\rho} - \left(1 - \frac{1}{\rho}\right)(|T_0| - T)\right\}\left\{q - q\frac{\log\rho}{\log p}\right\}\left(\text{for } 1 \leq \rho \leq p, \frac{|T_0| + T}{|T_0| - T}\right)$$

$$\Lambda_g^1(\rho) = 0 \ \text{(otherwise)} \tag{3.23}$$

証明

まず、プロファイルのピークは離散的に現れることに注意する。$k$ を整数として、ドップラー量 $\rho = p^{k/q}$ において、相互相関値は局所的なピークを持つ。このとき、プロファイルの値が定理 2 に示された式の値を持つことを示す。

周波数の異なるキャリア間の相関は 0 であり無視できるとする。従って、相互相関値は、周波数の一致するキャリア数と送受信波で位相の一致する重複区間の積により決定される。

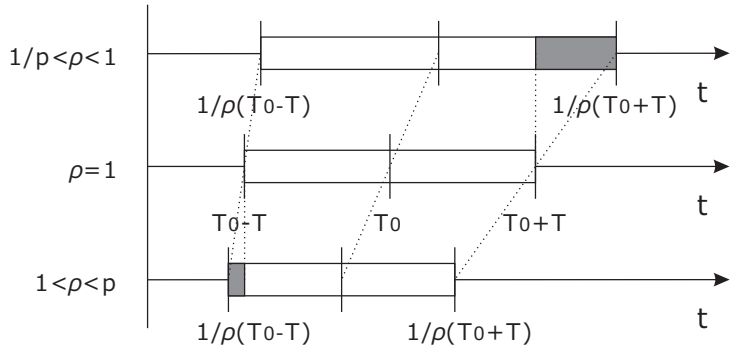


図 3.25　ドップラーシフトによる重複区間の変化 ($|T_0| > T$)

まず周波数の一致するキャリア数について考察する。ドップラーシフトがない場合には、送受信波のすべてのキャリアの周波数が一致し、その数は $q$ である。ドップラー効果が密の方向に作用する場合 ($\rho \geq 1$) 、$\rho = 1$ のとき一致するキャリア数は $q$ 個であり、$\rho = p$ のとき周波数の一致するキャリア数が 0 となるが、$1 \leq \rho < p$ の間は一律に減少する。キャリア間隔は指数的に決定されているために、キャリア数はシフト量の対数となり、一致するキャリアの数は、

$$N = q - q\,(\log \rho - \log p) \tag{3.24}$$

と表現される。いっぽうドップラー効果が疎の方向に作用する場合 ($\rho < 1$) には、$\rho = 1/p$ のとき、周波数の一致するキャリア数が 0 となり、$\rho = 1$ で $q$ 個となり、$1/p < \rho \leq 1$ の間は一律に増加する。同様の考察により、一致するキャリアの数は、

$$N = q + q\,(\log \rho - \log p) \tag{3.25}$$

となる。

次に送受信波で位相の一致する重複区間について考察する。やはりドップラー量が $\rho \geq 1$ の場合と $\rho \leq 1$ の場合に場合分けして考える。ドップラーシフトがない場合 ($\rho = 1$) には、波形前端が $T_0 - T$、後端が $T0 + T$ で表され、ドップラー効果がある場合には波形前端は $(1/\rho)(T_0 - T)$ であり、後端は $(1/\rho)(T_0 + T)$ と表される。

ドップラー効果が密の方向に作用する ($\rho \geq 1$) 場合には、受信波の波形長は短くなり位相一致区間は $t = 0$ の方向に向けて移動する。送信波の前端と後端を考慮すると、位相一致区間の長さ $L$ は

$$L = 2T/\rho - (1 - 1/\rho)\,(T_0 - T) \tag{3.26}$$

となる。但し $\rho \geq p$ のとき $L = 0$ になる。

ドップラー効果が疎の方向に作用する $(\rho \leq 1)$ 場合には、受信波の波形長は長くなり位相一致区間は $t = 0$ の方向に向けて移動する。やはり送信波の前端と後端との関係により、長さ $L$ は、

$$L = 2T/\rho - (1/\rho - 1)(T_0 + T) \tag{3.27}$$

となる。但し $\rho \leq 1/p$ のとき $L = 0$ になる。

最後にエネルギー保存則の要請により、相互相関値には $\sqrt{\rho}$ が乗ぜられる。従って、最終的に相互相関値 $\sqrt{\rho} \cdot N \cdot L$ が得られる。

□

なおここでの計算は、相関計算に複素数を用いた場合である。相関計算に実数を用いた場合、値は複素数による場合の $1/2$ となる。

ここで示したプロファイル包絡線関数 $\Lambda_g^1(\rho)$ で計算したグラフを、数値計算により求めたプロファイルと比較した結果を図 3.26 に示す。プロファイルの山型の部分において、うまく包絡線を近似していることが確認できる。

プロファイル包絡線関数は、数値計算に依らない方法で包絡線の形を求める事ができるため、送信波に用いる整列位相ログステップマルチキャリア波のパラメーター設計に役立てることができる。特に、帯域幅と信号長が固定された状態で、大きなドップラー量に対しても失われにくい相関値を得るためには、初期時刻 $T_0$ の大きさを出来るだけ小さくするという重要な指針がプロファイル包絡線関数から得られる。

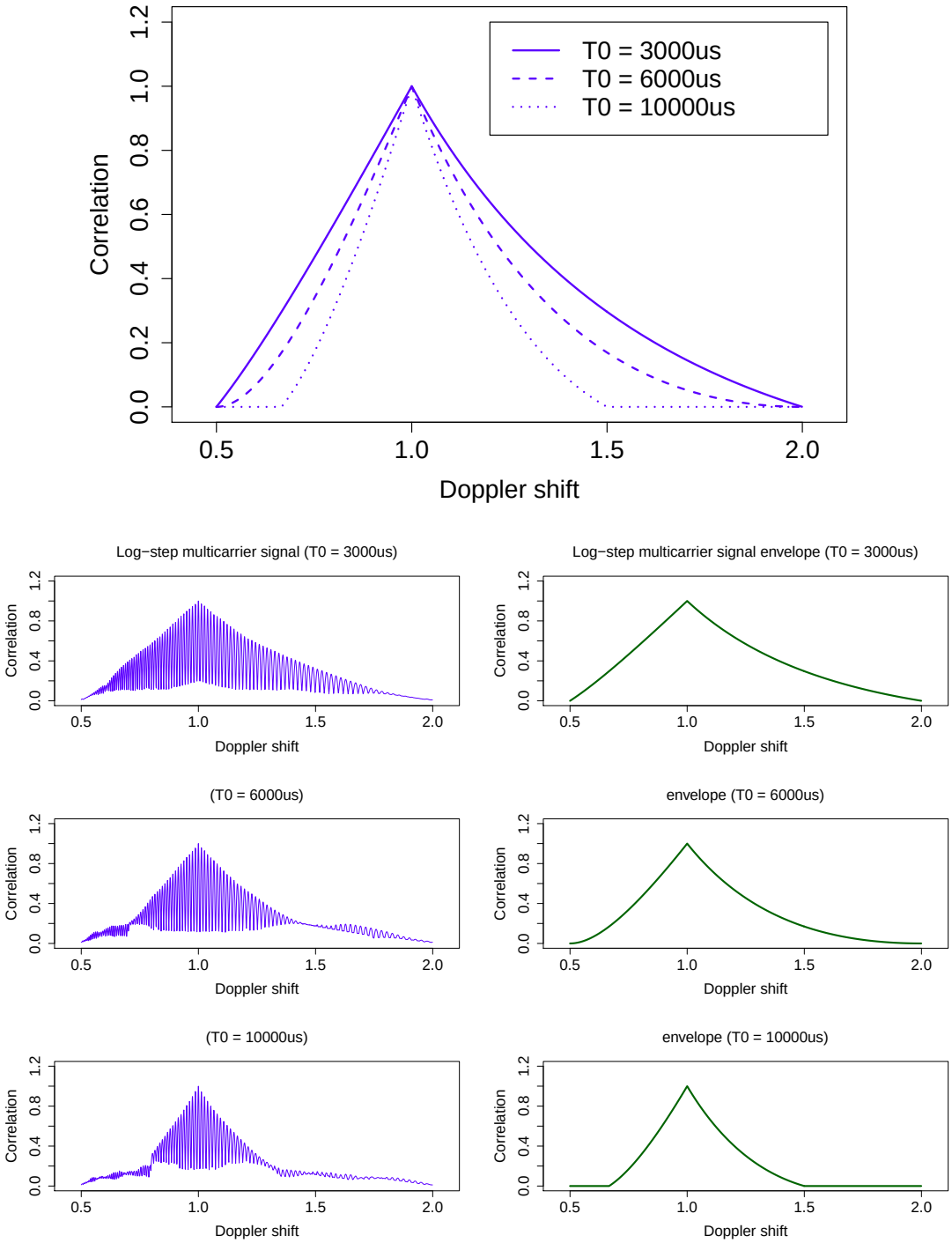


図 3.26　$T_0$ を変化させたときのプロファイルと包絡線の変化

## 3.8　プロファイル微細構造

相互相関プロファイルの微細構造について考察する。プロファイル微細構造とはプロファイルとその包絡線関数との差であり、相互相関値の極大値の近傍での急峻な落ち込みがこれを特徴づけている。そこでプロファイルがどのような点において極大値を持つのかについて復習しておく。

整列位相ログステップマルチキャリア波 $g_{p,q}(t,T_0)$ は複数のキャリアから構成されており、キャリア周波数比が一定である。ドップラーシフトはこのキャリア周波数を乗法的に移動するため、キャリア周波数比を一定のまま保つ。そのためもともとの周波数比と同じだけのドップラーシフトが受信波に作用して受信波のすべてのキャリアの周波数を高くすると、受信波のキャリアの周波数が、もともと隣接していた送信波の別のキャリアの周波数と一致する。この現象は受信波のすべてのキャリアについて起こり、受信波のキャリアはそれぞれ周波数がもともと隣接していた送信波のキャリアと周波数が一致する。ただし最高の周波数を持つ受信波のキャリアは周波数の一致する送信波のキャリアを失って相互相関値の計算上欠落する。この現象はドップラーシフトがキャリアの周波数を低くする場合にも同様であり、この場合には最低の周波数を持つ受信波キャリアが一致する相手を失って欠落する。

この現象がドップラーシフトが離散的な段階に至るごとに順繰りに起こり、ドップラーシフトが受信波の周波数を等比級数的に高くすると、周波数の一致するキャリアの数が 1 つずつ減少するということになる。周波数を低くする場合も同様である。相互相関の極大値は送信波と受信波で各キャリアの周波数と位相の一致する時に与えられ、相互相関の系統偏移のところで示されたように、整列位相ログステップマルチキャリア波において、周波数の一致するキャリアの位相は系統偏移の時点で一致する。これらのことから、プロファイルの極大値はドップラー量 $\rho$ が $k$ を整数として離散的な値 $\rho = p^{k/q}$ を持つときに現れることになる。

次にこれらの極大値の近傍でのプロファイルの振る舞いについて議論する。整列位相ログステップマルチキャリア波のプロファイルは $\rho = 1$ において最大値をとり、$rho = p^{k/q}$ において極大値をとり、最大値は $\rho = 1$ すなわち $k = 0$ のとき与えられる。なぜなら $\rho = 1$ のときに周波数の一致するキャリア数が最大となり、位相一致区間の長さも最長になるからである。$k \neq 0$ の場合は、一致する周波数の帯域と一致するキャリア数が異なるだけなのでプロファイルの振る舞いは最大値の近傍でのプロファイルの振る舞いから類推して理解することができる。

そこで最大値の近傍でのプロファイルの振る舞いを見ることにする。整列位相ログ

ステップマルチキャリア波 $g(t) = r_{[-T,T]}(t)g_{p,q}(t, T_0)$ のプロファイルにつき、次の定理が成り立つ。

**［定理］ 4** 整列位相ログステップマルチキャリア波のプロファイル $\lambda_g(\rho)$ は $\rho = 1$ の近傍で次式 $\mu_g(\rho)$ により近似される。

$$\lambda_g(\rho) \simeq \mu_g(\rho) = 2T \sum_{k=0}^{q-1} \mathrm{sinc}\left[2\pi T f_0(1-\rho)p^{k/q}\right] \tag{3.28}$$

証明

まず定義により、$\lambda_g(\rho) = \max_{-\infty<t<\infty}\{\|\langle g|\mathfrak{D}_\rho g\rangle(t)\|\}$ である。この最大値は時間原点において与えられるので、$\lambda_g(\rho) = \langle g|\mathfrak{D}_\rho g\rangle(0)$ である。

初期時刻 $T_0 = 0$ の整列位相ログステップマルチキャリア波を信号区間を伴って考えると、その各キャリア $c_k$ は、周波数が $f_0 p^{q/k}$ の正弦波と、信号区間 $[-T, T]$ において値を持つ矩形関数 $r_{[-T,T]}(t) = \mathrm{rect}\left(\frac{t}{2T}\right)$ の積として表される。

$$\begin{aligned} c_k(t) &= r_{[-T,T]}(t)\ \exp\left[2\pi j f_0 p^{k/q} t\right] \\ &= \mathrm{rect}\left(\frac{t}{2T}\right)\ \exp\left[2\pi j f_0 p^{k/q} t\right] \end{aligned} \tag{3.29}$$

これをフーリエ変換して次式が得られる。

$$\mathfrak{F}c_k(\xi) = 2T\,\mathrm{sinc}\left[2T\pi(\xi - f_0 p^{k/q})\right] \tag{3.30}$$

この式はキャリア $c_k$ の周波数 $\xi$ の区間付き正弦波に対する相関特性を表しており、$\xi = f_0 p^{m/q}\rho$ として次式を得る。

$$\mathfrak{F}c_k(f_0 p^{m/q}\rho) = 2T\,\mathrm{sinc}\left[2\pi T(f_0 p^{m/q}\rho - f_0 p^{k/q})\right] \tag{3.31}$$

この式を使えば、複数キャリアから成るログステップマルチキャリア波の相互相関は、次式で表現されることになる。

$$\langle g|\mathfrak{D}_\rho g\rangle(0) = 2T \sum_{m=0}^{q-1}\sum_{k=0}^{q-1} \mathrm{sinc}\left[2\pi T(f_0 p^{m/q}\rho - f_0 p^{k/q})\right] \tag{3.32}$$

$\rho = 1$ の近傍において、寄与の小さな部分を無視すると、

$$\langle g|\mathfrak{D}_\rho g\rangle(0) \simeq 2T \sum_{k=0}^{q-1} \mathrm{sinc}\left[2\pi T f_0(1-\rho)p^{k/q}\right] \tag{3.33}$$

従って与式の近似式が得られた。

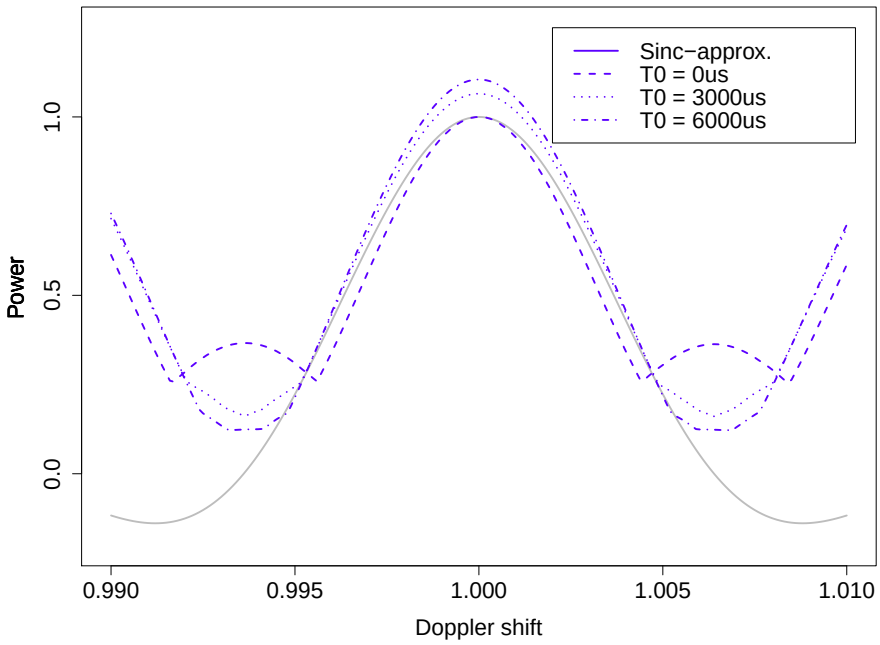


図 3.27　中心付近のプロファイル微細構造と sinc 近似

□

図 3.27 に、この近似式により求めたプロファイル微細構造（Sinc-approx.）と、数値計算により $T_0 = 0, 3000, 6000$ (us) の場合に求めたプロファイル微細構造を比較しておく。但し相関値は近似式による計算の最大値を 1 とする相対値である。

他の位相一致点 $\rho = p^{k/q}(k \neq 0)$ においても、一致キャリアの範囲と個数が変化するだけで同様の議論が成立するが、煩雑を避けるためにここでは議論を省略する。

この近似式の重要なところは、プロファイル微細構造の頂点に向かう部分の単調増加性と、頂点を離れる部分の単調減少性を保証しているということである。それは和を構成する sinc 関数の単調増加性と単調減少性から導きだされる。このことは後に、プロファイル微細構造を用いたドップラー速度推定アルゴリズムの設計をするにあたって役立てられることになる。ここで無視できるとした隔たりのあるキャリア間の干渉は、キャリア間隔が密になるとやがて無視できなくなり、プロファイルの谷を埋めて微細構造を消失させることになることにも注意しておきたい。

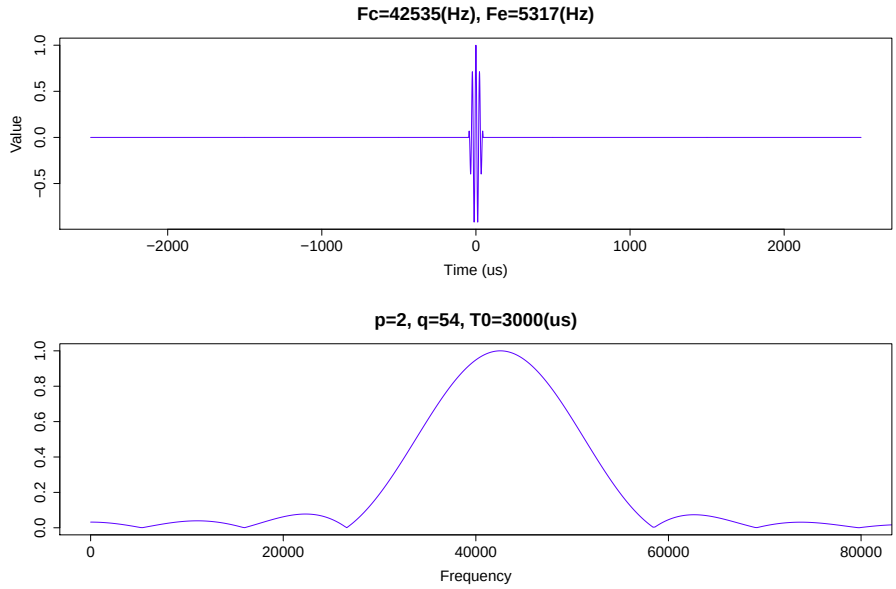


図 3.28　パルス波の波形とスペクトル

## 3.9　他の広帯域信号との比較

本章で提案したログステップマルチキャリア波の特長を理解するために、イメージングで用いられる他の広帯域信号との比較を行う。ここで比較するのは、パルス波、OFDM 波及びログスイープチャープ波である。

パルス波（図 3.28）とは非常に短い信号区間を持つ波形である。インパルスそのものは周波数帯域が無限に広がるため送信できないため、パルス波を構成するためにキャリアと包絡線の積が用いられる。パルス波は次式により定義される。

$$z(t) = e(t) \exp\left[2\pi j f_c t\right] \tag{3.34}$$

但しキャリア周波数を $f_c$ とする。$e(t)$ は包絡線関数であり、$f_e$ を包絡線関数の周波数として、余弦関数 $\cos(2\pi j f_e t)$ や、レイズドコサイン曲線 $\frac{1}{2}\cos(2\pi j f_e t) + \frac{1}{2}$ が良く用いられる。

パルス波はイメージングのためにインパルスを再構成する相関演算を必要としないのが最大の特徴である。カラードップラー法はパルス波の自己相関演算を実施しているがこれはインパルスの再構成ではなくドップラー位相の検出の目的で行っている。パルス波を送信波に用いると、相関演算を行う必要がないため高速な処理に向いているが、その場合は相関演算によるノイズの影響の低減効果を期待できない。パルス波もまた広帯域信号である。パルス波のスペクトルはキャリアのスペクトルと包絡線関

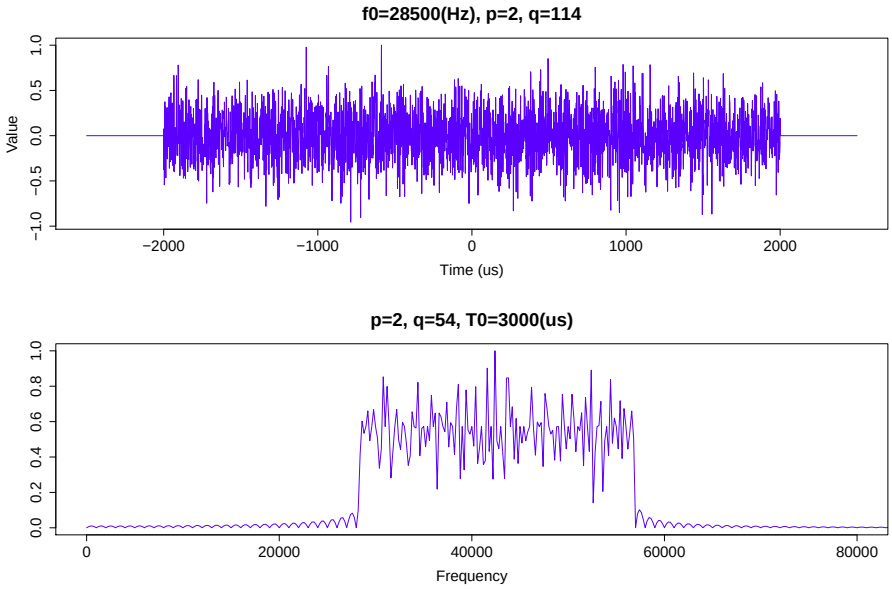


図 3.29　等ステップマルチキャリア波の波形とスペクトル

数のスペクトルの畳み込みにより決定するため、実際上包絡線関数のスペクトルがパルス波のスペクトルを決定することになるからである。

次に OFDM 波を紹介する。一般的にマルチキャリア波といえばもっぱら等ステップマルチキャリア波（図 3.29）のことをいう。等ステップマルチキャリア波は通信に用いる場合に変調効率が良く、簡単な装置で高速に変調及び復調できることから最近注目を集めている。等ステップマルチキャリア波は次式により定義される。

$$z(t) = \sum_{k=0}^{q-1} \alpha_k \exp\left[2\pi j \left(1 + \frac{k}{q}(p-1)\right) f_0 t + \phi_k\right] \tag{3.35}$$

但し $f_0$ を基本周波数とする。帯域幅 $p$ とキャリア数 $q$ の意味はログステップマルチキャリア波のものに合わせてある。各キャリアの振幅を $\alpha_k$ で決定し、初期位相は $\phi_k$ で決定する。

OFDM 波は等ステップマルチキャリア波の一種である。等ステップマルチキャリア波を OFDM 信号として用いる場合、信号長とキャリア数は基本周波数と帯域から自動的に決定する。また、OFDM 波には、通信データによる $\alpha_k$ と $\phi_k$ の変調が伴う。OFDM 波のスペクトルは、各キャリアのスペクトルが等間隔で並んだ櫛形となる。

ここで整列位相ログステップマルチキャリア波と同様の位相の決定方法が、OFDM 波では困難であることを指摘しておく。整列位相ログステップマルチキャリア波の場合と同様に、各キャリアの初期位相 $\phi_k$ を単一の定数 $T_0$ で決定すると、整列位

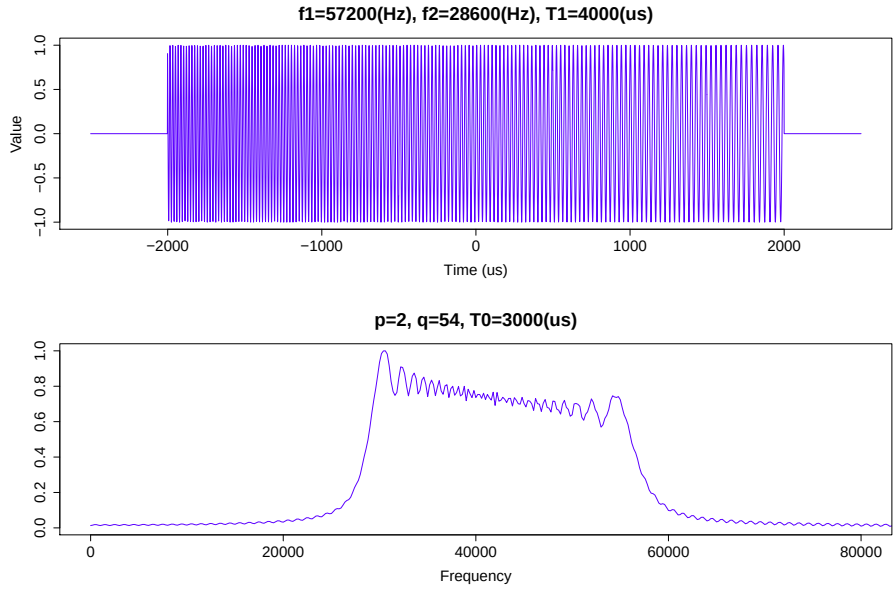


図 3.30　ログスイープチャープ波の波形とスペクトル

相 OFDM 波が定義できる。すなわち $\phi_k = 2\pi f_0 \left(1 + \frac{k}{q}(p-1)\right) T_0$ と仮定する。信号長を $2T$ とし、キャリア間隔は $1/(2T)$ とする。すると、整列位相 OFDM 波は $t = -T_0$ の時刻に鋭いピークを持つことがわかる。ところが、同波形は周期が $2T$ の周期関数であることから、 $t \equiv -T_0 (\mathrm{mod}\ 2T)$ が成り立つ位置にピークをもつことになる。このような時刻 $t$ は、区間 $[-T, T]$ 内にいつでも選ぶことができるため、整列位相 OFDM 波は信号区間内に必ず鋭いピークを持つことになってしまう。

チャープ波は単一キャリアの周波数が時間と共に変化する信号である。特徴としては、ピーク対平均電力比を小さくできること、連続スペクトルを持つことが挙げられる。チャープ波では、周波数を変化させるために時刻 $t$ に関数が乗ぜられる。この関数のことをスウィープ関数と呼び、その性質によってチャープ波はいくつかの種類に分けられる。中でもログスイープチャープ波は指数関数的に周波数を変化させて得られる連続波である。周波数の指数関数的な変化はドップラーシフトの乗法性と整合が良く、既存手法においてドップラーイメージングに良く用いられてきた。

ログスイープチャープ並は次式により定義される。

$$z(t) = \exp\left[2\pi j \frac{f_1 T_1}{\log(f_2/f_1)} \left(\exp\left[\frac{\log(f_2/f_1)t}{T_1}\right] - 1\right)\right] \tag{3.36}$$

ここで $f_1$ は開始周波数、$f_2$ は終了周波数、$T_1$ は信号長である。時刻 $t$ で式を微分すると係数に周波数が得られることに注意するとが良く理解できる。チャープ波は $f_1 < f_2$ が成り立つときアップチャープ波と呼ばれ、$f_2 < f_1$ が成り立つときダウン

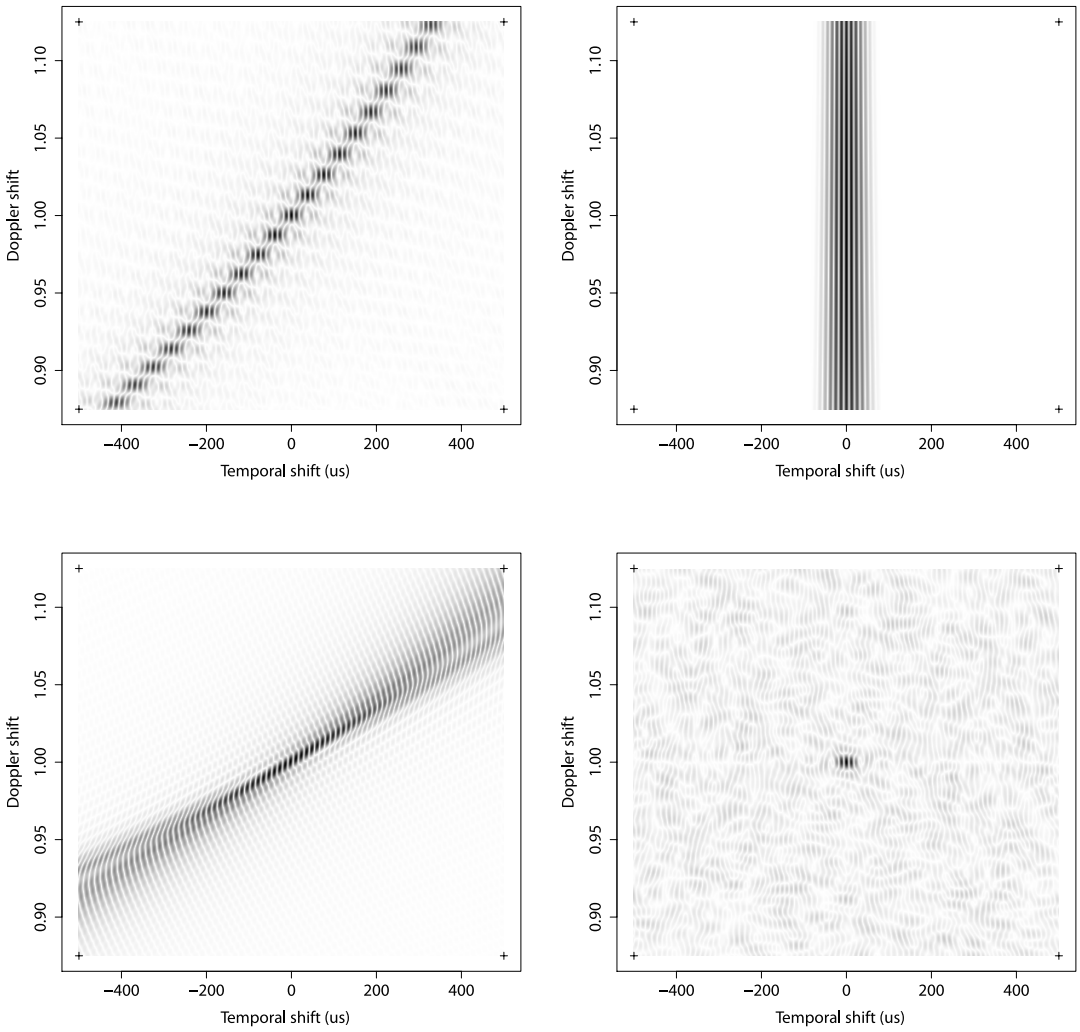


図 3.31　ログステップマルチキャリア波と、パルス波、ログスイープチャープ波、OFDM 波のドップラー解析結果の比較 (左上:ログステップマルチキャリア波、右上:パルス波、左下:ログスイープチャープ波、右下:OFDM 波)

チャープ波と呼ばれる。ログスイープチャープ波のスペクトルは独特のものであり、開始周波数と終了周波数のところに 2 つピークを持ち、2 つのピークの間が波打つ稜線によって結ばれた形となる。2 つのピークの外側はなだらかに減衰する。

次にドップラー解析を、パルス波、OFDM 波、ログスイープチャープ波に適用することにより、ドップラー感受性とドップラー耐性をログステップマルチキャリア波のものと比較する。図 3.31 がその比較結果である。

パルス波にドップラー解析を適用すると、時間方向に鋭くドップラー量方向にぼやけた像が得られる。時間方向の解像度はパルス幅に依存しているのでここでの比較は

あまり意味がない。ドップラー量方向にぼやけているということは、ドップラー感受性が鈍くドップラー耐性があるということであり、パルス波はログステップマルチキャリア波に比べて、大まかにドップラーシフトを検出するのに向いているということができる。

ログスイープチャープ波をドップラー解析すると、ログステップマルチキャリア波と同様にピーク時刻の系統偏移が確認できる。これは、ドップラーシフトが受信波の周波数に変化をもたらし、送信波の周波数と一致する部分を前後にずらす働きをするからである。

よく知られているように OFDM 波では、ドップラーシフトによって各キャリアの直行性が失われ、相互相関値もドップラーシフトによって急速に失われる傾向にある。OFDM 波のスペクトルはホワイトノイズのそれに近いため、ドップラーシフトによって他のホワイトノイズが作り出され、それらの相互相関は 0 に近くなるからである。このため、OFDM 波のドップラー解析結果はドップラー量 $\rho = 1$ の場合にのみ時間原点にピークを持つものとなる。

以上の比較から、ログステップマルチキャリア波に現れる相互相関ピーク時刻系統偏移は、ログスイープチャープ波に類似のものであるということができる。このため、チャープ波の系統偏移を利用して開発されたアルゴリズムが、ログステップマルチキャリア波にも類似的に適用できる可能性があると結論づけることができる。

次にパルス波、ログスウィープチャープ波、OFDM 波の相互相関プロファイルをログステップマルチキャリア波の相互相関プロファイルと比較する（図 3.32)。

パルス波のプロファイルでは、ドップラー効果が小さな間に保たれている相関が、ドップラー効果が大きくなるにつれ徐々に失われて小さくなる。これはキャリア周波数がドップラー効果によって変化するためであり、送信波と受信波でキャリア周波数が異なることから生じる。またログステップマルチキャリア波に見られた微細構造はパルス波には見られない。

ログスウィープチャープ波のプロファイルは、ドップラー効果が小さなうちはピークの値は高いレベルに保たれるが、ドップラー効果が大きくなるにつれてピーク値はゆるやかに失われて小さくなる。この現象はログステップマルチキャリア波に似ているが、ログステップマルチキャリア波に見られた微細構造はログスウィープチャープ波には見られない。

OFDM 波のプロファイルは、ドップラー量 $\rho = 1$ の位置にのみピークを持つものになる。このピークは急峻であるため、ドップラー感受性が高く、反面ドップラー耐性は低い。そのため、OFDM 波は小さなドップラーシフトを捉えるには向いているが、大きなドップラーシフトを捉えるには不向きである。

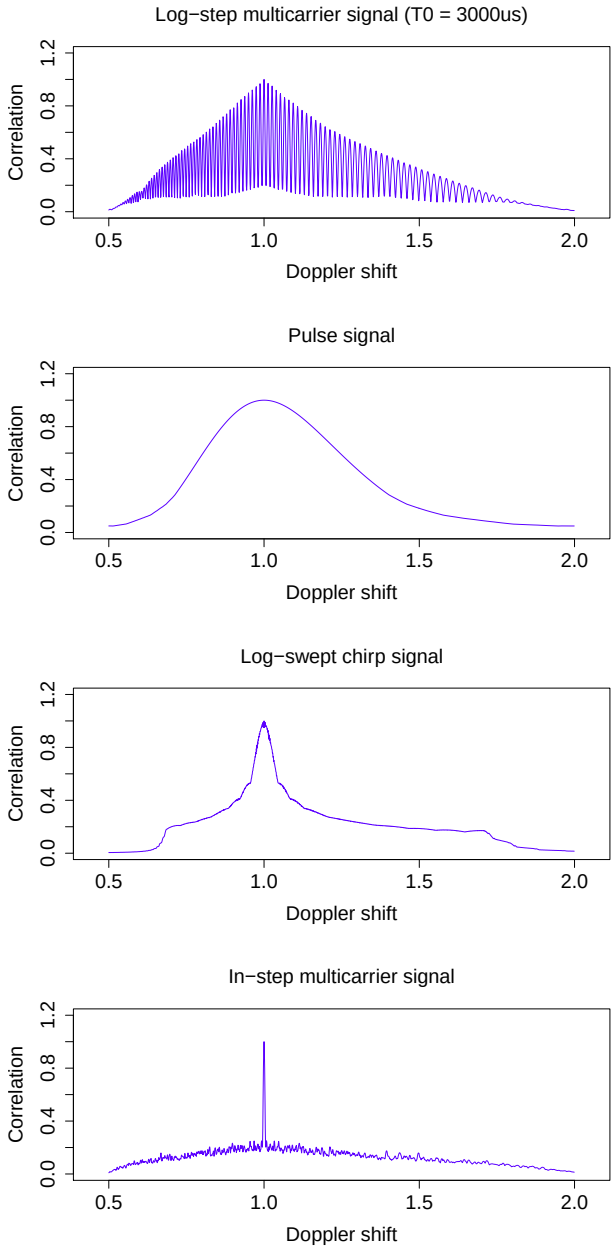


図 3.32　ログステップマルチキャリア波と、パルス波、ログスウィープチャープ波、等ステップマルチキャリア波のプロファイルの比較

以上の考察から、プロファイルに現れる微細構造とドップラー量の広い範囲にわたる高い相関値を併せ持つ事がログステップマルチキャリア波の特長であることがわかる。ログステップマルチキャリア波は、片方で OFDM 波のような鋭いドップラー感受性を持ちながら、もう片方でチャープ波のような広いドップラー耐性を持つ。これは次章以降で述べるアルゴリズムの設計に役立つものである。

# 第 4 章

# ログステップマルチキャリア波を用いたドップラーイメージング

本章では、前章で導入したログステップマルチキャリア波を用いたドップラーイメージング法を提案し、実機実験によりその検証を行う。

ドップラーイメージングに求められる要件の一つに、ドップラー量の大小に対する耐性がある。通常ドップラーシフトがわずかなときに結果が得られるのは、どのイメージング法でも同様であるが、ドップラーシフトが大きくなると結像は失われやすい。本章では、ログステップマルチキャリア波の特性を生かして、ドップラー量が大きなときにも確実に動作するアルゴリズムを提案する。

ドップラーイメージングにおいて、検出した速度情報は通常、画像を彩色することによって表示される。医用超音波のカラードップラー法はその代表例であり、近づく物体を赤、遠ざかる物体を青に彩色する。もう一つの代表例は各種の干渉法である。干渉法は相関計算によって得られた位相情報に彩色することによって速度を表示する。したがって干渉法には $2\pi$-曖昧性と呼ばれる表示速度の多義性が生じる。

本章で提案する速度表示の方法は、カラードップラー法のものとも干渉法のものとも異なるが、アルゴリズムの性質上速度表示の曖昧性が発生する。この曖昧性はログステップマルチキャリア並のプロファイル微細構造に起因するが、その解消を行うことは次章以降の課題とし、本章ではイメージング法のみを扱う。

## 4.1 提案手法のイメージング方法

イメージングを行うために、送信波と受信波の間にマッチドフィルタを適用し、到来時間に鋭いピークを持つディラック $\delta$ 関数を再構成する。送信波を整列位相ログス

テップマルチキャリア波 $g_{p,q}(t, T_0)$ とする。本章では各パラメーターを次のように設定し実験を進める。帯域幅 $p = 2$、キャリア数 $q = 54$、初期時刻 $T_0 = 3000$ (us) とする。基本周波数は $f_0 = 28000$ (Hz) と設定する。この整列位相ログステップマルチキャリア波の区間を切り取り送信波を構成する。半区間長 $T = 2000$ (us) と設定し、送信波 $g(t)$ を次の関数により構成する。

$$g(t) = r_{[-T,T]}(t) \cdot g_{p,q}(t, T_0) \tag{4.1}$$

反射波はセンサーアレイにより受信する。センサーアレイは $z = 0$ 平面上に配置され、各センサーの座標を $(\xi, \eta, 0)$ として、この座標系に置かれたセンサーの受信信号を $s_{\xi\eta}(t)$ と表すことにする。

この 2D センサーアレイにより受信された信号を、遅延和ビームフォーマーにより方向成分に分解する。

$$B_{xy} = \sum_{\xi=-X}^{X} \sum_{\eta=-Y}^{Y} s_{\xi\eta}(t + \Delta_{xy\xi\eta}) \tag{4.2}$$

ここで $\Delta_{xy\xi\eta}$ が各方向成分、各センサー座標ごとの遅延時間を定義する。

次にマッチドフィルターを適用して、受信波からインパルスを再構成する。通常マッチドフィルターの参照波に用いられるのは送信波と同一の波形である。これによりノイズの含まれる受信波から、送信波に類似する部分を抽出して到来時刻に鋭いインパルスを持つ波形を再構成するのである。

ところがドップラーイメージングの場合、受信波はドップラーシフトの影響を受けているため、単純に送信波を参照波としてマッチドフィルターを適用してもピークが得られない場合がある。今回は送信波及び参照波に整列位相ログステップマルチキャリア波を用いているので、ドップラーシフトの影響が大きい場合にでも相互相関によるピークは失われにくい。その整列位相ログステップマルチキャリア波も、ドップラーシフトに対して櫛の歯状の相関を示すに過ぎず、このままでは相関関数のピークがこの櫛の歯の谷に落ち込む可能性がある。そこで、次の方法でプロファイルの微細構造の間を参照波をいくつか用意してフィルターバンクを構成し、複数の相関計算を行うことによってこれを補うことにする。

送信波 $g_{p,q}(t, T_0)$ から周波数を上下にシフトさせた 3 種の参照波

$$\begin{cases} g^R(t) = g_{p,q}(p^{-1/3q}t, T_0) \\ g^G(t) = g_{p,q}(t, T_0) \\ g^B(t) = g_{p,q}(p^{1/3q}t, T_0) \end{cases} \tag{4.3}$$

を用意し、これらの参照波を用いて受信波との間にそれぞれマッチドフィルターを適用する。今回はビームフォーマーを先に適用しているので、マッチドフィルターは各方向成分 $B_{xy}(t)$ に対して適用されることになる。

$$\begin{cases} X_{xy}^{B}(t) = \int_{-\infty}^{\infty} g^{B}(\tau - t) B_{xy}(\tau) d\tau \\ X_{xy}^{G}(t) = \int_{-\infty}^{\infty} g^{G}(\tau - t) B_{xy}(\tau) d\tau \\ X_{xy}^{R}(t) = \int_{-\infty}^{\infty} g^{R}(\tau - t) B_{xy}(\tau) d\tau \end{cases} \tag{4.4}$$

こうして得られた $X_{xy}(t) = (X_{xy}^{B}(t), X_{xy}^{G}(t), X_{xy}^{R}(t))$ を時刻 $t$ における青、緑、赤の輝度として用いてイメージングを行う。

## 4.2 既存手法のイメージング方法

カラードップラー法のイメージング方法には Kasai ら [KNKO85] のものを採用する。伝統的な方法の多くは包絡線検波のために直交検波と低域透過フィルターを使用しているが、ここではヒルベルト変換を用いた方法を紹介する。

カラードップラー法は一定の間隔でパルス波を連続的に発生させる。パルス波とはキャリアを包絡線関数により振幅変調して得られる波形であり、$z(t)$ を複素包絡線関数として次式により定義される。

$$g(t) = z(t) e^{j\omega_0 t} \tag{4.5}$$

この参照波の実部を間隔 $T_1$ を持って連続送出し、送信波とする。

受信波 $h(t)$ は送信波と同様に実信号であるため、このままではカラードップラー法の処理に適さない。この実信号に対して因果性を付与し符号付きの位相を求めるために、ヒルベルト変換を用いて解析信号を再構成する。ヒルベルト変換を $\mathfrak{H}$ と表すと受信波の複素化は、

$$w(t) = h(t) + j\mathfrak{H}\left(h(t)\right) \tag{4.6}$$

で得られる。

包絡線とはすなわち複素化された受信波の絶対値であるから、

$$x(t) = \|w(t)\| = \sqrt{\left(h(t)\right)^2 + \left(\mathfrak{H}\left(\left(h(t)\right)\right)^2} \tag{4.7}$$

得られた実包絡線関数を更にヒルベルト変換して複素包絡線関数を再構成する。

$$z'(t) = x(t) + j\mathfrak{H}(x(t)) \tag{4.8}$$

カラードップラー法はこの関数 $z'(t)$ をパルス間隔 $T_1$ をもって干渉法を適用する。

$$R_1(t) = \int_{t-T_1/2}^{t+T_1/2} \overline{z'}(\tau - T_1) z'(\tau) d\tau \tag{4.9}$$

こうして得られた $R_1(t)$ からドップラー位相を計算する。

$$\phi(t) = \tan^{-1} \frac{\Im\left[R_1(t)\right]}{\Re\left[R_1(t)\right]} \tag{4.10}$$

この式から最終的に所望の到来時刻 $t_0$ における位相 $\phi(t_0)$ を計算し、ドップラー位相を得る。得られたドップラー位相の符号に従って、赤、青の輝度を与えイメージングを行う。医用超音波における D モード画像では反射体がセンサーに向けて近づく場合に赤、遠ざかる場合に青の木戸が与えられるが、本章ではそれとは逆に光の赤方偏移、青方偏移になぞらえて近づく場合に青、遠ざかる場合に赤の輝度を割り当てることにする。

## 4.3　イメージングアルゴリズムの計算量

提案イメージング手法における各方向成分に対する計算量は純粋に 3 種の参照波に対する相関計算の計算量であり、$M$ を参照波のサンプル数、$N$ を受信波の方向成分内のサンプル数とし、乗算のコストを 1 とカウントして次のように決まる。

$$P_{RGB} = 3MN \tag{4.11}$$

相互相関は、高速フーリエ変換を使用し、参照波 $g(t)$ と方向成分波形 $h(t)$ の相互相関を $\langle g|h\rangle(t) = \mathfrak{F}^{-1}(\overline{\mathfrak{F}g} \cdot \mathfrak{F}h)(t)$ のように計算すると、 計算量は次式で表されるものになる。

$$P'_{RGB} = 2N \log_2 N + 3N \tag{4.12}$$

カラードップラー法における計算量は自己相関計算に加えてヒルベルト変換と低域透過フィルターのコストが加わるため、$M$ を送信波のサンプル数、$N$ を各方向成分内のサンプル数、$H$ をヒルベルト変換フィルターのサンプル数、$L$ を低域透過フィルターのサンプル数とし、乗算のコストを 1 とカウントして次のように決定する。

$$P_{CD} = MN + 2NH + NL \tag{4.13}$$

これらの式からはカラードップラー法の計算量の方が多いように見えるが、実際には参照波のサンプル数 $M$ や、方向成分内のサンプル数 $N$ に比べてヒルベルト変換

フィルターのサンプル数 $H$ や 低域透過フィルターのサンプル数 $L$ はずっと小さいため、項 $MN$ が支配的になり、提案イメージング手法の計算量のほうが多くなる。

また、カラードップラー法にも高速フーリエ変換が使用できるが、高速フーリエ変換のコストが支配的であるため、この場合には提案イメージング手法に対するカラードップラー法の計算コストにおける優位性はない。

## 4.4　イメージング実験方法

実験装置の配置を (図 4.1) に示す。あらかじめ設定した送信波は、20kHz〜120kHz の再生可能帯域を持つ送信機 (Pioneer PT-R4) から発せられ、MEMS マイクロフォンアレイによって受信する。反射体は 1000mm のロッドの先に取り付けられ、モータードライブにより最高 180rpm で回転して、ドップラーシフトを伴って超音波を反射する。

回転するロッドに付けられた反射体は、マイクロフォンアレイの平面と垂直をなして運動する時に最大のドップラー効果をもたらす。高速回転するロッドがちょうどこの位置に来た時に適切なトリガをシステムに一度だけ送出する必要がある。今回の実験では、円板とフォトインタラプタ及びワンショットマルチバイブレーターを用いた回路を製作し、これを用いた。ロッドの下に取り付けられたフォトインタラプタが、回転する円板の切欠き部を検出し、ワンショットマルチバイブレーター回路により処理されて、適切なタイミングでシステムにトリガが送出される。

送信機と受信機の構成を (図 4.2) に示す。受信機は、今回製作したシステムを用い

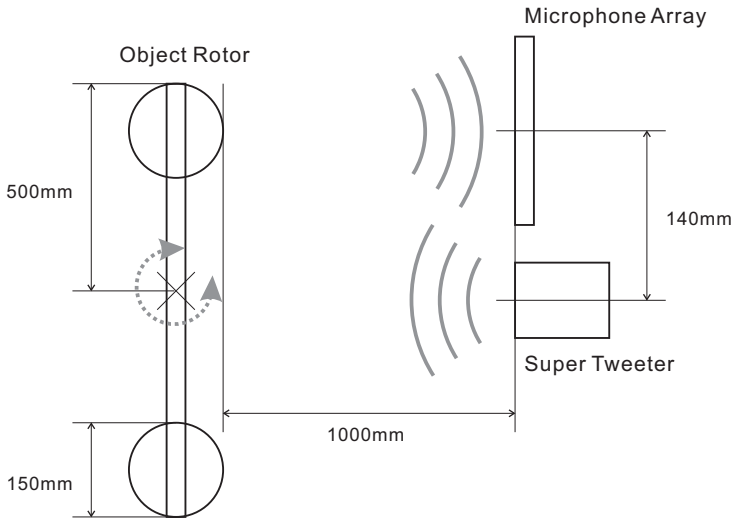


図 4.1　実験装置の配置

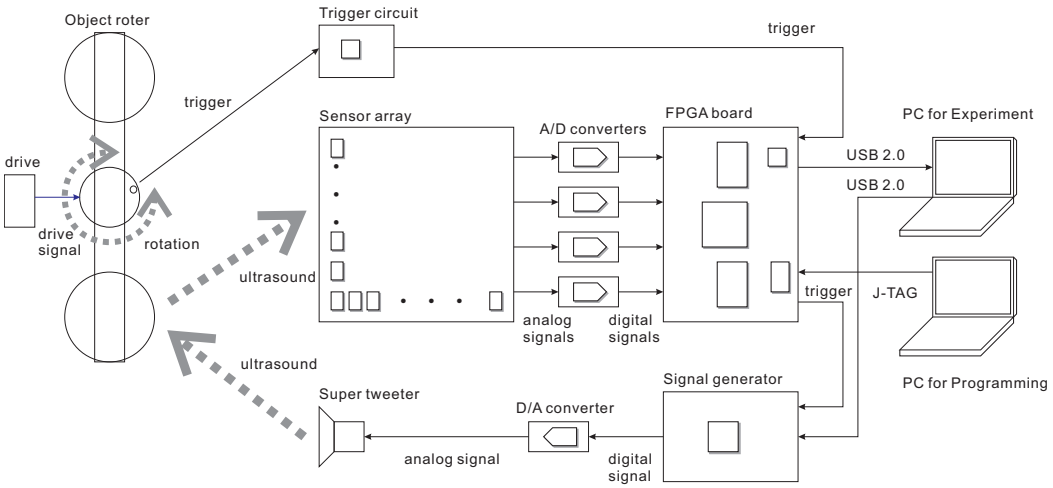


図 4.2　送信機と受信機の構成

たもので、128ch、12bit、500ksps のサンプルデータを最大 128ms 分格納する。受信機システムは外部トリガに与えられたパルスを検出して受信を開始すると同時に送信機にトリガを送出する。送信機は、16bit、1Msps のサンプルデータを最大 64ms 分格納することのできるもので、トリガー信号に応じて送信波の信号を増幅器に送り出す。送信機から出た送信波は 3ms 程度で反射体に到達し、ドップラーシフトを伴って反射される。反射体の動きは自由な速度で正転、逆転が可能なようにモータードライブが設計されており、様々な条件でドップラーシフトを実験することができる。反射体で反射した音波はやはり約 3ms で送信機に近接して設置されたセンサーアレイに到達し、128ch 分同時に受信機に送られる。これらは 同数の A/D コンバータによりサンプリングされて、受信機に内蔵されたバッファメモリに格納される。受信がすべて終了した後、バッファメモリに格納されたデータは USB 2.0 インターフェースを介して、PC に送られる。PC 上でビームフォーマ、マッチドフィルタの処理を行い、最終的にイメージング画像が得られる。

カラードップラー法と提案手法のそれぞれについて、この実験装置を利用したイメージング実験を行った。

送信波としては次のものを使用した。カラードップラー法のパルス波をキャリア周波数 42535Hz、包絡線周波数 5317Hz とし、パルス間隔 $4000\mu s$ とした（図 4.3）。

提案手法のログステップマルチキャリア波を基本周波数 28600Hz、帯域幅 $p = 2$、キャリア数 $q = 54$ 整列位相オフセット $T_0 = 3000\mu s$、波形長 $2T = 4000\mu s$ とした（図 4.4）。

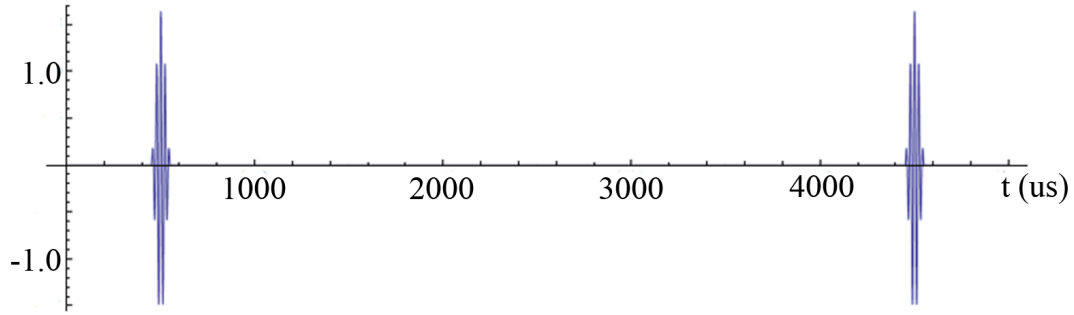


図 4.3　カラードップラー法で使用したパルス波

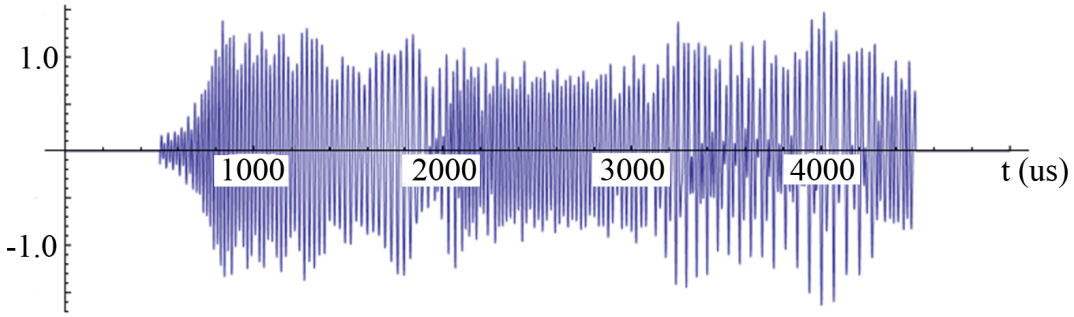


図 4.4　提案手法で使用したログステップマルチキャリア波

## 4.5　イメージング実験結果

実験の結果得られた、既存手法（カラードップラー法）による反射体位置のセクタースキャン画像及び、提案手法による同画像を次に示す。

掲載するイメージング結果は反射体速度によって全体で大きく 4 種類に大別され、微速（30rpm）、低速（60rpm）、中速（120rpm）、高速（180rpm）の順に挙げる。

まず微速の場合（図 4.5 図 4.6 図 4.7 図 4.8 ）には、既存手法、提案手法の間にはあまり差が見られない。反射体が接近する場合には、既存手法のほうが像が鮮明であり、離反する場合には、提案手法のほうが単一像が得られておりイメージング結果は良好である。

低速の場合（図 4.9 図 4.10 図 4.11 図 4.12 ）には、提案手法で色相が微速の場合と比べて交代していることが確認できる。また、既存手法に比べ提案手法がいくらか鮮明な像が得られる傾向が見られる。

中速の場合（図 4.13 図 4.14 図 4.15 図 4.16 ）には、既存手法で反射体像がぼやけて周辺部のノイズが目立つのに対し、提案手法では鮮明な反射体像が得られている。

表 4.1　既存手法（カラードップラー法）と提案手法によるイメージング結果

| 図番号 | 速度 | 方向 | 手法 |
|---|---|---|---|
| 図 4.5 | 30rpm | 接近 | 既存手法 |
| 図 4.6 | 30rpm | 接近 | 提案手法 |
| 図 4.7 | 30rpm | 離反 | 既存手法 |
| 図 4.8 | 30rpm | 離反 | 提案手法 |
| 図 4.9 | 60rpm | 接近 | 既存手法 |
| 図 4.10 | 60rpm | 接近 | 提案手法 |
| 図 4.11 | 60rpm | 離反 | 既存手法 |
| 図 4.12 | 60rpm | 離反 | 提案手法 |
| 図 4.13 | 120rpm | 接近 | 既存手法 |
| 図 4.14 | 120rpm | 接近 | 提案手法 |
| 図 4.15 | 120rpm | 離反 | 既存手法 |
| 図 4.16 | 120rpm | 離反 | 提案手法 |
| 図 4.17 | 180rpm | 接近 | 既存手法 |
| 図 4.18 | 180rpm | 接近 | 提案手法 |
| 図 4.19 | 180rpm | 離反 | 既存手法 |
| 図 4.20 | 180rpm | 離反 | 提案手法 |

また、提案手法で色相が回転して更に色が交代してことが確認できる。

高速の場合（図 4.17 図 4.18 図 4.19 図 4.20 ）には、既存手法では反射体の結像が難しくなっているが、提案手法では依然として鮮明な像が得られている。傾向としては接近する場合のほうが離反する場合よりも高精度であり、回転体する反射隊の後ろに発生する乱流の影響が現れているものと推察される。

## 4.6　考察

イメージング結果によって、ログステップマルチキャリア波を用いたドップラーイメージングアルゴリズムが有効に機能し、反射体速度に対応した色相が得られることが確かめられた。提案手法の大きなドップラー良に対する耐性は高く、カラードップラー法ではイメージング不可能な領域での安定したイメージング性能が確認できた。

いっぽう小さなドップラー量領域における、提案手法のカラードップラー法に対す

る優位性は、今回得られた結果からはそれほど明らかではなく、より正確な速度設定のできる実験装置を用いた実験および検証が今後必要とされる。

なお本章で紹介したログステップマルチキャリア波によるイメージング方法と結果は、[MSH11c] [MSH12a] [MSH12b] [MSH14] において発表したものである。

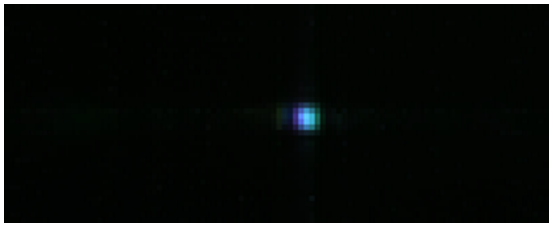

図 4.5　既存手法（カラードップラー法）により得られたドップラー画像 (接近 30rpm)

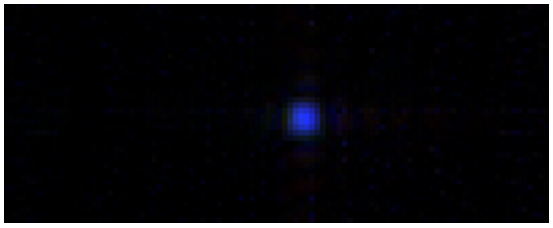

図 4.6　提案手法により得られたドップラー画像 (接近 30rpm)

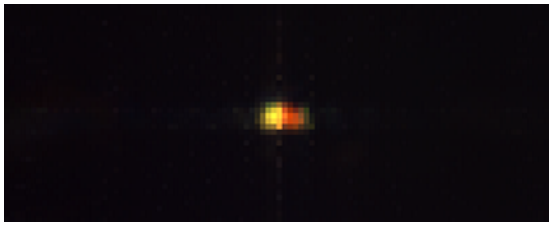

図 4.7　既存手法（カラードップラー法）により得られたドップラー画像 (離反 30rpm)

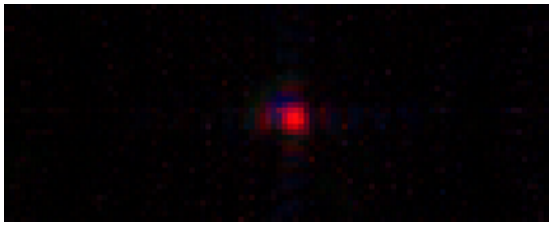

図 4.8　提案手法により得られたドップラー画像 (離反 30rpm)

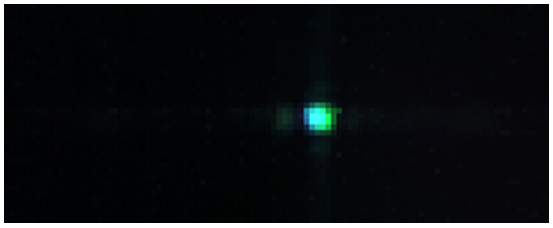

図 4.9 既存手法（カラードップラー法）により得られたドップラー画像 (接近 60rpm)

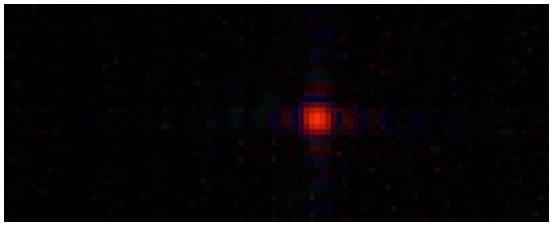

図 4.10 提案手法により得られたドップラー画像 (接近 60rpm)

図 4.11 既存手法（カラードップラー法）により得られたドップラー画像 (離反 60rpm)

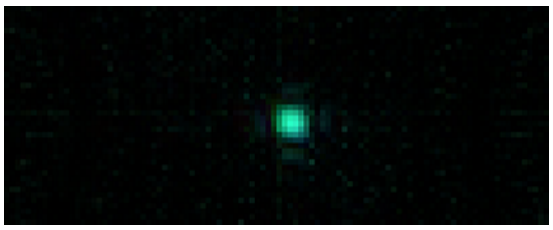

図 4.12 提案手法により得られたドップラー画像 (離反 60rpm)

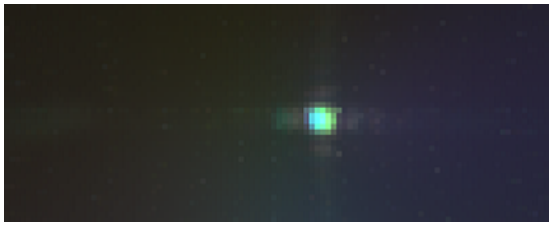

図 4.13　既存手法（カラードップラー法）により得られたドップラー画像 (接近 120rpm)

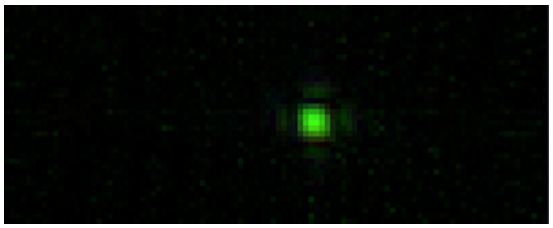

図 4.14　提案手法により得られたドップラー画像 (接近 120rpm)

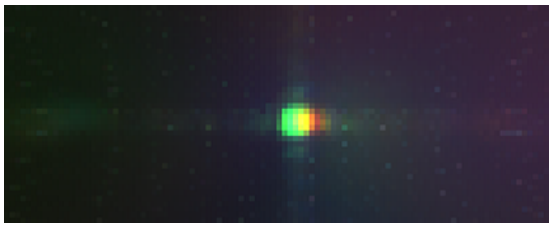

図 4.15　既存手法（カラードップラー法）により得られたドップラー画像 (離反 120rpm)

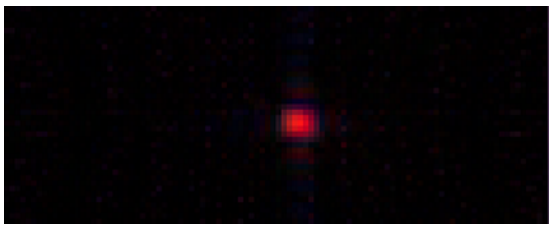

図 4.16　提案手法により得られたドップラー画像 (離反 120rpm)

図 4.17 既存手法（カラードップラー法）により得られたドップラー画像 (接近 180rpm)

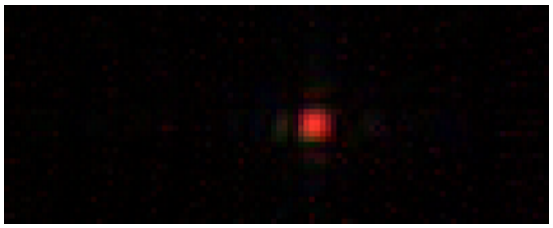

図 4.18 提案手法により得られたドップラー画像 (接近 180rpm)

図 4.19 既存手法（カラードップラー法）により得られたドップラー画像 (離反 180rpm)

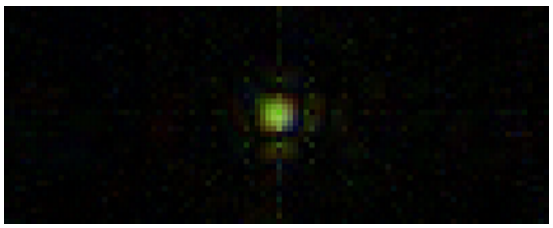

図 4.20 提案手法により得られたドップラー画像 (離反 180rpm)

# 第 5 章

# ログステップマルチキャリア波を用いたドップラー速度推定

前章のイメージング実験において、反射体速度に対応して交代的にピークを生じる３種のマッチドフィルター計算が導入され、反射体位置に彩色された像が得られることが確認された。本章ではこの色情報から速度情報を計算する方法について述べ、このアルゴリズムの速度推定能力についてシミュレーションにより検証を行う。

反射体速度はイメージングによって得られた画素の彩色情報を基にドップラー量を逆算して求めるが、この際に複数の速度が同一色にマッピングされていることがあり、曖昧性の問題と呼ばれている。一般的に曖昧性の解消は難しい課題であり、多くの場合、別手段による計算を必要とする。本章では曖昧性の解消のために便宜的にスペクトル解析による簡易な方法を与え、大きなドップラー量に対してのアルゴリズムの有効性を確かめる。

比較対象として再びカラードップラー法を採り上げる。カラードップラー法を用いても色情報から速度情報を算出することができ、具体的方法を本章で記述する。本章で行うシミュレーションによって、ログステップマルチキャリア波を用いたドップラー速度推定方法とカラードップラー法の比較を行う。このシミュレーションでは、提案手法は同一最大振幅の送信波を用いた場合で既存手法の約 20 倍、同一平均パワーの送信波を用いた場合で約 3 倍の推定精度を持つことが示される。

## 5.1　提案手法によるドップラー速度推定

本章で提案するログステップマルチキャリア波によるドップラー量推定では、ピーク比の曖昧性を解消するために正確な推定に先立ってドップラーシフト量の概算値を

知ることが必要になる。ドップラー量の概算値は例えば次のように求める。しきい値 $H$ を適当に定め、受信波のフーリエ変換 $\mathfrak{F}s(f)$ からスペクトルの上限値と下限値を得る。

$$f_1 = \arg\min_f \{\|\mathfrak{F}s(f)\| > H\} \tag{5.1}$$

$$f_2 = \arg\max_f \{\|\mathfrak{F}s(f)\| > H\} \tag{5.2}$$

$f_0$ を送信波に用いるログステップマルチキャリア波の基本周波数とすると、ドップラー量概算値 $\rho'$ は次式により求めることができる。

$$\rho' = \frac{1}{2f_0}\left(f_1 + \frac{f_2}{p^{(q-1)/q}}\right) \tag{5.3}$$

このスペクトル解析の結果から得られるものは、ドップラー量の概算量に過ぎない。次に我々は、これを用いてドップラー量の精密な推定を行う。

ドップラー量の精密な推定値は、イメージングの際に用いた 3 種の参照波による相互相関のピーク比を用いる。これらのピークは、お互いに他を補いながら広いドップラー範囲にわたって高い相関値を示す。加えてこれらの比は信号の減衰率に依存しないというドップラー量算出のために好ましい性質を持っている。

形式的な定義を与える。ピーク値 $x = (x^B, x^G, x^R)$ とは次の量である。

$$\begin{cases} x^B = \max_t X^B(t) \\ x^G = \max_t X^G(t) \\ x^R = \max_t X^R(t) \end{cases} \tag{5.4}$$

ドップラー量を推定するために、あらかじめ理想的な条件の下で構築された辞書を準備する。理想的な条件の下での相互相関関数 $M(\rho,t) = (M^B, M^G, M^R)(\rho,t)$ は次の計算により求めることができる。

$$\begin{cases} M^B(\rho,t) = \sqrt{\rho}\int_{-\infty}^{\infty} g^B(\tau - t)g(\rho\tau)d\tau \\ M^G(\rho,t) = \sqrt{\rho}\int_{-\infty}^{\infty} g^G(\tau - t)g(\rho\tau)d\tau \\ M^R(\rho,t) = \sqrt{\rho}\int_{-\infty}^{\infty} g^R(\tau - t)g(\rho\tau)d\tau \end{cases} \tag{5.5}$$

ドップラー量 $\rho$ を与えた時のこれらの関数のなすピーク値を用いて辞書 $m(\rho) = (m^B, m^G, m^R)(\rho)$ を構成する。

$$\begin{cases} m^B(\rho) = \max_t M^B(\rho,t) \\ m^G(\rho) = \max_t M^G(\rho,t) \\ m^R(\rho) = \max_t M^R(\rho,t) \end{cases} \tag{5.6}$$

辞書 $m(\rho)$ を用いて $x$ から $\rho$ を推定する問題とは、方程式 $x = m(\rho)$ を $\rho$ に関して解くことにほかならない。しかし $m(\rho)$ の持つ周期性のために、このままでは逆関数 $m^{-1}$ を求めることができない。

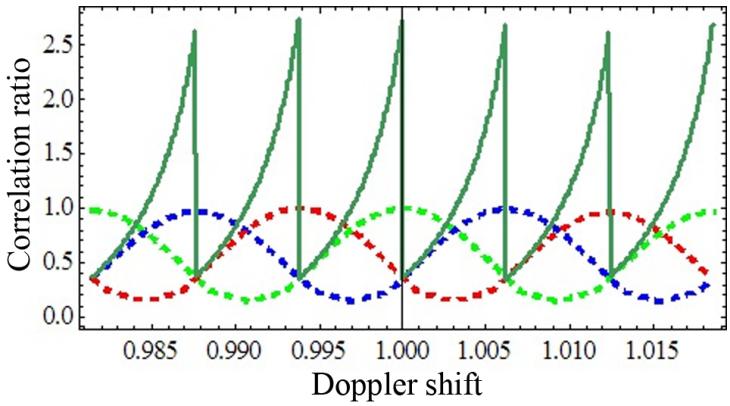


図 5.1　相互相関ピーク比による区分的単調増加関数の構成

逆関数 $m^{-1}$ を求めるために我々の採る方法は、ピーク値の為す比 $m^B(\rho)/m^G(\rho)$, $m^R(\rho)/m^B(\rho)$ 又は $m^G(\rho)/m^R(\rho)$ のいずれかが単調増加となる区間を選び、この区間の範囲内で推定を行うことである。$m^G(\rho)$ のピークは $k$ を整数として $\rho = p^{k/q}$ に現れ、$m^B(\rho)$ のピークは $\rho = p^{k/q+1/(3q)}$ に現れ、$m^R(\rho)$ のピークは $\rho = p^{k/q-1/(3q)}$ に現れることに注意しておく。

ピーク比にドップラー量を関連付ける関数 $\Psi(\rho)$ を次のように定義する。

$$\Psi(\rho) = \begin{cases} m^B(\rho)/m^G(\rho) \\ \quad \text{for } p^{k/q} \le \rho < p^{k/q+1/(3q)} \\ m^G(\rho)/m^R(\rho) \\ \quad \text{for } p^{k/q-1/(3q)} \le \rho \le p^{k/q} \\ m^R(\rho)/m^B(\rho) \\ \quad \text{for } p^{(k-1)/q+1/(3q)} \le \rho < p^{k/q-1/(3q)} \end{cases} \tag{5.7}$$

但し $k$ を整数とする

この定義により $\Psi(\rho)$ は区間的に単調増加関数となるため、推定する $\rho$ の属する区間を与えて $\rho = \Psi^{-1}(y)$ が計算できる（図 5.1）。ここで $y$ は、$m^B(\rho)/m^G(\rho)$、$m^G(\rho)/m^R(\rho)$、$m^R(\rho)/m^B(\rho)$ のいずれかとし、与えられた区間に応じて選択されるものとする。

しかるに $\rho$ の概算推定値 $\rho'$ はスペクトル解析により既に与えられている。従って推定する $\rho$ の属する区間は、この概算推定値 $\rho'$ を用いて決定することができる。ひとたび区間が決定されれば、例えば二分検索法などを用いてピーク比の一致する点を容易に検索することができる。

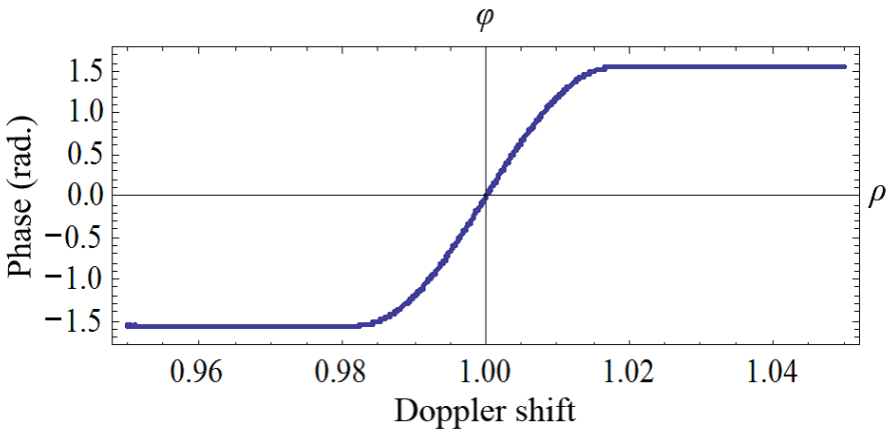


図 5.2　カラードップラー法包絡線位相関数

## 5.2　既存手法によるドップラー速度推定

本章で行うシミュレーションでは既存手法であるカラードップラー法についてもイメージングで得られた色情報からドップラー量を計算する方法を与える必要がある。これについて明示的に次にアルゴリズムを記載する。カラードップラー法で得られる色情報とは反射体速度に応じてシフトする包絡線位相 $\phi$ である。ここではこれを用いてドップラー量 $\rho$ を計算する方法を与える。

例えば、カラードップラー法で用いるパルス波を $g(t) = \exp(j\omega_1 t)\ \exp(j\omega_0 t)$ とすると、理論的には包絡線位相は $(\rho - 1)\, T_1\omega_1$ と検出される。ところが実際にはパルス波の両端がスペクトルに及ぼす影響と実装されたヒルベルト変換器の持つ周波数特性の影響で系統的な誤差が生じる。

この問題を回避するために、我々はあらかじめノイズのない設定で計算を行い辞書を作成した。辞書 $\Phi$ は、ドップラー量 $\rho$ に位相 $\phi$ を対応させる関数である。(図 5.2)。この関数 $\Phi$ は、エンベロップ関数が値を持つ区間において単調増加な連続関数であり、この区間における逆関数の存在により辞書の逆引きが可能となる。これにより検出された位相からドップラー量 $\rho = \Phi^{-1}(\phi)$ を推定する。

一般的に包絡線とは複素振幅の絶対値のことであり、その複素化である複素包絡線の位相の範囲は $[-\pi/2, \pi/2]$ に限定される。また、カラードップラー法で行う自己相関計算において、パルスの信号長を超えてドップラーシフトの影響があるときには、エンベロップ関数の自己相関関数は 0 となり、位相の検出ができなくなる。そのためドップラー検出範囲は自己相関関数が値を持つ範囲に限定される。

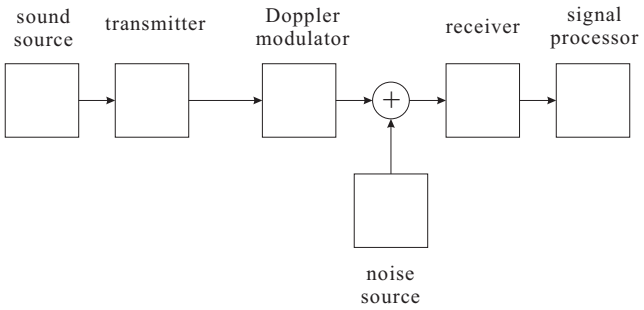


図 5.3　シミュレーションで用いた信号とノイズの付加方法

## 5.3　シミュレーション方法

提案アルゴリズムの有効性を確かめるために、コンピュータシミュレーションによる検討を行った。

カラードップラー法と提案手法の比較を行うために、ログステップマルチキャリア波の波形長 $2T$ と一致するパルス間隔 $T_1$ を持つパルス波を用意し、カラードップラー法用の送信波として使用した。パルス波の振幅は、最大振幅を一致させたものと、パルス波 1 波あたりの平均パワーをログステップマルチキャリア波の平均パワーと一致させたものの 2 種類を用意した。

カラードップラー法のパルス波をキャリア周波数 42535Hz、包絡線周波数 5317Hz とし、パルス間隔 $4000\mu s$ とした。提案手法のログステップマルチキャリア波を基本周波数 28600Hz、帯域幅 $p = 2$、キャリア数 $q = 54$、整列位相オフセット $T_0 = 3000\mu s$、波形長 $2T = 4000\mu s$ とした。

シミュレーションは次の 3 つの構成に大別して行った。

1. カラードップラー法と提案手法のそれぞれについて、ドップラー量推定範囲を検証する。
2. カラードップラー法と提案手法のそれぞれについて、ドップラー量推定精度を比較する。
3. SNR を変化させてカラードップラー法と提案手法のそれぞれについて狭いドップラー推定範囲でのノイズ耐性を検証する。但し広いドップラー量推定範囲でのノイズ耐性については次章の課題とする。

まず、(1) 推定範囲の検証では、ドップラー量を $0.95 < \rho < 1.05$ の範囲に設定し、SNR を 13dB に設定し、1000 回の試行を行った。なおこの場合、カラードップラー

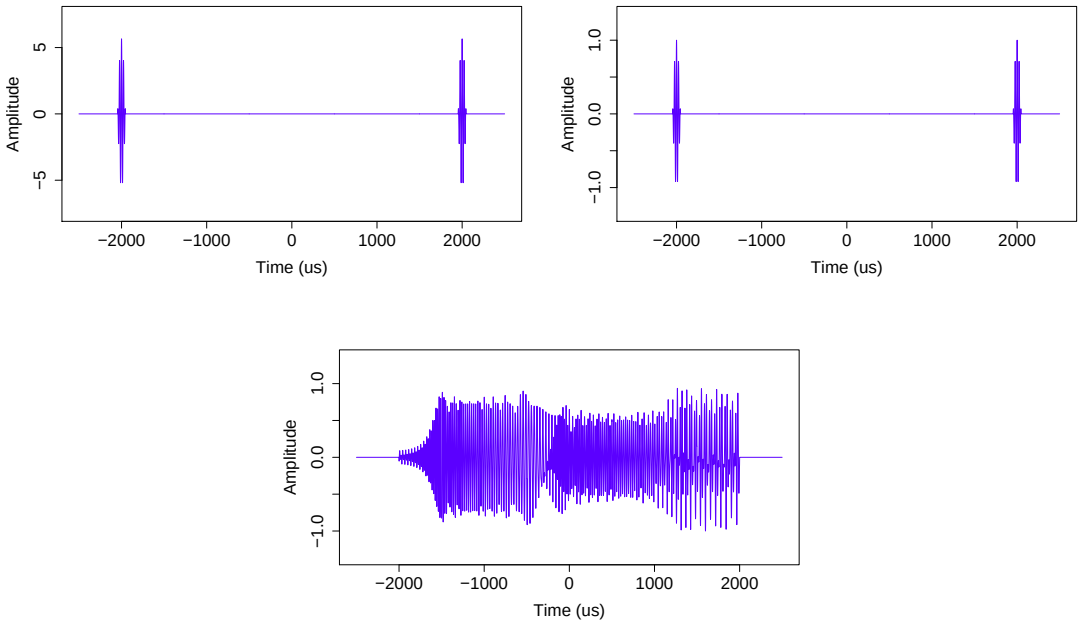


図 5.4　カラードップラー法で使用したパルス波 (同一平均パワー、同一最大振幅) と提案手法で使用したログステップマルチキャリア波

法のパルス波には、同一最大振幅のものを使用した。(2) 推定精度の比較では、ドップラー量を $0.99 < \rho < 1.01$ の範囲に設定した。この範囲にドップラー量を限定した理由は、カラードップラー法で無理なく推定が可能な範囲にあわせる必要性があったからである。SNR は提案手法を基準として 13dB に設定し、1000 回の試行を行った。この場合のカラードップラー法のパルス波には、同一最大振幅のものと同一平均パワーのものの両方を使用した。最後に (3) 狭いドップラー推定範囲でのノイズ耐性の比較では、ドップラー量を $0.998 < \rho < 1.002$ に設定し、SNR を提案手法を基準として 0dB から 25dB まで変化させて、それぞれの SNR に対し 100 回ずつの試行を行った。ドップラー量をこの範囲に限定した理由は、提案手法で低 SNR 時にスペクトル解析による概算推定が機能しなくなるためである。そのため、$\rho = 1$ のごく限られた周辺にドップラー量を限定し、提案手法では精密推定のみを行うようにした。この場合のカラードップラー法のパルス波には、同一最大振幅のものと同一平均パワーのものの両方を使用した。

カラードップラー法では更に、低域透過フィルターの適用による推定精度の改善が期待される。両極端の結果を得るために、カラードップラー法でパルス波により振幅の大きい同一平均パワーのものを用いた場合には、包絡線検波の後に低域透過フィルターを適用することにした。

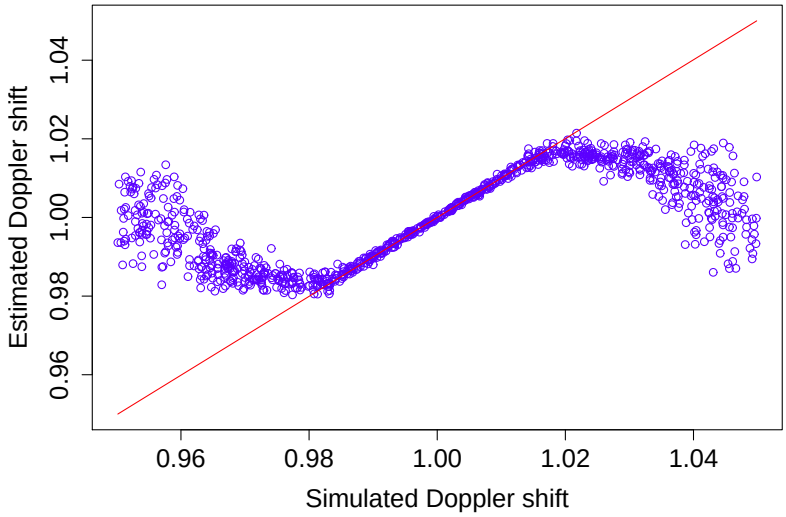


図 5.5　カラードップラー法による推定範囲の検証における推定結果

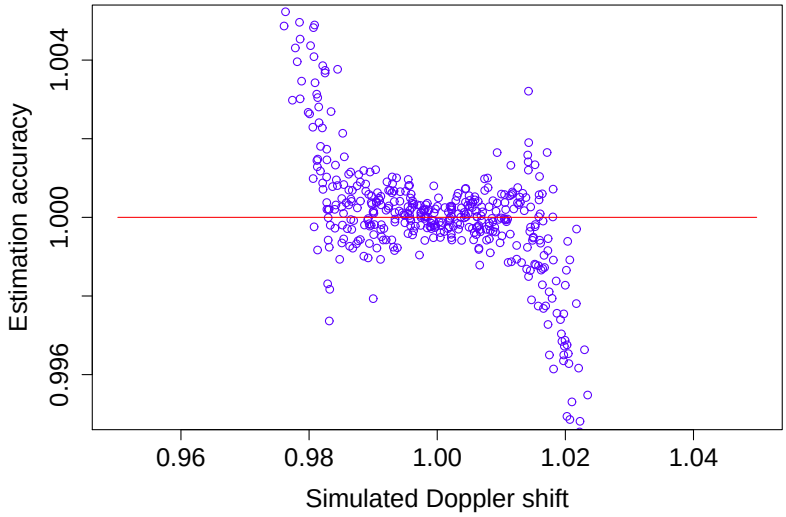


図 5.6　カラードップラー法による推定範囲の検証における推定精度

## 5.4　シミュレーション結果

最初に (1) 推定範囲の検証におけるシミュレーション結果を、カラードップラー法（図 5.5)、提案手法（図 5.7）の順に示す。

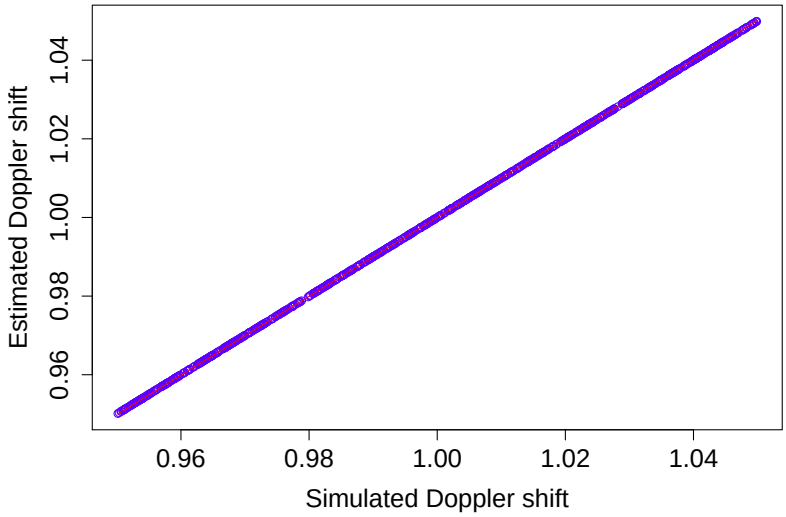


図 5.7　提案手法による推定範囲の検証における推定結果

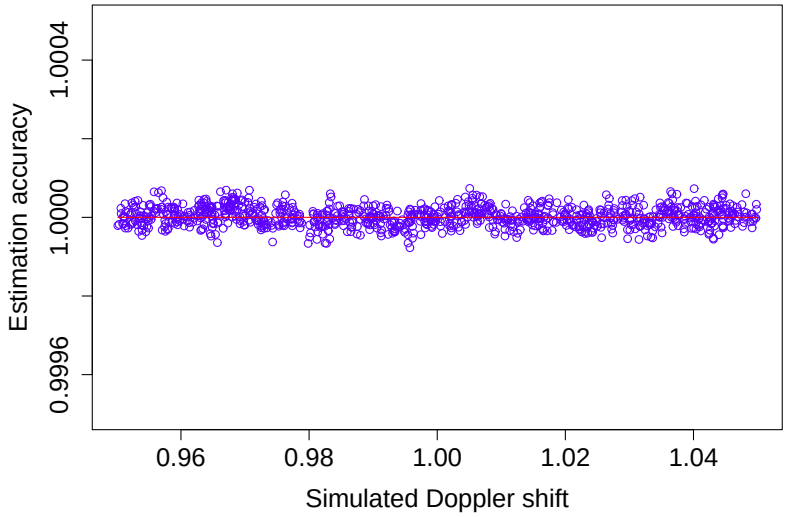


図 5.8　提案手法による推定範囲の検証における推定精度

推定精度とは、推定ドップラー量と設定ドップラー量の比とし、カラードップラー法（図 5.6、提案手法（図 5.8）の順に示す。

カラードップラー法でドップラーシフト量が 1 より大きくはずれる場合に推定精度が著しく悪化しているのに対し、提案手法ではそれが見られず、広い範囲で高いドッ

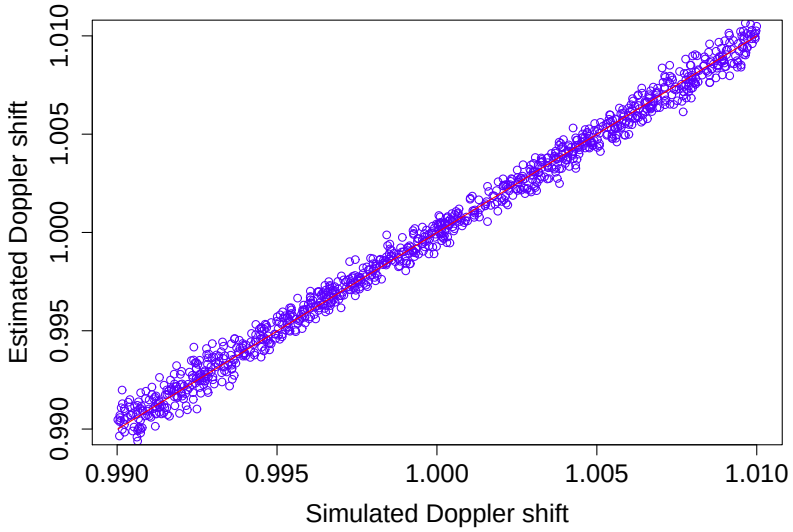


図 5.9　カラードップラー法（同一最大振幅）による推定精度の比較における推定結果

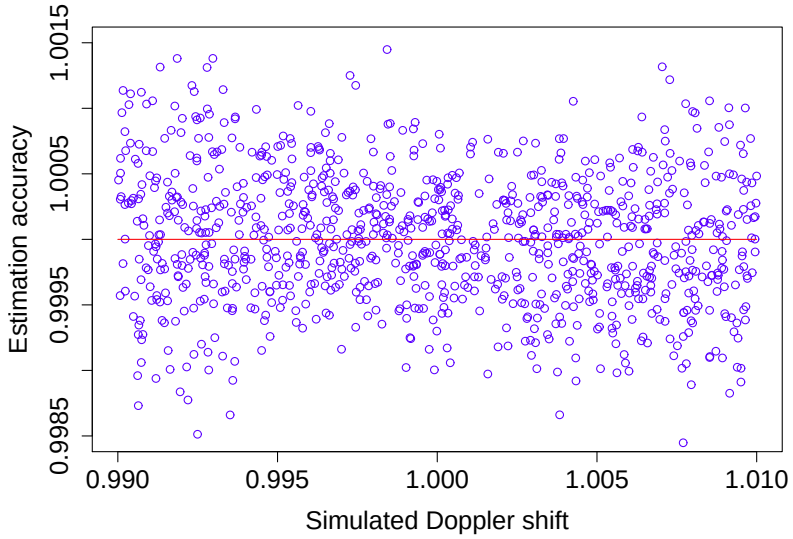


図 5.10　カラードップラー法（同一最大振幅）による推定精度の比較における推定精度

プラー推定精度を実現できていることが確認できる。

次に (2) 推定精度の比較におけるシミュレーション結果を、カラードップラー法の同一最大振幅のパルス波による推定結果（図 5.9）、同推定精度（図 5.10）、カラードップラー法の同一平均パワーのパルス波による推定結果（図 5.12）、同推定精度（図

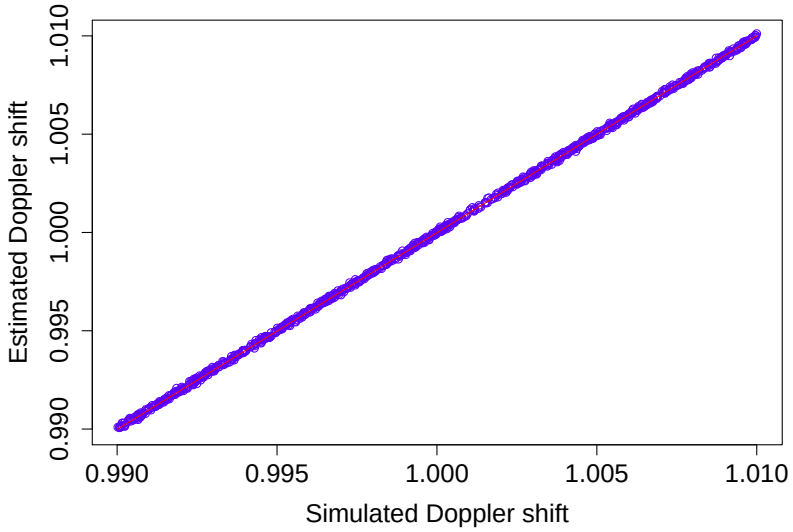


図 5.11　カラードップラー法（同一平均パワー）による推定精度の比較における推定結果

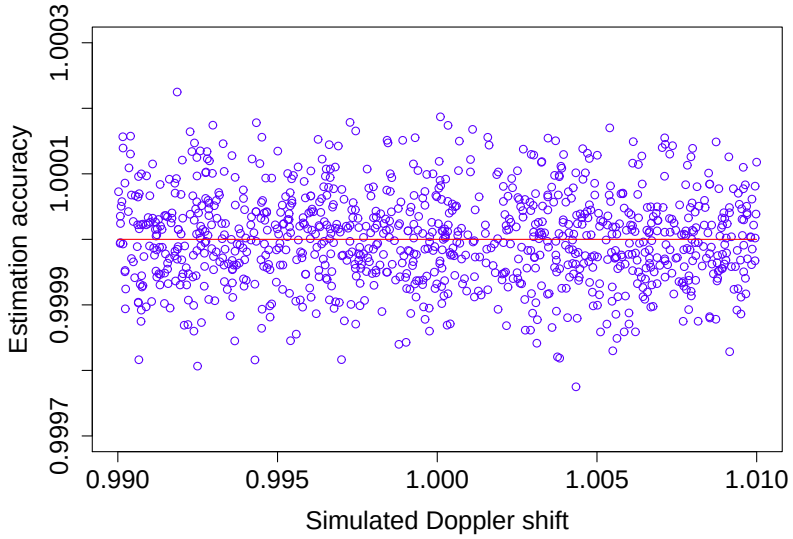


図 5.12　カラードップラー法（同一平均パワー）による推定精度の比較における推定精度

5.12)、提案手法の推定結果（図 5.13)、同推定精度（図 5.14）の順に示す。

このドップラー量推定範囲では、カラードップラー法と提案手法の双方において、推定精度のドップラー量による影響はなく良好であることが確認できる。カラードップラー法と提案手法で推定精度を比較すると、カラードップラー法に同

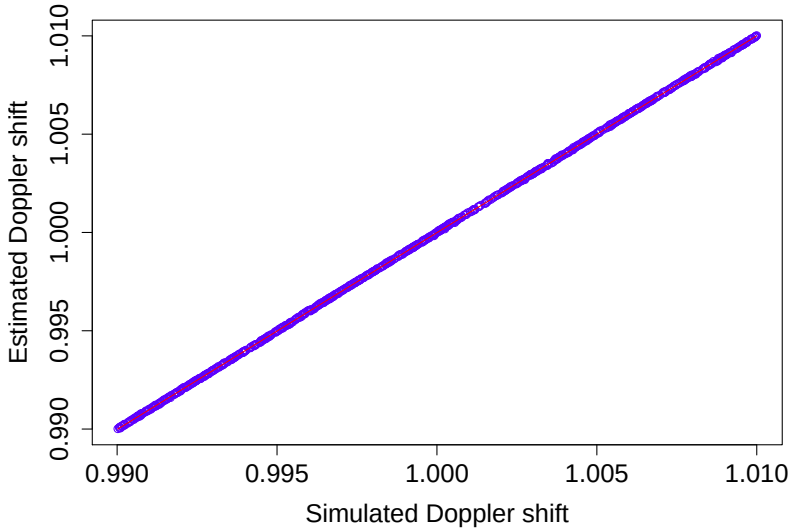


図 5.13　提案手法による推定精度の比較における推定結果

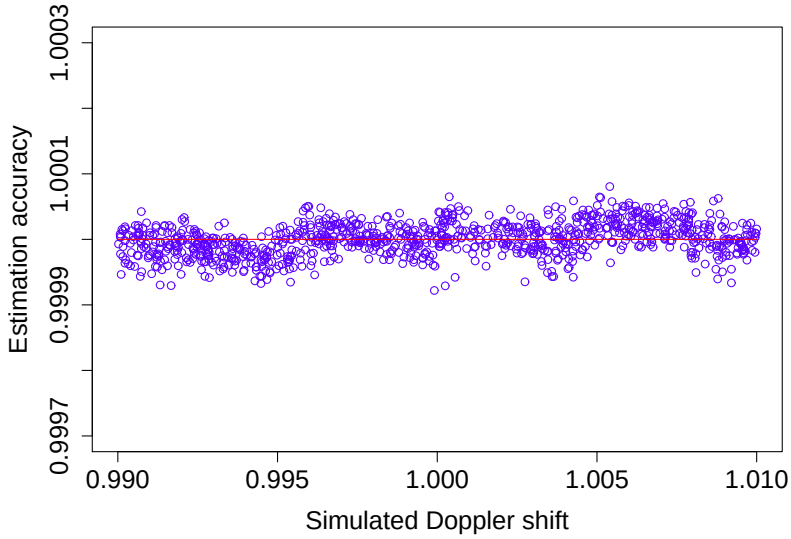


図 5.14　提案手法による推定精度の比較における推定精度

一最大振幅のパルス波を用いた場合で、提案手法はカラードップラー法に対して $20.8 = (5.04 \times 10^{-4})/(2.51 \times 10^{-5})$ 倍の推定精度を示し、カラードップラー法に同一平均パワーのパルス波を用いた場合で、提案手法はカラードップラー法に対して $2.80 = (7.04 \times 10^{-5})/(2.51 \times 10^{-5})$ 倍の推定精度を示した。

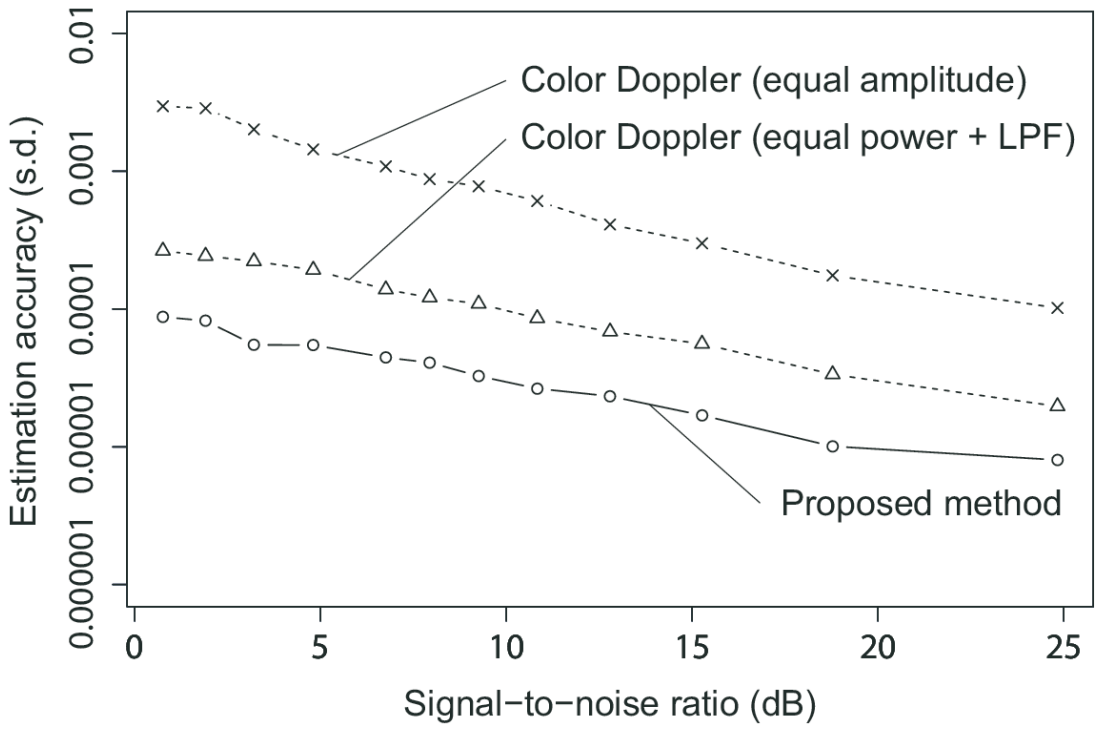


図 5.15　SNR の変化によるカラードップラー法と提案手法のドップラー量推定精度の比較

最後に SNR 変化によるノイズ耐性の比較結果（図 5.15）を示す。この結果においては、設定された SNR の全域にわたり、提案手法の優位性が確認できる。提案手法は同一最大振幅のパルス波を用いたカラードップラー法と比較した場合に 12.69〜36.68 倍、同一平均パワーのパルス波を用いたカラードップラー法と比較した場合に 2.54〜4.04 倍の推定精度を示した。

## 5.5　考察

本章のシミュレーションによって、提案手法はカラードップラー法に対して広いドップラー量推定範囲を持つことが示された。これはカラードップラー法のドップラー量推定範囲が用いるパルス波の性質、特に包絡線の幅とパルス間隔による制約を受けるのに対して提案手法にはその制約がなく、帯域幅というより自由に設定可能なパラメーターにより制御されることが関係している。

推定可能域における推定精度については、提案手法はカラードップラー法に対して、同一最大振幅の送信波を用いた場合で約 3 倍、同一平均パワーの送信波を用いた場合にも約 3 倍の性能を持つことが示された。この理由としては、ドップラー量の検出のために、ノイズの含まれた受信波の自己相関計算を用いるカラードップラー法よりも、

受信波とノイズの含まれない参照波の相互相関計算を用いる提案手法のほうがよりノイズの影響を除去し易いためだと考えられる。

更に、ドップラー量が少ない場合には SNR 悪条件下でも提案手法がカラードップラー法に対して優位性を保つことが示された。これは提案手法により得られるイメージング結果の色情報が持つ曖昧性の解消を行う必要がない場合の結果である。色情報が曖昧性を持つ場合には、概算推定アルゴリズムの改善が課題となり、次章でそれを扱う。

なお、本章で示したドップラー速度推定結果は、[MSH12a] [MSH12b] [MSH14] において発表したものである。

# 第 6 章

# 対称ログステップマルチキャリア波を用いた速度推定法の改善

第 3 章において、ログステップマルチキャリア波から相互相関計算でピークを再構成すると、ピーク時刻にドップラーシフトに応じた系統偏移が生じることを示した。ピーク時刻系統偏移は、イメージングの際には距離誤差として現れるため、一般には除去することが望まれるが、逆にこれをうまく利用することによって、系統偏移量からドップラー量を逆算することができる。

本章では、偏移傾向の異なる 2 つのログステップマルチキャリア波を連接して送信することによって、相互相関計算によって生じるピーク時刻の間にドップラー量に応じた偏移差が得られ、これを用いたノイズ環境においても頑健なドップラー速度推定アルゴリズムが構成できることを示す。このアルゴリズムにおいては、得られた偏移差を用いてドップラー量の概算推定値が得られ、その推定値を前章で行った精密推定の区間指定に用いることによって相互相関ピーク比に基づくドップラー量の精密推定値を得る。

こうして得られた複合アルゴリズムによって、SNR が 0dB を下回るような過酷なノイズ環境下においても、周波数偏移を知る事ができ、結果として頑健性の高いドップラー量推定アルゴリズムを構成することができる。これは前章で紹介した高精度のドップラー量推定アルゴリズムの耐ノイズ性における弱点を補うことができるため、これを補完し、推定精度、耐ノイズ性のいずれの点においても高性能のドップラー速度推定手法を提供する。本章では、この複合アルゴリズムの提案を行い、その性能をシミュレーションにより検証する。

## 6.1　対称波によるドップラー速度推定法

ログステップマルチキャリアの特徴の一つに、相互相関計算によって得られる到来時刻に系統偏移が現れる性質があった。系統偏移の大きさはログステップマルチキャリア波のキャリア位相を調整する初期時刻 $T_0$ で決まる。この値を絶対値の等しい正負の値に設定することにより、系統偏移の傾向が逆の 2 つのログステップマルチキャリア波が得られる。

すなわち $T_0 > 0$ の場合には、ドップラーシフトで周波数が高くなる時には相互相関計算から得られるピーク時刻は遅れ、周波数を低くなる時にはピーク時刻は早まるのに対し、$T_0 < 0$ の場合には、周波数が高くなる時にはピーク時刻は早まり、周波数が低くなる時にはピーク時刻は遅れるのである。

こうして初期時刻が $T_0$ と $-T_0$ の 2 つのログステップマルチキャリア波を前後に並べると、原点において左右対称の波形が得られる。この波形を送信波として用いると、ドップラーシフトによって対称波形全体が短くなるときには、前半部から得られるピークは遅れ、後半部から得られるピークは早まるため、ドップラーシフトによる波形全体の伸縮を強調する方向に系統偏移の影響が現れる事になる。そこで、対称波形の前半部と後半部で、それぞれ別々の相関演算を実施し、その結果得られたピーク間隔からドップラー量を逆算するのが本章で提案する手法である。

相互相関計算を実施するために、対称波形の前半部と後半部でそれぞれ別々の参照波を用意し、エポックを波形中心に設定しておく。整列位相ログステップマルチキャリア波においてエポックとは各キャリアの位相が $T_0$ によって決定される時点であるため、このエポックの設定には本質的な意味がある。2 つの参照波におけるエポック間の間隔を $T_I$ と設定する。

この 2 つの参照波による相関ピークの計算は、次のように定式化することができる。

$W(t) = u(t) + u(t)$ を送信波とする。ここで、$u(t)$ 及び $v(t)$ は次のように定める。

$$u(t) = r_{[-T_I/2-T,-T_I/2+T]}(t) \cdot g_{p,q}(t + T_I/2, T_0) \tag{6.1}$$

$$v(t) = r_{[T_I/2-T,T_I/2+T]}(t) \cdot g_{p,q}(-t - T_I/2, -T_0) \tag{6.2}$$

すなわち $W(t)$ を原点を中心に繋ぎあわされた左右対称の整列位相ログステップマルチキャリア波とする。定義式の中に現れる $r_{[A,B]}(t)$ は、区間 $[A, B]$ で値を持つ矩形窓関数である。

2 つの参照波 $U(t)$ 及び $V(t)$ は、$u(t)$ 及び $v(t)$ の平行移動とし、$u(t)$ 及び $v(t)$ と

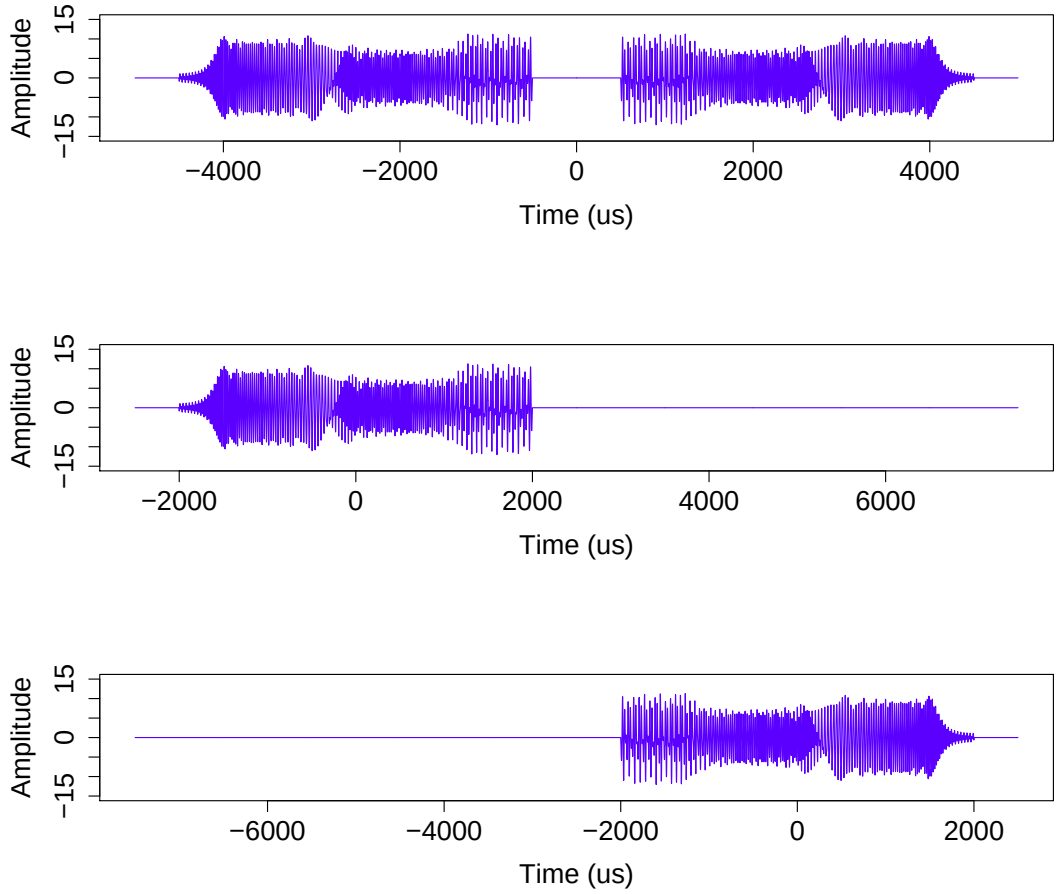


図 6.1　概算推定に使うログステップマルチキャリア波（$T_0 = 3000, -3000, T_I = 5000, T = 2000$ (us)）上：送信波 $W(t)$、中：参照波 $U(t)$、下：参照波 $V(t)$

はエポックのみが異なるものとする（図 6.1）。

$$U(t) = r_{[-T,T]}(t) \cdot g_{p,q}(t, T_0) \tag{6.3}$$
$$V(t) = r_{[-T,T]}(t) \cdot g_{p,q}(-t, -T_0) \tag{6.4}$$

前半部 $U(t)$ を参照波とする相互相関及び後半部 $V(t)$ を参照波とする相互相関値は、次式により計算される。

$$\langle U|\mathfrak{D}_\rho W\rangle(t) = \int_{-\infty}^{\infty} \overline{U}(\tau - t) \cdot \sqrt{\rho} W(\rho\tau) d\tau \tag{6.5}$$
$$\langle V|\mathfrak{D}_\rho W\rangle(t) = \int_{-\infty}^{\infty} \overline{V}(\tau - t) \cdot \sqrt{\rho} W(\rho\tau) d\tau \tag{6.6}$$

これらの相互相関ピーク時刻はログステップマルチキャリア波の性質により異なる傾

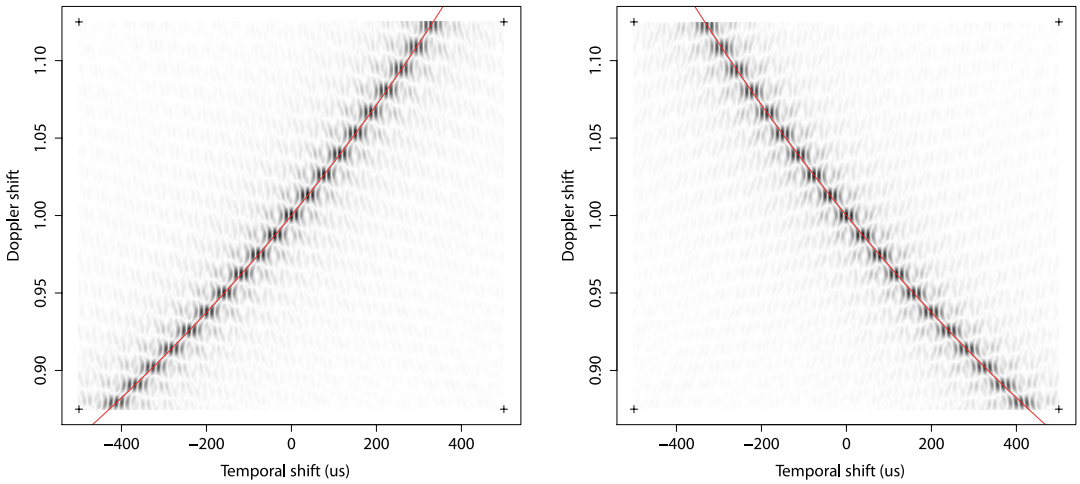


図 6.2　対称ログステップマルチキャリア波の前半部と後半部から得られる相互相関ピーク

向の系統偏移を示す（図 6.2)。前半部相互相関ピーク時刻 $t_U$ と後半部相互相関ピーク時刻 $t_V$ はそれぞれ、次式で求められる。

$$t_U = \arg\max_t \{\|\langle U|\mathfrak{D}_\rho W\rangle(t)\|\} \tag{6.7}$$

$$t_V = \arg\max_t \{\|\langle V|\mathfrak{D}_\rho W\rangle(t)\|\} \tag{6.8}$$

このときピーク時刻偏移 $s$ は、ドップラーシフトがないときのピーク間隔 $T_I$ からの変化とし、$s = t_V - t_U - T_I$ と定義する。

このとき、次の定理が成立する。

**［定理］ 5** $T_0$ を整列位相ログステップマルチキャリア波の初期位相、$T_I$ を対称波のエポック間隔とする。このとき、ドップラー量 $\rho$ は観測されたピーク間隔 $s$ から次式により計算できる。

$$\rho = \frac{T_I + 2T_0}{s + T_I + 2T_0} \tag{6.9}$$

証明

受信波はドップラーシフトの影響により、$\sqrt{\rho}W(\rho t)$ と変化する。この場合にエポック間隔も $1/\rho$ 倍に変化するため、その間隔変化を $t'$ とすると、$t' = T_I\left(\frac{1}{\rho} - 1\right)$ となる。

相互相関ピークの間隔変化には、エポック間隔の変化に整列位相ログステップマルチキャリア波のピーク時刻系統偏移による影響が加わる。前半部、後半部でそれぞれ

$t''$ だけ系統偏移が現れるとすると、全体でピーク間隔に対して $-2t'' = 2T_0\left(\frac{1}{\rho} - 1\right)$ だけの寄与が現れる。

受信波の伸縮の影響と系統偏移による影響が足し合わされた結果、

$$s = t' - 2t'' = (T_I + 2T_0)\left(\frac{1}{\rho} - 1\right) \tag{6.10}$$

なるピーク間偏移が起こる。これを $\rho$ について解いて、与式を得る。

□

こうして得られた $\rho$ を概算推定値として用い、これを前章で提案した精密推定の区分的単調増加の区間を指定するために用いて最終的なドップラー量を求める。

ここで系統偏移から得られる推定量は、ログステップマルチキャリア波の性質から、離散的なものになるということに注意しておく。ログステップマルチキャリア波のキャリアは、キャリア間干渉が十分に小さくなる程度に離散的に配置されており、送信波と受信波のキャリア間の周波数の一致が階段状のドップラー相関を示すからである。この状況は、連続関数によって与えられる理論式との違いを示しているが、この不一致は、系統偏移ピークの理論式が、実は離散的な点を結んで得られた概算式によって与えられているために現れる。いっぽう信号長の伸縮は連続的に起こるため、これに起因するピーク間隔の偏移は連続的なものになる。

離散量と連続量が足し合わされた結果、区分的に連続な関数が得られる。関数全体の不連続性は前章で得た区分的単調増加関数の推定区間を指定するのに都合が良く、結果的に効率的なアルゴリズムの構成に役に立つ。

## 6.2 シミュレーション方法

提案アルゴリズムを検証するためにシミュレーションを行った。整列位相ログステップマルチキャリア波のパラメータを、$p = 2$, $q = 54$, $f_0 = 28600(Hz)$, $T_0 = 3000$ (us) と定める。信号長は　$T = 4000$ (us) とし、信号区間を $[-2000, 2000]$ (us) に設定した。

１つめのシミュレーションでは、ドップラーシフトを伴った波形に対し、おおよそ SNR $= 0$dB のノイズを加え、最初に本章で提案したアルゴリズムを適用して概算推定値を求め、それを前章で提案したアルゴリズムの区間推定に利用して精密推定値を求めた。本シミュレーションではドップラー量を $0.95 < \rho < 1.05$ まで変化させて、全 1000 回の試行を行った。

２つめのシミュレーションでは、アルゴリズムのノイズ耐性を調べるために、ドッ

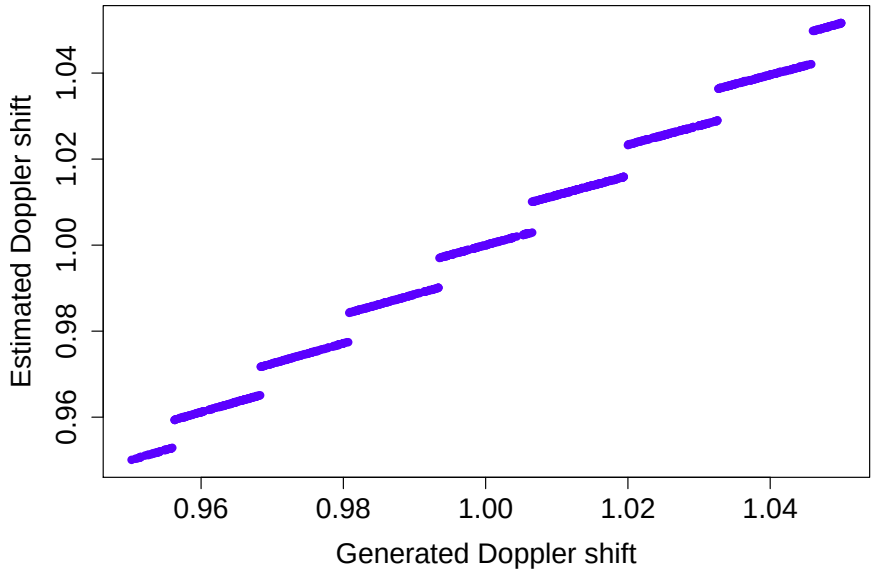


図 6.3　ドップラー概算量の計算

プラー量 $0.95 < \rho < 1.05$ の範囲において、SNR を $-12.75$dB から 21.80dB まで変化させて有効帯域内にホワイトノイズを付加した。試行回数は各 SNR に対してそれぞれ 100 回とした。

## 6.3　シミュレーション結果

1 つめのシミュレーションでは、ドップラー概算量の推定結果 (図 6.3) を見ると、ドップラー量概算量が、2 つの要因により決定されているということが確認できる。一つは受信波の伸縮によるピーク間隔の偏移であり、設定ドップラー量に比例した傾向を推定ドップラー量にもたらしている。もう一つはピーク位置の系統偏移による要因であり、ドップラー量がキャリア間隔にかかる間は一定であるが、キャリア間隔をまたぐ毎に突然不連続に偏移する。これら 2 つの要因がお互いを強調する方向に足しあわされた結果、グラフは単調増加となり、不連続点を境界に用いることによってドップラー量の離散的な概算推定が可能になった。

ドップラー量概算推定結果を前章で与えたドップラー量詳細推定アルゴリズムに与えて、精密なドップラー量の推定を行う。試行回数 1000 回、SNR 比を 0.2297dB としたときの複合的アルゴリズムのドップラー量推定結果 (図 6.4) 及び推定精度 (図 6.5) を示す。結果を見ると、設定したドップラー量に等しい推定結果が得られていること

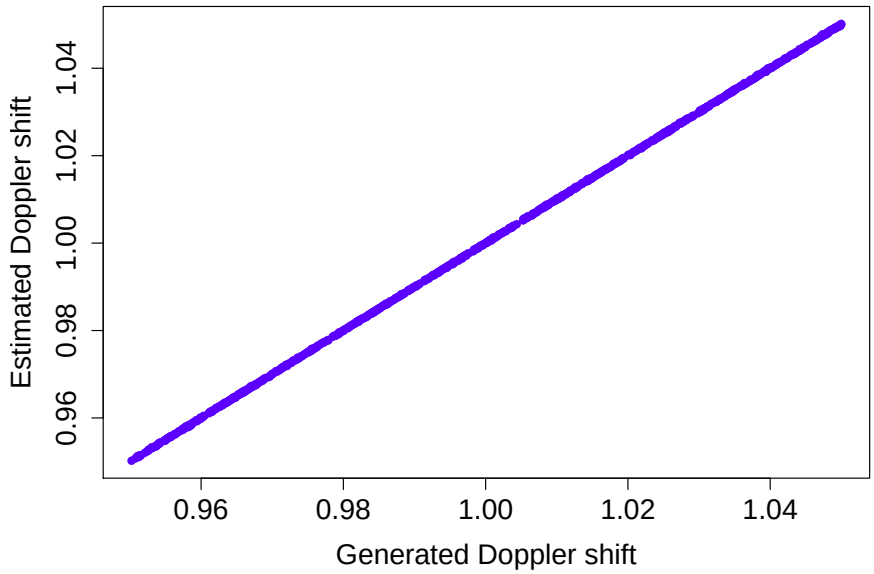


図 6.4　複合アルゴリズムによるドップラー量推定結果

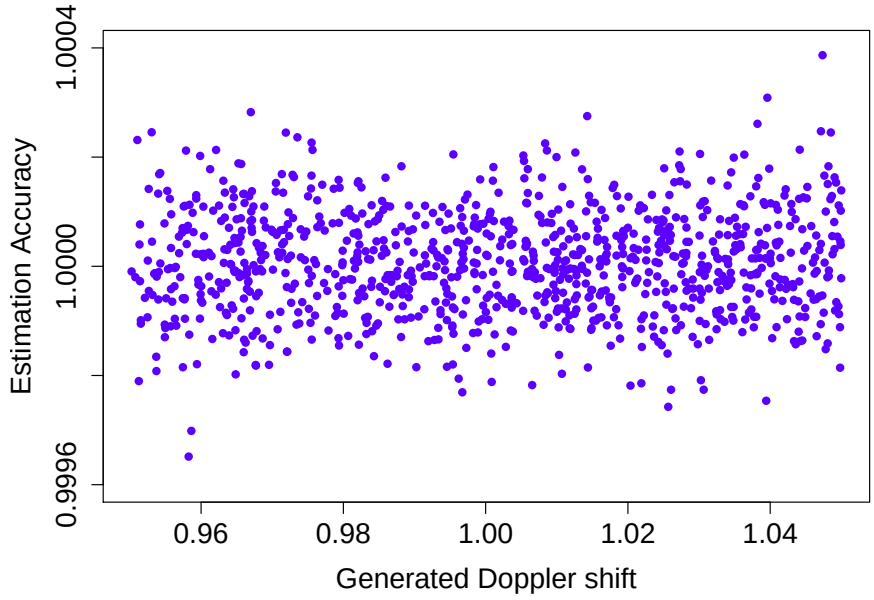


図 6.5　複合アルゴリズムによるドップラー量推定精度

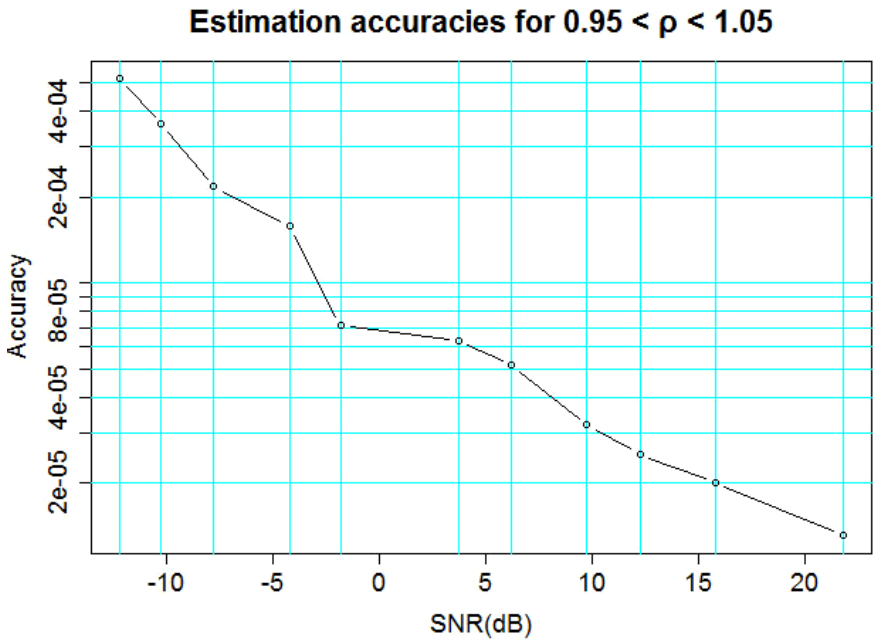


図 6.6　改良されたノイズ耐性

が確認できる。設定したドップラー量の全域にわたり、複合アルゴリズムが低 SNR 条件下においても機能していることが確かめられた。

２つめのシミュレーション結果として、図 6.6 に各ノイズ量に対してそれぞれ 100 回の試行を行った結果を示す。SNR が 0 以下となりノイズ量が増加しても、依然として推定精度は SNR が良好な場合と同様の傾向を示し、複合アルゴリズムが正常に機能していることが確認できた。

## 6.4　考察

本章で提案した対称ログステップマルチキャリア波を送信波に用いる方法によって、難しいスペクトルの周波数解析の問題を、より簡単な時間領域でのピーク間隔の検出の問題に置き換えることによって、低い SNR 条件下でも有効に機能するドップラー量概算推定アルゴリズムが構築できた。

このアルゴリズムを前章で提案したドップラー良精密推定と併用して複合アルゴリズムが得られ、SNR が −10dB 以下の悪条件下においても、精密なドップラー量推定を行う事ができる SNR 頑健性が示された。この複合アルゴリズムによって、ログステップマルチキャリア波を用いたドップラー速度推定の最大の弱点であったノイズ耐性の問題が解決された。

本章で述べた成果は以下の会議で発表した。[MSH13b] [MSH13a]

# 第 7 章

# 超音波イメージング装置

超音波イメージングシステムでは、ビームフォーマーにより方向成分を抽出するため正確に配置された多数のマイクロフォンと、センサーから送られるデータを一度にサンプリングするための機能と、大量のサンプリングデータをリアルタイム処理するための装置が必要になる。

本研究においても、大量のデータを高速でリアルタイム処理する必要があり、その要求は既存の PC ハードウエアの性能を上回るものであった。そのため実験装置を設計から製作までを行う必要があり、近年大幅に集積度と性能向上が見られる MEMS (Micro Electro Mechanical System) 及び FPGA (Field Programmable Gate Array) はこの目的に合致した。超音波 MEMS マイクロフォンは形状と大きさが超音波イメージングに必要とされるマイクロフォンアレイを高密度、高精度に実装するのに都合が良く、ユーザー定義可能なピン配置を持ち容易にプログラミングできる FPGA は、設計段階からの実験装置の試作のために適切なデバイスであった。

本章では、これらのデバイスを用いて製作した超音波イメージング装置について、その概要と構成を記述する。

## 7.1 超音波イメージングシステムの全体構成

本研究で製作した超音波イメージングシステムの中核となるものは多数のマイクロフォンにより反射波を受信し、一定時間に得られる受信信号をサンプリングしてデジタル信号に変換して蓄積する信号処理回路である。超音波ドップラーイメージングのためには、このほかに信号を発生して送信する送信回路と、反射波にドップラー効果を発生させるために反射体を一定速度で運動させるローターとその制御回路、受信機、送信器、ローターの間のタイミングの制御を行うトリガー制御回路が必要となる。

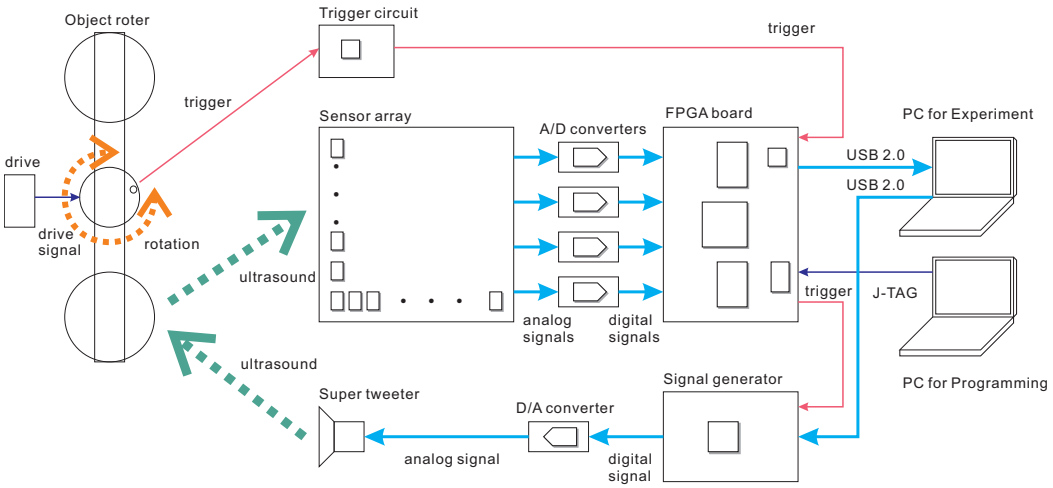


図 7.1　システムの全体構成

図 7.1 に本超音波イメージングシステムの全体構成を示す。ローターは PWM 制御により一定速度で回転するが、ローター中央部に設けられた円盤の切欠きフォトインタラプタで検知してトリガーを発生する。トリガーは信号処理装置に送られマイクロフォンアレイから送られる受信信号のサンプリングを開始すると同時に、トリガーを信号発生器に送る。トリガーを受けた信号発生器はあらかじめ用意された波形を発生させ、アナログ信号に変換した後、スーパーツイーターにより空中に超音波を送信する。

サンプリングは 128ms の間だけ行われ、空中を伝搬して伝わる超音波を受信する。このサンプリング時間によって、原理的には送信機、受信機から反射体までの距離が約 2m までのイメージングが可能である。

## 7.2　マイクロフォンアレイ

マイクロフォンアレイは多数のマイクロフォンから成る超音波受信装置であり、マイクロフォンの間隔により視野角が、開口により解像度が決定される。本装置は MEMS マイクロフォンを横 5.08mm、縦 10.16mm の間隔で、16 列 8 行に並べ、横 76.2mm、縦 71.12mm の開口が得られるように配置した（図 A.2）。これにより、横方向に 113.56 度、縦方向に 49.46 度の視野角が得られ、縦横共に約 7.5 度の分解能が得られた。

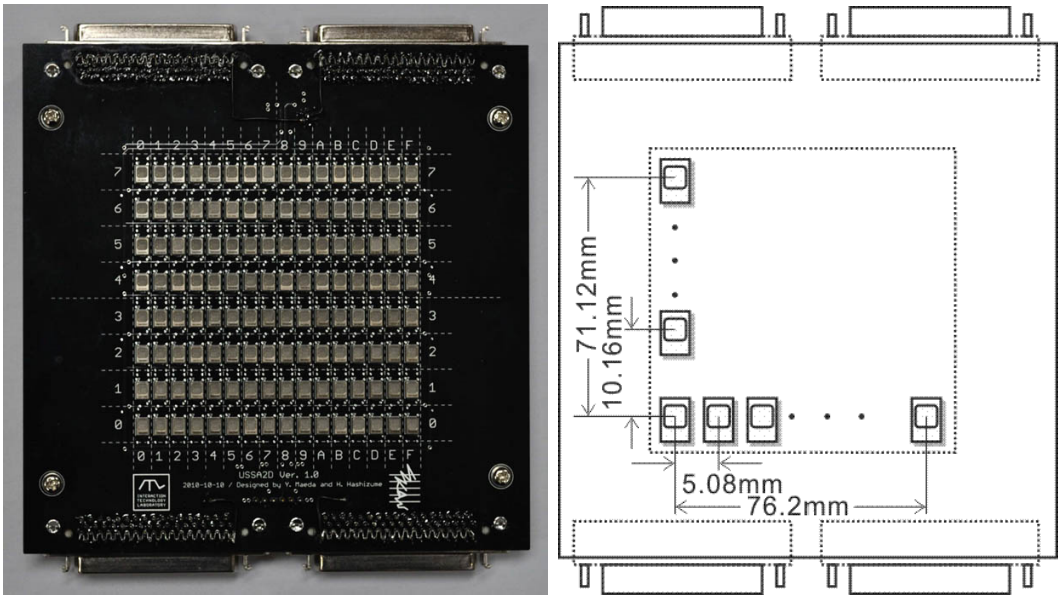


図 7.2　MEMS 超音波センサを用いたマイクロフォンアレイ・間隔と開口

## 7.3　信号処理装置

信号処理装置は多数の AD コンバーターと周辺回路、処理機能を実装する FPGA と、信号を蓄積する RAM 及び周辺回路により構成されるアナログ・デジタル混載回路である。サンプリング時間とサンプリング周波数はこの装置の性能によって決まり、それぞれイメージングの到来方向の検出可能距離と、イメージングの時間解像度を決定する。本装置は、サンプリング時間 12.8ms、サンプリング周波数 500ksps を実現した。

## 7.4　考察

本装置を用いて、従来は困難であった広帯域空中超音波による実機実験が可能となり、本論文の第 4 章で述べたログステップマルチキャリア波を用いたドップラーイメージング実験によって装置の実用性が検証された。得られた装置は 128 個の MEMS マイクロフォンにより構成されるセンサーアレイからの信号を、サンプリング周波数 500ksps、サンプリング期間 12.8ms にわたって受信することができた。

いっぽう信号処理の観点からは、本装置は改良の余地があり、リアルタイムでのビームフォーマーやマッチドフィルターの適用には課題が残る。システムクロック等の動作速度の向上と、FPGA における素子の集積と回路の工夫が今後の課題である。

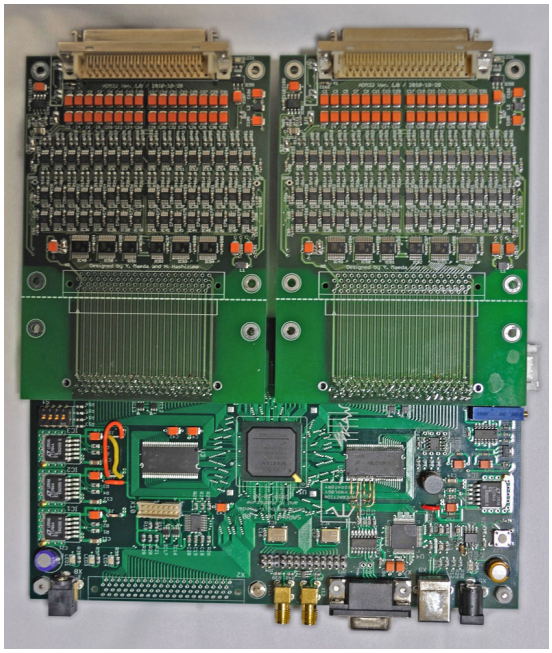

図 7.3　FPGA を用いた信号処理回路

本装置および本装置を用いた実験結果は [MSH11a][MSH11b][MSH11c] において発表したものである。

なお、本装置の実装の詳細を、付録 A 及び 付録 B に記載する。

# 第 8 章

# 結語

## 8.1 結論

本論文では、超音波によるドップラーイメージングとドップラー効果を利用した速度推定法の精度改善の問題を扱い、送信波のデザインによって反射体イメージングと速度推定精度の改善が可能であることを示した。特に新たに考案した広帯域信号であるログステップマルチキャリア波を定式化し、その性質について解明した。これにより同波は広いドップラー耐性とドップラー感受性を併せ持つことが示され、その性質を利用したドップラーイメージング法及び速度推定法についてアルゴリズムを与えた。提案アルゴリズムに対してイメージング実験及び速度推定シミュレーションを行い、その有効性を検証した。

第 2 章では、広くドップラーイメージングについて概観し、本研究の背景を紹介した。また医用超音波において一般的に用いられてきたカラードップラー法についての紹介を行い、本論文で比較対象とした既存手法についての理解を深めた。

第 3 章では、ログステップマルチキャリア波の概念を導入した。多数キャリアから成る同波の構成方法を示し、特に所定の時刻において一斉に各キャリアの位相が揃う整列位相について定義を与え、ログステップマルチキャリア波の特定の系列がこの整列位相を持つことを示した。続いて整列位相ログステップマルチキャリア波についてのいくつかの特性を明らかにした。受信波のドップラーシフトによって相互相関のピーク時刻に系統的な偏移が生じることを示し、その偏移量についての解明を行った。さらに相互相関ピークの値を繋げたプロファイルの概念を導入し、このプロファイルの解析によってログステップマルチキャリア波が広いドップラー範囲にわたって高い相互相関ピーク値を保つドップラー耐性と、相互相関ピークの周りで急峻にピークが減衰する感受性を併せ持つことを示した。

第 4 章では、ログステップマルチキャリア波を用いたドップラーイメージング法についてアルゴリズムを与え、その有効性を実機を用いた実験により示した。

第 5 章では、ログステップマルチキャリア波を用いたドップラー速度推定法についてアルゴリズムを与え、その性能を既存手法であるカラードップラー法との対比シミュレーションを行うことにより確認した。シミュレーションの結果として、ログステップマルチキャリア波を用いた提案手法は、ドップラー推定範囲を拡大し、従来のカラードップラー法に見られなかった高い性能を示した。提案手法は、等振幅の送信波で比較した場合に約 20 倍、等パワーの送信波で比較した場合に約 3 倍の精度向上を示した。

第 6 章では、ログステップマルチキャリア波を用いたドップラー速度推定手法を拡張し、同波の到来時刻系統偏移の性質を利用して、偏移傾向の異なる 2 波を連結して送信波として用いる事により速度推定の際に現れる曖昧性の解消が可能であることを示し、新たにアルゴリズムを考案して、シミュレーションにより有効性を検証した。

このアルゴリズムのシミュレーションでは、ノイズ耐性が劇的に改善されることが示された。これにより、従来行うことのできなかった SNR が $-10$dB を下回るような劣悪な条件下においても、ドップラー速度推定が可能となり、推定精度もノイズ量に応じた変化を見せるにとどまり、低い SNR におけるアルゴリズムの有効性が示された。

第 7 章では、本研究で用いたドップラーイメージング装置についての開発と実装方法についての紹介を行った。

最終的に、ログステップマルチキャリア波は、乗法相似性という波形の持つ特徴によって、ドップラーシフトを伴った反射波の検出に有効であり、強いドップラー耐性と精密なドップラー感受性を合わせ持つことが解明された。加えてこの波形を用いた高精度なドップラーイメージング及び速度推定アルゴリズムの性能と頑健性が示された。

## 8.2　今後の展望

本論文によって、ログステップマルチキャリア波の到来時刻系統偏移は、ドップラー速度推定には有効であることが示されたが、一方ドップラーイメージングにおいて反射体位置の検出に不利に働く。この系統偏移は同波のデザインパラメーターによってその大きさが決定するため、同波を再デザインを行うことによって影響をキャンセルすることが可能である。これをドップラー速度推定と両立させることは検討を要する課題であり、より良いアルゴリズムの開発を今後の課題としたい。

送信機と受信機の周波数特性や媒体中を伝搬する音波の伝搬特性、反射体の反射特

性の影響により、送信波のスペクトルが変化して受信される場合の提案アルゴリズムの検証と伝達関数に合わせたキャリブレーションは本論文によって扱わなかった課題である。実際の装置においてアルゴリズムを実装する時に必須になるこれらの課題についても、今後検討したい。

本研究で示した動く物体を高速高精度で検出する空中超音波を用いたドップラーイメージングの技法は、衝突防止や障害物検知のための車載システムへの応用が期待される。また下水道や車道のトンネル内を移動しながら用いることにより、建築構造物の異常検出などにも役立てることが出来る。更に高度な応用としては、人体により反射する超音波をイメージングすることにより、ジェスチャーや身振りを検出することが可能なことから高度なセキュリティーシステムに用いることが考えられる。

また、本研究では専ら反射波のみを用いたイメージングを扱ったが、小型化した発信機を動く物体に取り付けて用いることにより、より確実にイメージングを行うことができる。これにより、能動的に超音波を発して発信機の位置と速度を検出する、新しいマン・マシンインターフェースへの応用が考えられる。

本論文で提案した乗法相似性を持つ波形と相互相関計算を用いる手法は、空間超音波のみならず、広く音響イメージング及びドップラー速度推定に用いることができる。これについては今後、医用超音波、水中での動体検出などへの広範な応用が期待される。

# 謝辞

本論文は、新たに考案した広帯域信号であるログステップマルチキャリア信号を提案し本研究をまとめるにあたり、本論文の主査であり，主任指導教官でもある橋爪宏達教授には、数学、物理学および電子工学の基礎から装置の製作に至るまで、数えきれないアイデアをはじめ、丁寧なご指導をいただけましたことを感謝いたします。本研究の中心となるアイデアは、橋爪教授のご指導なしでは得ることができなかったもので、広範で多彩なアイデアをご教示頂けましたことをここにお礼申し上げます。

本論文の副査であり、ITL ラボ主催者の杉本雅則教授には、類似研究の指摘をはじめ論文の構成に至るまで、緻密で丁寧なご助言をいただきました。また共同研究の貴重なリソースをご提供くださり、東京大学のラボのメンバー同様に研究に参画する機会を与えてくださいましたことを深く感謝いたします。

副査の後藤田洋伸教授には、論文の趣旨を明瞭にするための論文の構成に関する貴重なご助言を頂き、特に序論の展開方法について実際的なご指導をいただきましたことを深く感謝いたします。

副査の小野順貴教授には、マルチキャリア波の周波数と位相の関係性につき貴重なご助言を頂き、類似する研究分野の立場から、専門的なアドバイスやプレゼンテーションの方法についてのご指導をいただきましたことを深く感謝いたします。

副査の高須淳宏教授には、論文の構成についての落ち着いたアドバイスを頂き、中でも計算機科学の立場から、アルゴリズムの計算量に関するご助言をいただきました。深く感謝する次第であります。

東京大学 ITL ラボの皆様には、数年間にわたり毎週開かれるゼミに参加させて頂き、終始積極的なお互いの進捗状況の発表と討論の機会を与えてくださったことを感謝いたします。中でも、Laokulrat Natsuda さんには、製作した装置の実証実験や応用をはじめ、共著者として一緒に研究や開発を行ってくださったことは本研究をすすめる上で貴重な励みとなりましたことを感謝いたします。今回ここに名前をあげることのなかった方々にもゼミを通して様々なアイデアを交換させていただいたことを感

謝いたします。

また、有限会社シンフォニー様には、条件の厳しい中を本研究で用いた装置の開発にご協力いただいたことを感謝いたします。

# 付録 A

# 超音波イメージング装置の製作

本章では、本研究において製作した超音波イメージング装置について、その設計と実装方法について記述する。

## A.1 ハードウエアの設計と実装

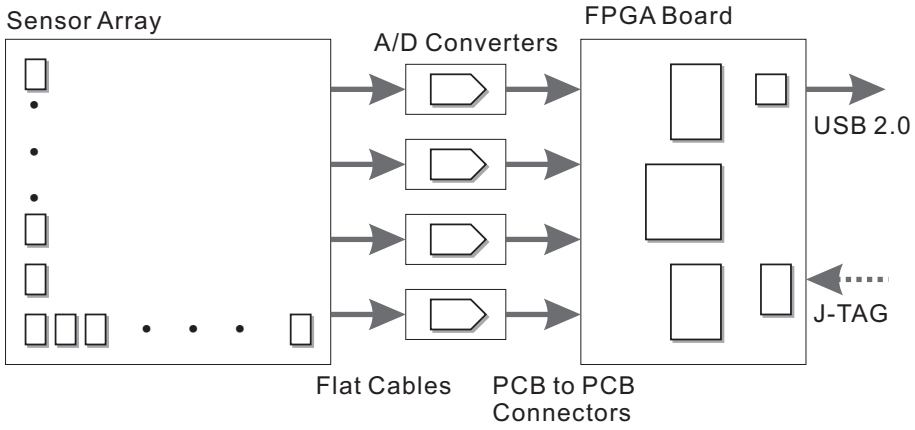


図 A.1 ボードの分割と信号の流れ

この超音波イメージング装置は、機能と回路デザイン上要求される仕様から分割した次の 3 つの部分から構成される。

1. センサーアレイ（128 個の MEMS マイクロフォンが搭載される）
2. A/D コンバーターモジュール （32 個の A/D コンバータが搭載され、計 4 つのモジュールから成る）
3. 信号処理ボード（信号処理用の FPGA とバッファメモリが搭載される）

センサーアレイはその性格上、他の装置からは離れた位置に自由な角度で設置できることが望ましく、A/D コンバーターモジュールとの間をケーブルで接続する。A/D コンバーターモジュールと信号処理ボード間は、信号遅延を生じさせないため近接して設置することが望ましく、基板間をコネクターを用いて接続する。

表 A.1　装置を構成するコンポーネントと特徴

| コンポーネント | 配線幅間 | 層数 | 精度 | アナデジ種別 | 枚数 |
|---|---|---|---|---|---|
| センサーアレイ | 0.127mm | 4 層 | 高 | アナログ | 1 |
| ADC モジュール | 0.127mm | 4 層 | 中 | アナデジ混載 | 4 |
| 信号処理ボード | 0.100mm | 4 層 | 高 | デジタル | 1 |

入力装置の電子回路の設計において、アナログ動作部分とデジタル動作部分の分離が課題となる。特に電圧の安定性と配線間から信号に混入するノイズの低減が要求されるアナログ電子回路にとって、高速なスイッチング動作と大電流が要求されるデジタル電子回路の動作はノイズ源であり、電源電圧の変動をもたらす要因でもある。このためアナログ部分とデジタル部分における電源配線の分離が重要な課題になる。機能毎の基板の分割は、この観点から言っても好ましい効果をもたらすものである。また、基板設計に要求される精度は特に信号処理ボードで高く、基板の分割はこの点でも有利になる（表 A.1)。

## A.2　MEMS センサーアレイ

センサーアレイに要求されるデザイン上の仕様は、アレイの開口とセンサー間ピッチから決定される。

センサーアレイの持つ性能のうち、周波数帯域と指向性はもっぱら使用するマイクロフォン素子の性能に還元される性能である。従来型の圧電素子 (Piezoelectric Element) に比較して、MEMS マイクロフォンは広帯域特性を大幅に改善するもので、アレイの装置の高密度化と同時に広帯域化を実現するものとして期待されている。本装置では、MEMS 超音波センサーとして SPM0404UD5 (Knowles Acoustics) を使用した。

センサー間ピッチは横方向に 5.08mm、縦方向に 10.16mm であり、これにより中心周波数 40kHz の音源に対して横方向に 113.56 度、縦方向に 49.46 度の視野角が実現される。

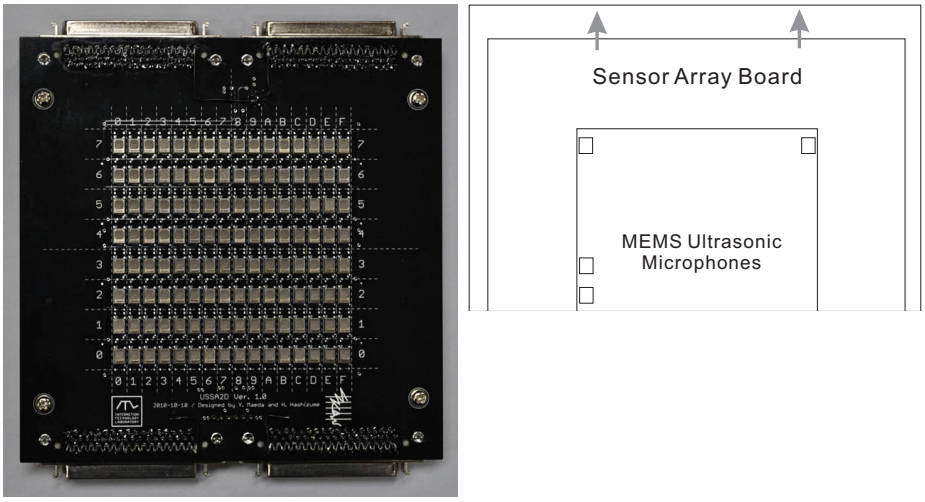


図 A.2　センサーアレイ

横方向に 16 列、縦方向に 8 列の長方形グリッド状に 128 個のセンサを配置した (図 A.2)。これにより実現されるアレイ開口は横方向に 76.20mm、縦方向に 70.42mm となり、これによる解像度は中心周波数 40kHz の音源に対し横方向に 7.2 度、縦方向に 14.5 度である。

高密度によるセンサーの実装と、数メートルにおよぶ信号の安定した伝送を両立するために、基板裏面に OP アンプを配置してバッファーアンプとし、低出力インピーダンスを確保した。個々のセンサに対して、インピーダンス調整と増幅のために反転増幅器を後段に配している。センサ基板上の実装面積確保のために両面実装を行い、通常の部品面にはセンサのみを置き、他の部品は通常の半田面に配している。内層は 1 層を GND、もう 1 層をセンサ用電源電圧 2.5V の配線に使用した。

## A.3　A/D コンバーターモジュール

ADC モジュールには、サンプリング周波数 1MHz、量子化ビット数 12bit の小型でシリアル出力を持つ A/D コンバータ AD7475 (Analog Devices) を 1 枚あたり 32 個搭載する。128 個のセンサ信号をデジタル化するために、本システムはこの ADC モジュールを計 4 枚使用する。

我々のシステムにおいて、ADC モジュールは唯一のアナログ回路とデジタル回路を同時に搭載する基板であり、PCB デザイン上注意すべき点は多い。このような基板では、デジタル回路のスイッチングノイズが、電源ライン等を通してアナログ回路にまわりこんで、SNR の低下などの問題をひき起こすことがある。この問題を避けるた

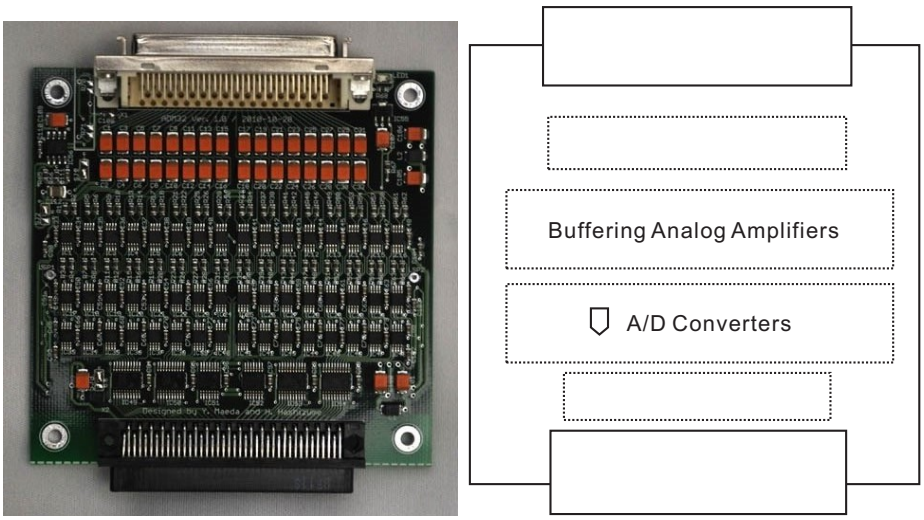


図 A.3　A/D コンバータボード

め、ADC モジュールでは、アナログ系とデジタル系で電源電圧を別々に供給し、それぞれ十分な量のキャパシタで GND とのカップリングを行った。内層の 1 層はアナログ・デジタル共通 GND とし、もう 1 層を増幅回路用 OP アンプの動作に配慮して中点電位 1.65V を供給している。

## A.4　信号処理ボード

信号処理ボードには、ロジック回路を実装するため大規模 FPGA XC3S1000FG456 (Xilinx) [Xil09] とコンフィグ用 ROM、FPGA に回路を書き込むための J-TAG インターフェースを設けている。FPGA ボードにはまた、PC にデータを送出するために USB 2.0 インターフェース用の IC FT2232HL (FTDI) を搭載し、大容量のキャプチャデータを格納するために SDRAM を 2 つ配している（図 A.4)。また、ADC モジュールを 4 つ装着するためのコネクタと、各ボードに電源を供給するためのレギュレータ回路を設けている。

信号処理ボードの PCB レイアウト上の特徴は、多数の配線を FPGA から引き出し、同時に低いインピーダンスで数種類の電源電圧を FPGA に供給するために、FPGA まわりにおいて多層基板の精細な配線技術が要求されることにある。

本信号処理ボードで使用した FPGA は、BGA パッケージに収められており、外周に 6 重にユーザ定義可能なピンが配置されている。これらのピンから配線を引き出すために、4 層基板を使用し、パッド径 0.45mm、ビア径 0.5mm、線幅と線間共に 0.1mm のデザインルールを用いて配線を行った。FPGA ボードの内層 2 層のうち 1

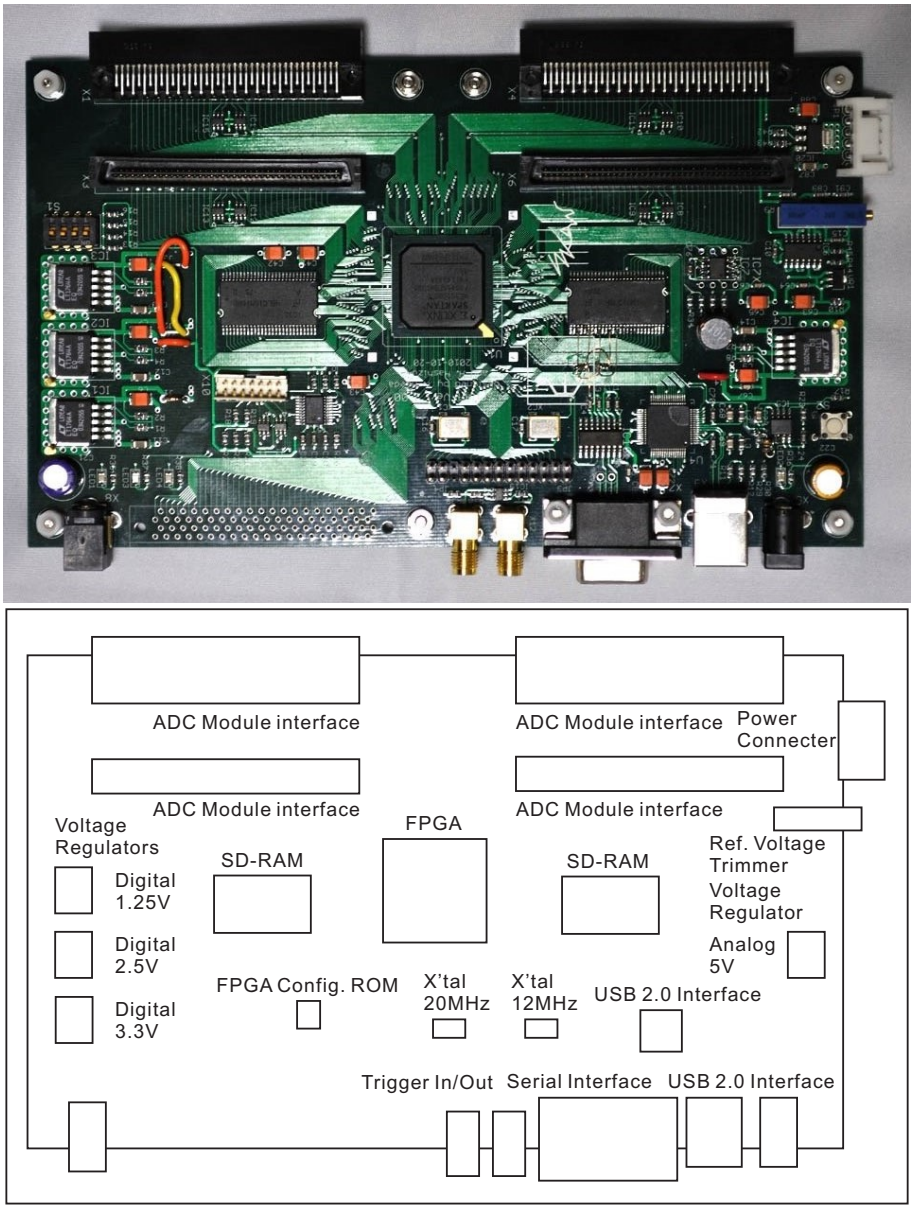


図 A.4　デジタル信号処理ボード

層は GND 層ベタパターンとした。もう 1 層は電源層とし、FPGA の使用する 3 種の電源電圧 1.2V、2.5V、3.3V を供給するため、銅箔領域を 3 分割した。

## A.5　モータードライブとトリガー回路の実装

ドップラーイメージング実験のためには再現性のある安定した速度で反射体を動かす装置が必要とされる。この目的のために、モータードライブ装置及びモーター制御

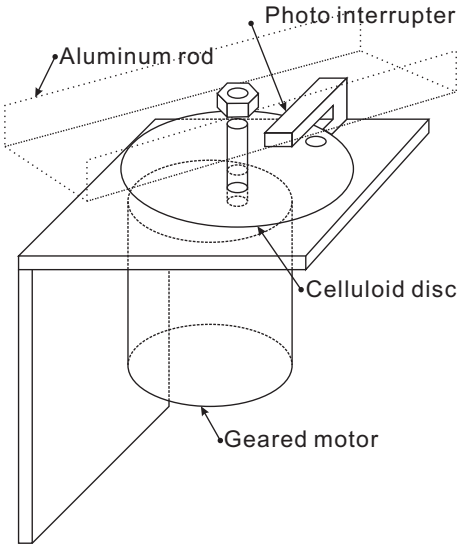


図 A.5　モータードライブ装置

回路とトリガー送出回路から成る反射体回転装置を製作し、イメージング実験のために使用した。

モータードライブ装置はロッドに取り付けられた反射体を一定の速度で回転する。回転軸上には切欠きのあるセルロイド製の円板が取り付けられて、切り欠きをフォトインタラプタで検知してトリガーを送る役割を果たす（図 A.5)。回転速度とトルクの兼ね合いから、モーターには回転速度を調整するギア付きのもの (Tamiya 540K75) を使用した。

回転速度を可変するために、パルス幅変調を行い、電界効果トランジスタによりスイッチングを行ってモーターを駆動する（図 A.6)。回路の複雑化を避けるために、回転方向は機械式スイッチで切り替えるようにした。

フォトインタラプタは円板の 1 回転ごとに切欠きを検出してトリガーを送るが、これを受信機の要求するパルスの条件にあわせるために、ワンショット・マルチバイブレーター回路による制御を行った（図 A.7)。

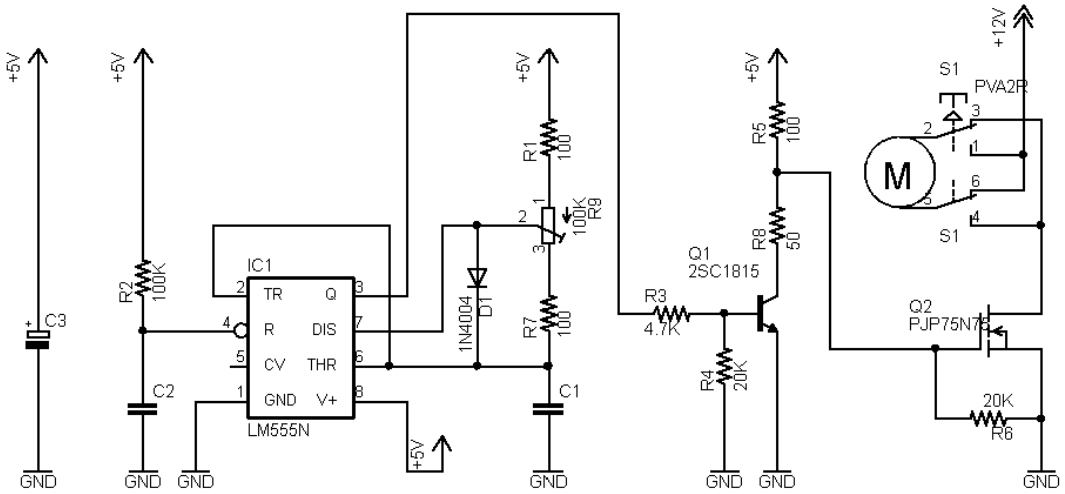


図 A.6　モータードライブ回路

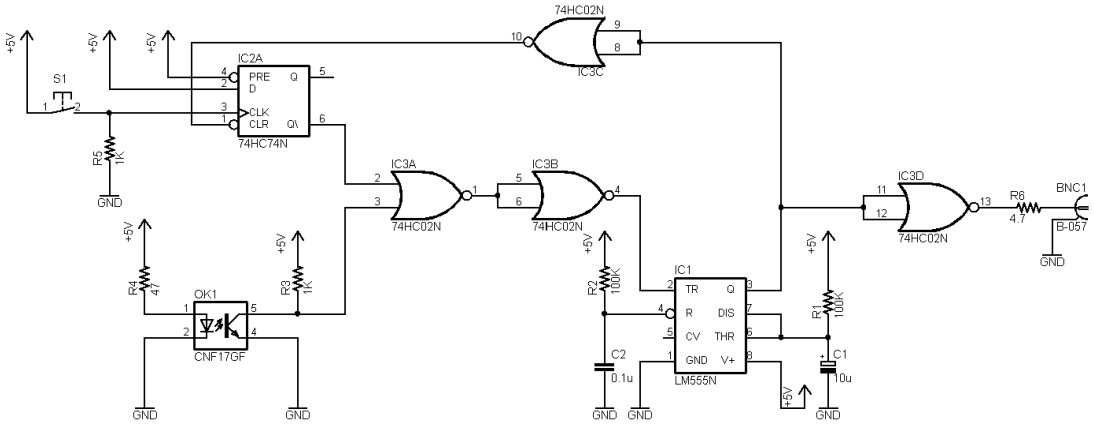


図 A.7　トリガー回路

# 付録 B

# HDL による FPGA 回路の実装

本章では、Verilog-HDL を用いて記述した FPGA 内信号処理回路について記載する。

## B.1 信号処理回路の設計と HDL による実装

前節までで紹介した PCB の製作で、信号処理ボード上の FPGA と他の回路との物理的結線が完成した。本節では、FPGA 内部の回路設計と実装について、その詳細を述べる。FPGA 内部で実現する回路は、デジタル回路であり、通常行われる回路設計の技法に基づいて必要回路の設計を行う。

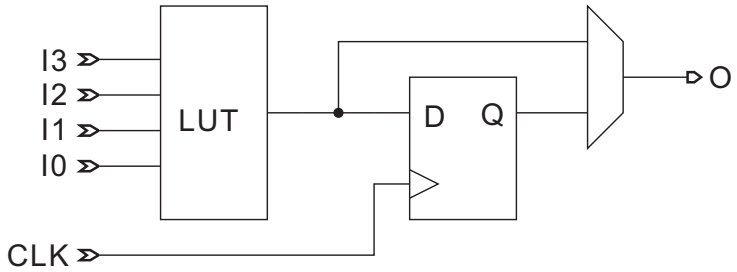


図 B.1 ルックアップテーブル

FPGA の代表的アーキテクチャは、ルックアップテーブル (LUT) と呼ばれる 4bit アドレス入力 1bit データ出力の小規模な RAM と、D-レジスタの組み合わせ（図 B.1）によって任意の論理回路と順序回路を実現するものである [Sak09]。実際のデバイスにおいては、高速化と効率化を図るための追加配線がなされている [CQ05] [Xil09]

FPGA 内部ではまた、クロック逓倍回路も実装されるが、これは FPGA デバイスに固有のデジタルクロッキングモジュール (DCM) と呼ばれる特別の組み込みモジュー

ルを用いて実現される。

本装置では、MEMS マイクロフォンアレイにより受信された超音波をチャンネル数分用意された A/D コンバータでデジタルデータとしてサンプリングし、SD-RAM バッファメモリに格納することが必要とされる。SD-RAM バッファからデータを読み出し、USB2.0 インターフェースを介して PC に送り出す動作も必要である。これらの回路の動作クロックは、一般的な組み込み用 CPU のシステムクロックと同等のものが求められるため、CPU プログラムによっては実現することができず、論理回路の設計が必要不可欠となる。この論理回路に要求される仕様は次のものとなる。

1. 128 チャンネルの A/D コンバータ出力を取りこぼしや遅延を生じさせることなく、バッファメモリに格納すること。
2. バッファメモリに格納したデータを USB2.0 インターフェースを介して PC に送出すること。
3. サンプリングレートは 500ksps、サンプリング期間は 10ms 程度とする。
4. 外部トリガ入出力と連動して動作すること。

必要とされる回路は有限状態機械 (FSM) によって制御され、多数のレジスタを持ったパイプライン型のデータ処理装置である。A/D コンバータや SD-RAM とのインターフェースの必要性とタイミング設計の容易さの観点から、回路全体を通して数 10MHz のシステムクロックを供給し同期式により動作させる。

FPGA 内の回路はハードウェア記述言語により設計を行う。回路の入出力関係を列挙して入力に対する出力の振る舞いを記述するビヘイビア記述レベルに対して、更に詳細に、フリップフロップとその間の組み合わせ論理回路を用いて、内部構造を含め回路の動作記述を行う記述レベルのことをレジスタトランスファーレベル（RTL）と呼ぶ。良く用いられるハードウェア記述言語に VHDL と Verilog-HDL 及び AHDL がある [Har03] が、これらの言語はいずれも、RTL によるレベル記述を行う。VHDL と Verilog-HDL は良く似ており、機能面においてほぼ互換性があるため、本装置の設計においてはいずれも使用することが可能であったが、過去に開発した装置との互換性やソースの融通という点での優位性から、Verilog-HDL を用いて回路記述を行った。

本装置の回路記述において、主要機能はモジュール化して分割することが可能である。各機能における FSM の動作や入出力関係はモジュール化することによって可読性が高まり開発段階における把握が容易になる。そのため本装置の回路は、次に掲げるモジュールに分割して記述した。

表 B.1 FPGA 内回路を構成するモジュールとインスタンス

| モジュール | モジュール名 | インスタンス名 |
|---|---|---|
| トップレベルシーケンサ | topbeat | topbeat() |
| A/D コンバーター駆動回路 | | |
| シリアル・パラレル変換 | mpx128to32sp | mpx128to32sp() |
| SD-RAM コントローラー | sdram16x16 | sdram0(), sdram1() |
| USB インターフェース | | |

## B.2 パイプライン型処理

パイプラインとは、複数の命令を少しずつずらして同時並行的に実行するための回路構成法であり [PH06]、複数のレジスタ群とその間を結ぶ組み合わせ回路から構成される。パイプラインを用いると、従来型の回路よりも組み合わせ回路による遅延が短い時間に収まるように設計しやすく、クロックスピードを上げることができる。

本装置のパイプラインは 4 つのレジスタ群と、その間に配置されてそれぞれ固有の機能を果たす 3 つの組み合わせ回路のモジュールから構成される（図 B.2)。

先頭に置かれた sreg は 128bit 幅を持ったシリアルデータ格納用のレジスタであり、A/D コンバータからの出力を 10MHz クロックに同期して格納する。その後段に置かれた mpx128to32sp はシリアル・パラレル変換の機能を持つ組み合わせ回路で、128bit 幅、帯域 10MHz のシリアルデータを、32bit 幅、帯域 40MHz のパラレルデータに変換し、その後に置かれたレジスタ preg に格納する。preg に格納された 32bit 幅のデータは、上位 16bit、下位 16bit に分けられてその下段のモジュール sdram16x16 に送られる。sdram0() 及び sdram1() は、同一モジュールの別々のインスタンスであり、それぞれ個々の物理デバイスに対してデータの書き込みと読み出しの機能を果たす。パイプライン動作は、モジュール sdram16x16 の段までで一度完結してキャプチャーデータの格納の機能を果たすようになっている。

キャプチャーデータの読み出しは、次に述べるトップレベルシーケンサーの状態遷移により、再びモジュール sdram16x16 から始まる。2 つの SD-RAM からの一度の読み出しにより 32bit 幅のデータが得られ、レジスタ qreg に格納される。qreg から読み出されたデータは、8bit 幅に分割されて、セレクタにより選択され、その後の ureg に格納される。最後に ureg のデータが、物理デバイスに対応したモジュール

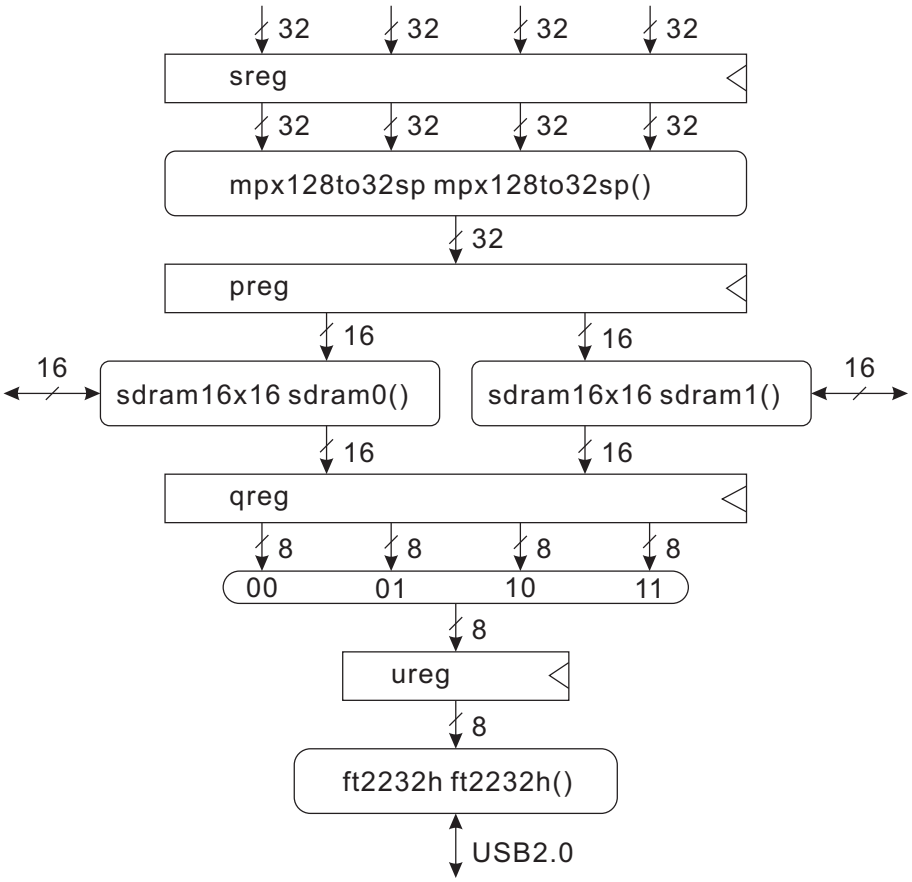


図 B.2　パイプライン構造

`ft2232h` に送られて USB 通信が行われる。

## B.3　トップレベルシーケンサ

パイプラインの制御回路を、同期型順序回路により制御する。同期型順序回路の設計法には、大きく分けてミーリ型順序回路とムーア型順序回路が知られている [Tak10] が、出力にハザードが出やすく設計の難しいミーリ型に対して、ムーア型はより安全に高速動作する論理回路を設計できるとされる。本研究では、制御回路の出力がすべてレジスタ出力になるようにムーア型に更に制約を加えて設計を行った。

通常の CPU を持つ組み込み装置が CPU に置かれた命令によって装置全体をコントロールするように、本装置は最上位に置かれた有限状態遷移機械によって全体をコントロールするように設計した。これをトップレベルシーケンサと呼ぶ。

トップレベルシーケンサは、500kHz で遷移動作する有限状態機械であり、その状態と遷移は図 B.3 によって示される。

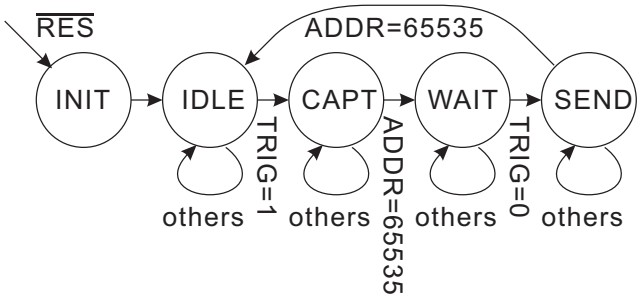


図 B.3　トップレベルシーケンサー

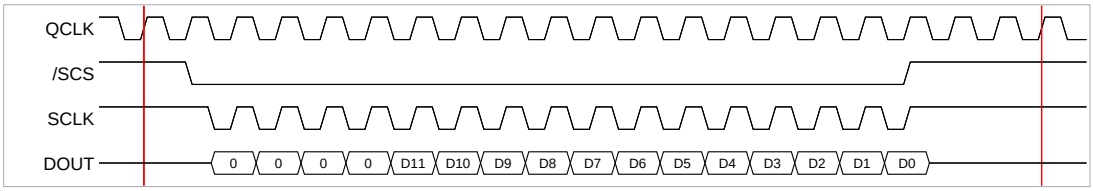


図 B.4　シリアル A/D コンバーターの駆動タイミング

トップレベルシーケンサーの状態は、/RES 信号によって INIT 状態に初期設定される。トップレベルシーケンサーに与えられる 500kHz クロックによって、すぐに次の IDLE 状態に遷移する。IDLE 状態においては装置全体は何もせずに入力の変化を待ち続ける。入力のために本装置にはトグルスイッチが接続されることが想定されており、その入力信号が TRIG である。TRIG の入力が 1 になると、トップレベルシーケンサーは CAPT 状態に遷移する。この状態においては、パイプラインの前段が駆動されてサンプリングデータの SD-RAM への格納動作を行う。SD-RAM に与えるアドレスは、アップカウンター ADDR に保持されており、0 から 65535 までの値を順にインクリメントしながら走査する。ADDR の値が 65536 に達すると、トップレベルシーケンサーは状態遷移を行い、次の WAIT 状態になる。この WAIT 状態においては、格納されたデータを維持するために必要な、SD-RAM のリフレッシュ動作を繰り返す。次にトグルスイッチからの入力が 0 に戻と、トップレベルシーケンサーは SEND 状態に遷移する。この状態においては、パイプラインの後段を使ったデータの USB インターフェースへの送出が行われる。アドレスカウンタ ADDR が 65535 まで達すると、トップレベルシーケンサーは再び状態を IDLE 状態に戻す。

## B.4　A/D コンバーター駆動回路

4 枚の A/D コンバーターモジュールに実装された A/D コンバーター IC は、FPGA 内の回路により制御信号を発生し駆動する。A/D コンバーター IC の制御信号はサンプリングの開始とデータの送出のタイミングを決定することが目的であり、複数の A/D コンバーター IC を同時に駆動するため FPGA により発生した信号をバッファで増幅して各モジュールに分配するようハードウェアが製作されている。したがって FPGA プログラムに必要とされる機能は、適切なタイミング要件を満たすように制御信号を発生することである。

今回用いた A/D コンバータ IC は、サンプリング周波数 1Msps、分解能 12bit の性能を持つ逐次比較式 (SAR) 型であり、3 線式ペリフェラルインターフェースである SPI を単一入力デバイス向けに特殊化したインターフェースを採用している。データ線はシリアル式であり、サンプリングしたデータを MSB から順にデータ線に送り出す。このデバイスはデータ入力を行う配線を省略し、デバイスを選択するための配線を加え、合計 3 線でデバイス・マイクロプロセッサ間通信を行う。

AD7475 は 2 本の信号線 SCLK と /SCS により駆動する。/SCS はコンバートの開始を知らせるトリガ信号を兼ねており、/SCS をアサートすると、アナログデータの変換を開始すると同時に、FPGA に向けてデータの送出を開始する。SCLK の立下りエッジに対応して、A/D コンバータ内部に設けられたシフトレジスタのシフトを行い、最上位ビットから順に SDATA にデータを送出する。データは 16bit 分送られるが、最初の 4bit は 0 でパディングされダミーデータであり、実効 12bit 分のデータが続けて送出される。この仕様にあわせて設計した FPGA のタイミングチャートを図 B.4 に示す。A/D コンバーター駆動回路はこれにタイミングをあわせて、128 個の A/D コンバーターからの信号を一斉にラッチする。

タイミング設計上、注意すべきことは次の点である。すなわち、SCK の立下がりエッジから、A/D コンバータのデータ出力遅延、A/D コンバータと FPGA の間に設けられたバッファの遅延、FPGA 上のデータ入力用 FF のセットアップタイムの合算が、次の SCK の立下がりエッジを超えないことである。今回設計した回路は、パルス周期の最高 20MHz までこの制約を満足する。このタイミング上の制約から、SCLK は 20MHz 最大動作周波数である。本回路は SCK に 10MHz を与えている。

## B.5　シリアル・パラレル変換

シリアル・パラレル変換機は A/D コンバータから送られるシリアル・データを SD-RAM に格納可能なパラレル・データに変換する。

128 個の A/D コンバータからのシリアル信号を集めると、FPGA は 128bit のバス幅で時系列に 16 ワード分並んだデータを 1$\mu$s 毎に受け取ることになる。しかしこのままでは SD-RAM に格納できず、信号処理に適さないので、シリアルパラレル変換を施して 1 サンプルが同一ワード内にパックされた形式に変換する。 16bit 幅のデータが 2 組、64 ワード並んだ形に変換する。これがシリアルパラレルコンバータ回路の機能である。

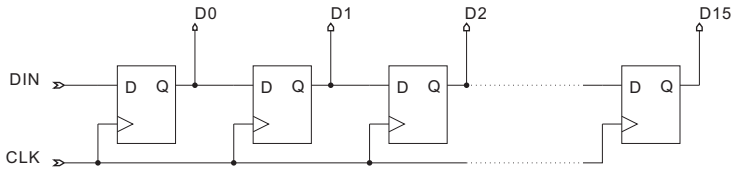


図 B.5　従来型のシフトレジスタによるシリアル・パラレル変換機

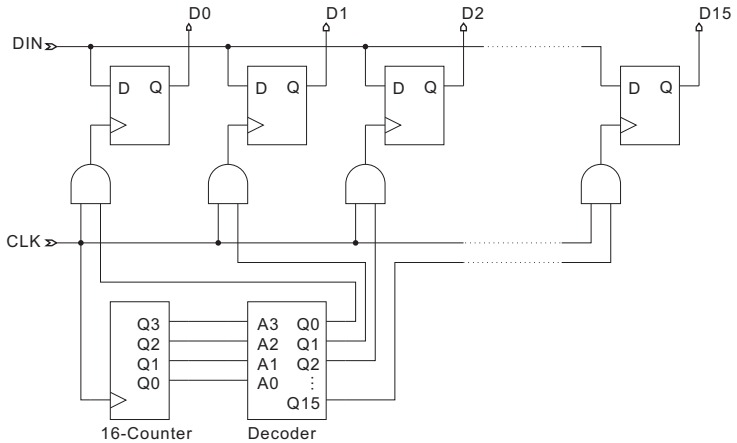


図 B.6　メモリアドレッシング型のシリアル・パラレル変換機（概念図）

従来、小規模なシリアル・パラレル変換機の実現のために D-フリップフロップを多重連結したシフトレジスタが用いられてきた（図 B.5）。この回路を FPGA 上で実現することは容易であるが、大規模化した場合に、クロック信号のファンアウト数の増加と、データシフト時の消費電力の増大を招いてしまう。今回の設計では、シフトレジスタによる方法に代えて、メモリアドレッシングによるシリアル・パラレル変換機の実装方法を用いることにした（図 B.6）。この方法はシステムクロック s に従って

増加するカウンタの出力をメモリ型レジスタのアドレスに与えることによりシリアルデータの格納先を指定する。この方法はデータ移動の頻度が低いため回路設計が容易で消費電力を抑えることができる。

## B.6　SD-RAM コントローラー

従来から、高速アクセス可能な記憶装置として用いられる RAM には、スタティック式 (S-RAM) とダイナミック式 (D-RAM) のものがあった。組み込み用として良く用いられてきたのは制御が簡単なスタティック式のものである。ところがスタティック式のものは容量の限界から、多チャンネルのサンプルデータを長時間記録する目的には不向きであり、ダイナミック式の方が高速性と記憶容量を同時に達成する目的に適している。今回バッファメモリとして採用した同期ダイナミック式 (SD-RAM) は、総合的なバランスの良い規格であり、標準的なロジック回路で高速大容量の記憶装置を実現することができる。

SD-RAM が D-RAM から派生する時に、デバイス内部の回路設計が非同期式から同期式に変更された。ファースト・ページ・モードは、D-RAM の高速アクセスのための手段の一つとして考案された。この機能は SD-RAM に受け継がれ、最初に与えられた行アドレスを SD-RAM 内部に保持したまま、クロック毎に与えられる列アドレスに対応するデータを読み書きする機能として維持されている。

信号処理ボードの仕様においては、信号受信が約 2Gbps で行われるのに対し、データ出力のための USB 2.0 の転送スピードは理論値でも 480Mbps と低速である。そのため取得データのバッファリングを行う必要がある。

信号処理ボード上には、データの一時格納用に SD-RAM が設けられている。SD-RAM は IC 単体で独立した有限状態機械と見なすことが出来、そのドライブのためには、SD-RAM と同期して動く有限状態機械が必要である。これに加えて、有限状態機械の状態遷移に合わせて SD-RAM とのデータの授受を行う仕組みも必要とされる。PC 用に開発された CPU には、同時に SD-RAM 等の記憶装置をドライブするチップセットも同時に提供されるのが普通であるが、組み込み用途のためには、独自の回路設計が必要とされる。本研究では、独自の回路を FPGA 上に実現した。ハードウェア上回路の固定したチップセットを用いる場合に比べて、FPGA 上の回路は動作スピード等のパフォーマンス上では劣るが、SD-RAM ドライブ上の各種パラメーターや、ドライブ順序などを自由に決定できる利点は大きい。

Xilinx 社から提供される FPGA に SD-RAM を接続するためのライブラリとしては、 XAPP134 [Xil00] が知られている。 しかしこのライブラリは、扱えるアドレ

ス空間が小さく、バースト長の問題により使いづらい。本稿では目的に沿った形でSD-RAM アクセスを行うために回路を独自開発した。この回路の特徴は「ロングライトモード」を持つことである。このモードは、64 ワード分のデータを遅延なく書き込むもので、SD-RAM にもともと準備されたバースト転送モードを超える性能を持っている。ロングライトモードの特長は、行アドレスを最初に一度だけ与るためアクセスが速く、バーストモードと比べてより長いワード数データの転送を行うことができることである。本稿ではこのロングライトモードを、SD-RAM に受け継がれたファーストページモードを利用することによって実現する。

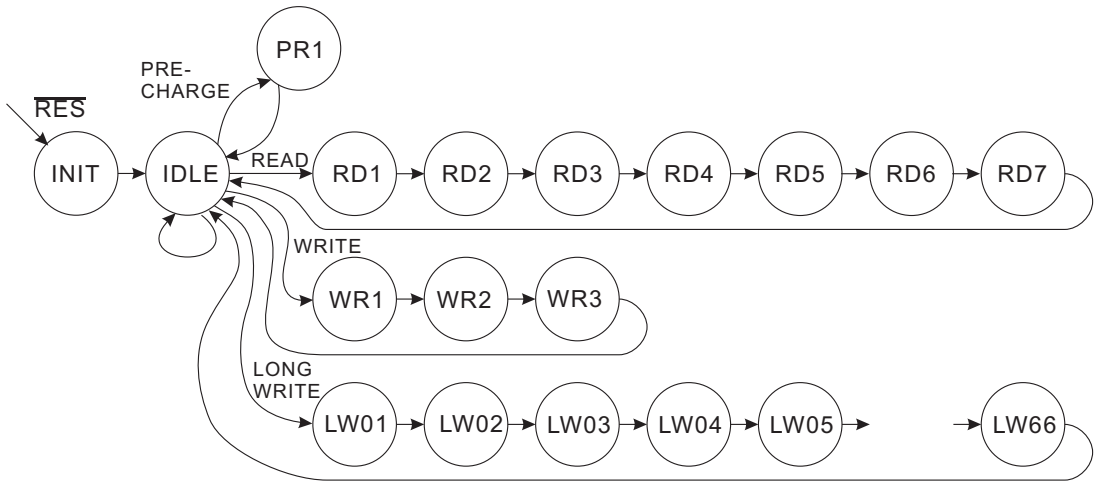


図 B.7 SD-RAM コントローラの状態遷移図

SD-RAM コントローラーは、大きな有限状態機械により構成される（図 B.7)。この有限状態機械は外部からの RESET 入力によって、INIT コマンドを発行後ただちに IDLE 状態に遷移する。IDLE 状態においては、$\hat{\text{I}}$NIT 以外の任意のコマンドを受け付けることが出来、コマンドを受け付けると直ちに当該コマンドの実行状態に遷移する。遷移と同時に FPGA に接続された SD-RAM の制御線を操作して必要な処理を行う。PRECHARGE コマンドを除き、コマンドの実行には複数の状態遷移が必要なため、サブステートを表すカウンタが設けられており、必要ステップ実行後に再び IDLE 状態に遷移する。表 B.2 に SD-RAM コントローラーのコマンドを示す。

次に SD-RAM コントローラーの各コマンドの動作を説明する

- INIT 外部リセット入力の直後に一度だけ実行される。SD-RAM に対して各種パラメータの設定を行う。
- PRECHARGE SD-RAM に対するアクセスがないときに定期的に実行される。SD-RAM に格納されたデータ保持のために SD-RAM のプレチャージコマンド

表 B.2　SD-RAM コントローラーのコマンドと機能

| コマンド名 | ステップ数 | 機能 |
|---|---|---|
| INIT | 1 | 各種パラメータの設定を行う |
| PRECHARGE | 1 | SD-RAM のプレチャージを行う |
| READ | 7 | 1 ワードのデータ読み出しを行う |
| WRITE | 3 | 1 ワードのデータ書き込みを行う |
| LONG_WRITE | 67 | 連続する 64 ワードのデータ書き込みを行う |

を発行する。

- READ SD-RAM から 1 ワードのデータ読み出しを行う。このコマンドの実行には CAS レイテンシが関係しており、書き込みに比べ余分なサブステートを必要とする。
- WRITE SD-RAM に対し 1 ワードのデータ書き込みを行う。データはアドレスと同時に与えることが出来るため、少ないサブステート数で終了する。
- LONG_WRITE SD-RAM に対し連続する 64 ワードのデータ書き込みを行う。ここで用いるファーストページアクセスは 1 回の書き込みに 1 クロックサイクルしか要しないので、理想的には、64 回の書き込みを 64 のサブステートで行えることになるが、コマンド発行及び行アドレスの指示、列アドレスの指示との間にはさむ余分なステートの関係でより多いサブステートが必要になる。

この同期式への設計変更によって、SD-RAM を FPGA から用いる場合に調整すべきポイントは、システムクロックからデータ入出力までのタイミングである。今回の設計では、SD-RAM の駆動クロックが 40MHz と SD-RAM の限界値に対して 2 倍以上の余裕があったため、有限状態機械の遷移が駆動クロックの立ち上がりエッジで行われるのに対し、SD-RAM からのデータ読み出しをその立下りエッジで行うことによって便宜的に解決を行った。

## B.7　USB インターフェース

USB2.0 は最高転送速度 480Mbps での転送を実現するペリフェラルインターフェースで、ホスト側の普及が広範に進んだことにより、組み込み機器での利用も増えている。今回使用したインターフェース用デバイス FT2232HL (FTDI) は、これらの条件を満たすものであり、ホスト側デバイスドライバの便宜性と安定性によって広くも

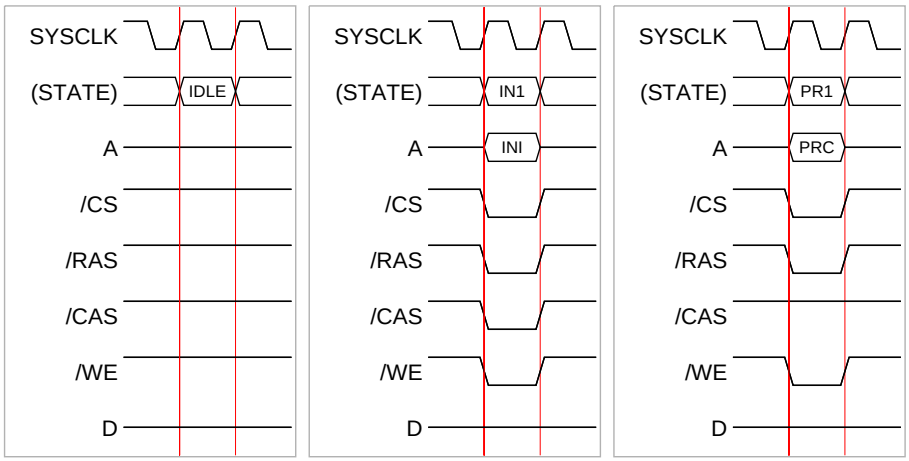


図 B.8　IDLE 、INIT 及び PRECHARGE シーケンス

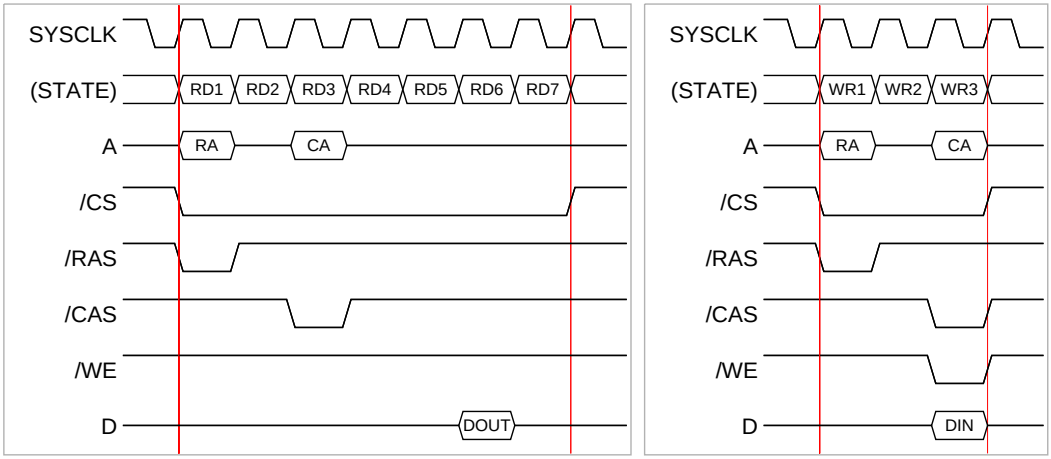


図 B.9　READ シーケンスと WRITE シーケンス

ちいられてきている。

高速転送を実現するインターフェースデバイスとマイクロプロセッサとの接続には通常バス接続型のものが用いられる。今回のデバイスも 8bit のデータ線と、4 本の制御線によるバス接続型の設定が可能であり、マイクロプロセッサからの制御が簡単に行える。デバイス内にはデータ転送を緩衝するリングバッファが設けられており、マイクロプロセッサ・デバイス間およびデバイス・ホスト間通信における転送スピードの差を吸収することができるようになっている。

今回のシステムにおいて考慮すべきことは、32bit 幅の SD-RAM インターフェースから得られたデータを 8bit 幅に変換する必要があることである。これはレジスタ出力をセレクタで切り替えることで実現できる。

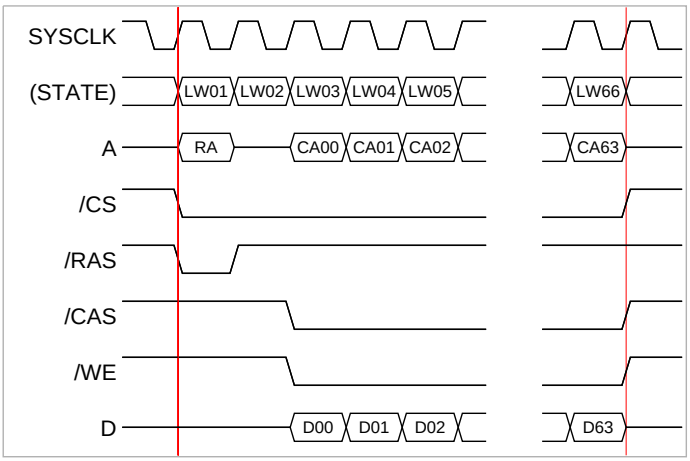


図 B.10　SD-RAM コントローラの `LONG WRITE` シーケンス

## B.8　論理合成と実装

以上の回路を試験的に Verilog HDL を用いて記述し、Xilinx 社の提供する論理合成ツールである ISE Webpack 8.1 を用いて論理合成してデバイスへの適応性を試験した。表 B.3 にその結果を示す。

表 B.3　データキャプチャ回路の論理合成結果

| **Function** | **In FPGA** | **In Use** | **Percentage** |
|---|---:|---:|---:|
| Slices | 7680 | 4061 | 52% |
| Slice Flip Flops | 15360 | 4758 | 30% |
| 4 input LUT | 15360 | 3548 | 23% |
| bonded IOBs | 333 | 275 | 82% |
| MULT18X18s | 24 | 0 | 0% |
| GCLKs | 8 | 6 | 75% |

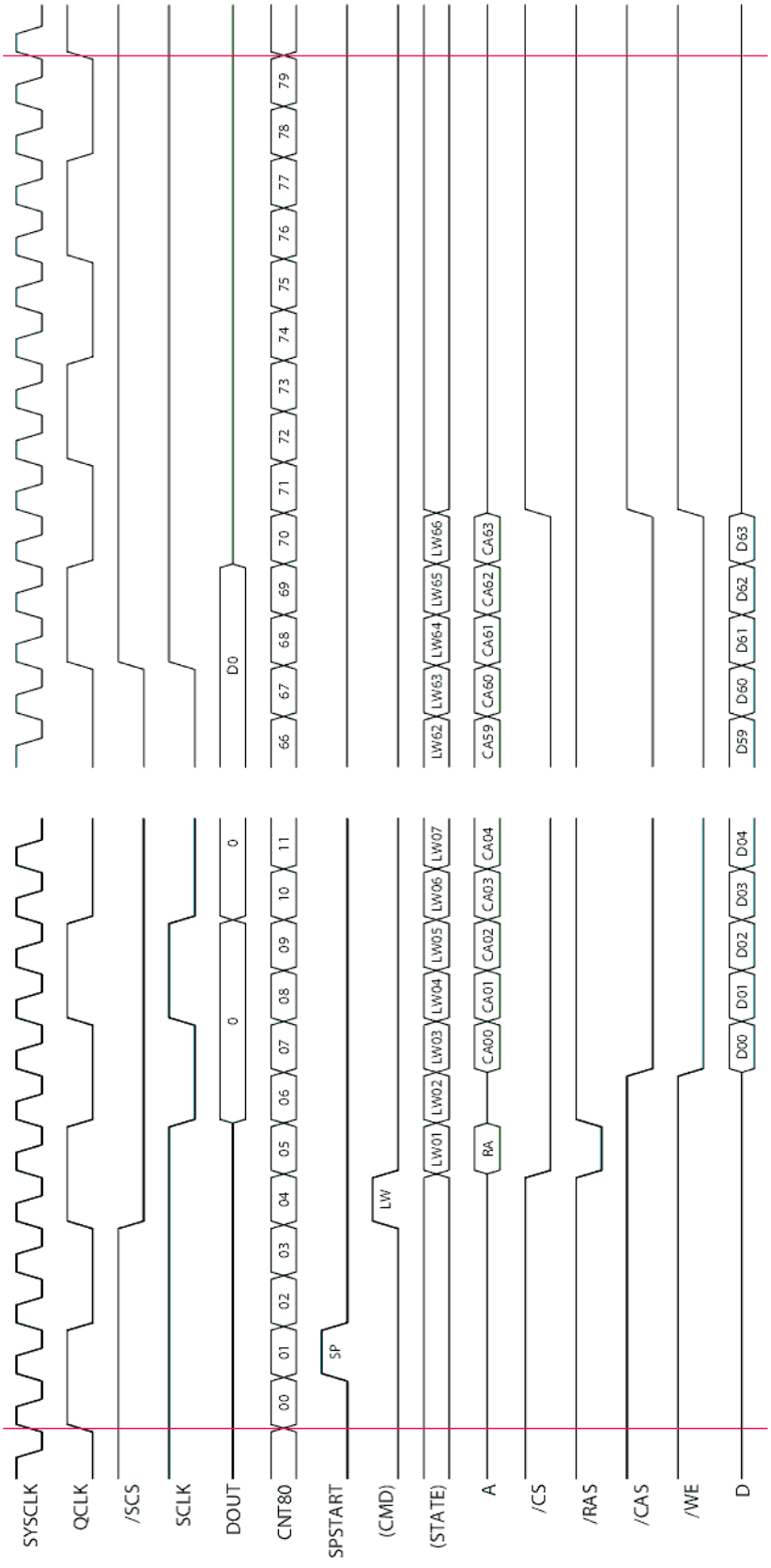


図 B.11　キャプチャ時のステート遷移全体図

# 付録 C

# FPGA の HDL 記述

本章では、FPGA の HDL 記述プログラムについて記載する。

## C.1 topbeat.v

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Create Date:    16:50:41 12/14/2010
// Module Name:    topbeat
//////////////////////////////////////////////////////////////////////////////////

‘define ADDR_MAXF 24’h3f_ff_ff
‘define ADDR_MAXC 24’h3f_ff_c0

module topbeat(
   FPGACLK, AUXCLK, nRESET,
   USBD,
   USBnRXF, USBnTXE, USBnRD, USBnWR,
   TRIGIN, TRIGOUT,
   RXD, TXD,
   LED3, LED4,
   P, X, Y,
   SDCLK, SDCLKFB,
   SD0A, SD0BA, SD0DQ,
   SD0nCS, SD0nRAS, SD0nCAS, SD0nWE, SD0DQML, SD0DQMH, SD0CKE, SD0CLK,
   SD0CLKFB,
   SD1A, SD1BA, SD1DQ,
```

```
    SD1nCS, SD1nRAS, SD1nCAS, SD1nWE, SD1DQML, SD1DQMH, SD1CKE, SD1CLK,
    SD1CLKFB,
    SCLK, SnCS,
    R0, R1, R2, R3, R4, R5, R6, R7
);
    input FPGACLK;
    input AUXCLK;
    input nRESET;
    inout [7:0] USBD;
    input USBnRXF, USBnTXE;
    output USBnRD, USBnWR;
    input TRIGIN;
    output TRIGOUT;
    input RXD;
    output TXD;
    output LED3, LED4;
    inout [18:1] P;
    inout [31:0] X;
    inout [7:0] Y;

    output SDCLK;
    input SDCLKFB;

    output [12:0] SD0A;
    output [1:0] SD0BA;
    inout [15:0] SD0DQ;
    output SD0nCS, SD0nRAS, SD0nCAS, SD0nWE, SD0DQML, SD0DQMH, SD0CKE;
    output SD0CLK;
    input SD0CLKFB;

    output [12:0] SD1A;
    output [1:0] SD1BA;
    inout [15:0] SD1DQ;
    output SD1nCS, SD1nRAS, SD1nCAS, SD1nWE, SD1DQML, SD1DQMH, SD1CKE;
    output SD1CLK;
    input SD1CLKFB;

    output SCLK, SnCS;

    input [15:0] R0;
    input [15:0] R1;
```

```
   input [15:0] R2;
   input [15:0] R3;
   input [15:0] R4;
   input [15:0] R5;
   input [15:0] R6;
   input [15:0] R7;

// Common variables

//   wire [7:0] ascii;

   wire [127:0] sdata;
   wire [31:0] pdata;
   wire [31:0] sddo;
   reg [31:0] sdlatch = 32'h0000_0000;
   reg [1:0] bytesel = 2'b00;
//   reg sdrdclk = 1'b0;
   reg usbwr = 1'b0;
   wire [7:0] usbbyte;

   wire [15:0] dataout;

// DCM dcm4x generates 80MHz clock out of 20MHz input

   wire clk1x;
   wire clk4x;
   wire clksd4x;

// And give 80MHz on FPGACLK
   dcm4x dcm4x (.CLKIN_IN(FPGACLK), .RST_IN(0), .CLKFX_OUT(clk4x),
      .CLKIN_IBUFG_OUT(), .CLK0_OUT(clk1x), .LOCKED_OUT());

// The Heart

   reg [6:0] cnt80;
   reg [23:0] addr = 24'h00_0000;

   always @(posedge clk4x) begin
      if (~nRESET) begin
         cnt80 = 0;
      end else if (cnt80 == 79) begin
```

```
            cnt80 = 0;
         end else begin
            cnt80 = cnt80 + 1;
         end
      end

   // Top-Level Sequencer

      reg [2:0] topseq = 3'b000;
      reg trigout_ = 1;

      always @(posedge clk4x) begin
         if (~nRESET) begin
            topseq = 3'b000;
            addr = 24'h00_00_00;
         end else begin
            if (topseq == 3'b000) begin            // INIT
               if (cnt80 == 79) begin
                  topseq = 3'b100;
                  addr = 24'h00_00_00;
               end else begin
                  topseq = 3'b000;
                  addr = 24'h00_00_00;
               end
            end else if (topseq == 3'b100) begin   // IDLE
               if ((cnt80 == 79) & (TRIGIN == 0)) begin
                  topseq = 3'b001;
                  addr = 24'h00_00_00;
               end else begin
                  topseq = 3'b100;
                  addr = 24'h00_00_00;
               end
            end else if (topseq == 3'b001) begin   // CAPT
               if ((cnt80 == 79) & (addr == `ADDR_MAXC)) begin
                  topseq = 3'b010;   // -> WAIT
                  addr = 24'h00_00_00;
               end else if (cnt80 == 79) begin
                  topseq = 3'b001;
                  addr = addr + 24'h00_00_40;
               end else begin
                  topseq = 3'b001;
```

```
            end
         end else if (topseq == 3'b010) begin   // WAIT
            if ((cnt80 == 79) & (TRIGIN == 1)) begin
               topseq = 3'b011;
               addr = 24'h00_00_00;
            end else begin
               topseq = 3'b010;
               addr = 24'h00_00_00;
            end
         end else if (topseq == 3'b011) begin   // SEND
            if ((cnt80 == 79) & (addr[7:0] == 8'hff)) begin
               addr = addr - 8'hff;               // (TRICK)
               topseq = 3'b111;
            end else if (cnt80 == 79) begin
               topseq = 3'b011;
               addr = addr + 1;
            end else begin
               topseq = 3'b011;
            end
         end else if (topseq == 3'b111) begin   // SEND (PAUSE)
            if ((cnt80 == 79) & (addr == `ADDR_MAXF)) begin
               topseq = 3'b100;
            end else if ((cnt80 == 79) & (addr[7:0] == 8'hff)) begin
               topseq = 3'b011;
               addr = addr + 1;
            end else if (cnt80 == 79) begin
               topseq = 3'b111;
               addr = addr + 1;
            end else begin
               topseq = 3'b111;
            end
         end
      end
   end

   always @(posedge clk4x) begin
      if (~nRESET) begin
         trigout_ = 1;
      end else if ((topseq == 3'b001) && (addr <= 24'h00_00_40)) begin
         trigout_ = 0;
      end else begin
```

```
            trigout_ = 1;
        end
    end

// Generates two signals sncs_ and sclk_ for Serial A/D converters

    reg sncs_;
    reg sclk_;
    reg spstart;

    always @(posedge clk4x) begin
        if (~nRESET) begin
            sncs_ = 1;
        end else if ((topseq == 3'b001) && (cnt80 == 3)) begin
            sncs_ = 0;
        end else if ((topseq == 3'b001) && (cnt80 == 67)) begin
            sncs_ = 1;
        end
    end

    always @(posedge clk4x) begin
        if (~nRESET) begin
            sclk_ = 1;
        end else if ((topseq == 3'b001) && (cnt80 >= 5) && (cnt80 <= 67)
         && (cnt80[1:0] == 2'b01)) begin
            sclk_ = 0;
        end else if ((topseq == 3'b001) && (cnt80 >= 5) && (cnt80 <= 67)
         && (cnt80[1:0] == 2'b11)) begin
            sclk_ = 1;
        end
    end

// Sequencer for Serial to Parallel Converter

    always @(posedge clk4x) begin
        if (~nRESET) begin
            spstart = 0;
        end else if ((topseq == 3'b001) && (cnt80 == 1)) begin
            spstart = 1;
        end else begin
            spstart = 0;
```

```
      end
   end

// Re-arrangement of serial input pins

   adcpins adcpins(.SO(sdata), .R0(R0), .R1(R1), .R2(R2), .R3(R3),
      .R4(R4), .R5(R5), .R6(R6), .R7(R7));

// Serial to Parallel Conversion

   wire mpxcs;
   assign mpxcs = (topseq == 3'b001);

   mpx128to32sp mpx128to32sp(.DX(sdata), .DY(pdata), .CLK4X(clk4x),
    .nRES(nRESET), .CS(mpxcs),
    .SPSTART(spstart), .DBGADDR(7'h00), .DBGDATA(dataout));

// SD-RAM Control

   sdram16m16 sdram0(.SYSCLK(clk4x), .nRES(nRESET),
    .A(addr),
    .DI(pdata[15:0]),
    .DO(sddo[15:0]), .CMD(cmd), .RDY(),
    .SDCKE(SD0CKE),
    .SDnCS(SD0nCS), .SDnRAS(SD0nRAS), .SDnCAS(SD0nCAS), .SDnWE(SD0nWE),
    .SDDQMH(SD0DQMH), .SDDQML(SD0DQML),
    .SDBA(SD0BA), .SDA(SD0A), .SDDQ(SD0DQ),
    .DBGOUT());

   sdram16m16 sdram1(.SYSCLK(clk4x), .nRES(nRESET),
    .A(addr),
    .DI(pdata[31:16]),
    .DO(sddo[31:16]), .CMD(cmd), .RDY(),
    .SDCKE(SD1CKE),
    .SDnCS(SD1nCS), .SDnRAS(SD1nRAS), .SDnCAS(SD1nCAS), .SDnWE(SD1nWE),
    .SDDQMH(SD1DQMH), .SDDQML(SD1DQML),
    .SDBA(SD1BA), .SDA(SD1A), .SDDQ(SD1DQ),
    .DBGOUT());

   assign SDCLK = ~clk4x;
```

```
    always @(posedge clk4x) begin
       if ((topseq == 3'b011) && (cnt80 == 12)) begin   // for SDRAM
          sdlatch[31:0] = sddo[31:0];
       end else begin
          sdlatch[31:0] = sdlatch[31:0];
       end
    end

    always @(posedge clk4x) begin
       if (~nRESET) begin
          usbwr = 0;
       end else if ((topseq == 3'b011) &&
        (cnt80 == 16 || cnt80 == 32 || cnt80 == 48 || cnt80 == 64)) begin
          usbwr = 1;
       end else begin
          usbwr = 0;
       end
    end

    always @(posedge clk4x) begin
       if (~nRESET) begin
          bytesel = 2'b00;
       end else if ((topseq == 3'b011) && (cnt80 == 14)) begin
          bytesel = 2'b00;
       end else if ((topseq == 3'b011) && (cnt80 == 30)) begin
          bytesel = 2'b01;
       end else if ((topseq == 3'b011) && (cnt80 == 46)) begin
          bytesel = 2'b10;
       end else if ((topseq == 3'b011) && (cnt80 == 62)) begin
          bytesel = 2'b11;
       end else begin
          bytesel = bytesel;
       end
    end

    assign usbbyte[7:0] = (bytesel == 2'b00) ? sdlatch[7:0] : (
     (bytesel == 2'b01) ? sdlatch[15:8] : (
     (bytesel == 2'b10) ? sdlatch[23:16] : (
     (bytesel == 2'b11) ? sdlatch[31:24] : 8'h00 )));

// USB Interface
```

```
   ft2232h_asyncfifo ft2232h(.SYSCLK(clk4x), .nRES(nRESET),
      .DI(usbbyte[7:0]), .DO(), .RD(), .WR(usbwr), .RDY(),
      .FTD(USBD), .FTnRD(USBnRD), .FTnWR(USBnWR),
      .FTnRXF(USBnRXF), .FTnTXE(USBnTXE),
      .DBGOUT());

// Output

   reg sd0test = 0;
   reg sd1test = 1;
   always @(posedge clk4x) begin
      sd0test = ~sd0test;
      sd1test = ~sd0test;
   end
   assign SD0CLK = sd0test;
   assign SD1CLK = sd1test;
   assign TRIGOUT = trigout_;
   assign SCLK = sclk_;
   assign SnCS = sncs_;
   assign LED3 = topseq[0];
   assign LED4 = topseq[1];
   assign TXD = 1;
endmodule
```

## C.2 mpx128to32sp.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Create Date:    14:12:04 08/13/2010
// Module Name:    mpx128to32sp
//////////////////////////////////////////////////////////////////////////////////

module mpx128to32sp(DX, DY, CLK4X, nRES, CS, SPSTART, DBGADDR, DBGDATA);
   input [127:0] DX;
   output [31:0] DY;
   input CLK4X;
   input nRES;
```

```
    input CS;
    input SPSTART;
    input [6:0] DBGADDR;
    output [15:0] DBGDATA;

    reg SCLK = 1;
    reg nSCS = 1;
    reg [31:0] DY;
    reg nDYEN = 1;

// INTERNAL
    reg [6:0] mpxstate = 0;
    reg [3:0] saddr = 0;
    reg [6:0] paddr = 64;
    reg [127:0] dxlatch;
    reg [127:0] sdata[15:0];
    reg [15:0] pdata[127:0];

    reg indicat;

    integer i;
    integer j;

    always @(posedge CLK4X) begin
        if (nRES == 0)
            mpxstate <= 0;
        else if (SPSTART)
            mpxstate <= 3;
        else if (mpxstate == 79)
            mpxstate <= 0;
        else if (mpxstate == 0)
            mpxstate <= 0;
        else
            mpxstate <= mpxstate + 1;
    end

// Serial Data Latch
    always @(posedge CLK4X) begin
        if ((mpxstate >= 5) & (mpxstate < 69) & (mpxstate[1:0] == 2'b01)) begin
             dxlatch[127:0] <= DX[127:0];
        end else begin
```

```
            dxlatch[127:0] <= dxlatch[127:0];
        end
    end

// Serial Data Address
// generation of saddr
    always @(posedge CLK4X) begin
       if ((mpxstate >= 5) & (mpxstate < 69) & (mpxstate[1:0] == 2'b01))
           saddr <= saddr - 1;
       else if (mpxstate == 0)
           saddr <= 0;
       else
           saddr <= saddr;
    end

     always @(posedge CLK4X) begin
      if ((mpxstate >= 5) & (mpxstate < 69) & (mpxstate[1:0] == 2'b00))
          sdata[saddr] = dxlatch;
      else
          sdata[saddr] = sdata[saddr];
     end

// Parallel Data Address
// generation of paddr
    always @(posedge CLK4X) begin
        if (mpxstate == 4)
            paddr <= 0;
        else if ((mpxstate >= 5) & (mpxstate < 69))
            paddr <= paddr + 1;
        else
            paddr <= paddr;
     end

     always @(posedge CLK4X) begin
        if (CS & (mpxstate >= 5) & (mpxstate < 69)) begin
            DY[31:16] = pdata[{paddr,1'b1}];
            DY[15:0] = pdata[{paddr,1'b0}];
        end else begin
            DY[31:16] = 16'hzzzz;
            DY[15:0]  = 16'hzzzz;
        end
```

```
    end

// Serial-Parallel Data Conversion

   always @(posedge SPSTART) begin
      for (i=0; i < 128; i=i+1) begin
         for (j=0; j < 16; j=j+1) begin
            pdata[i][j] <= sdata[j][i];
         end
      end
   end

   assign DBGDATA[15:0] = pdata[DBGADDR][15:0];

endmodule
```

## C.3 sdram16m16.v

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Create Date:    00:22:08 09/26/2010
// Module Name:    sdram16m16
//////////////////////////////////////////////////////////////////////////////////

module sdram16m16(SYSCLK, nRES, A, DI, DO, CMD, RDY,
   SDCKE, SDnCS, SDnRAS, SDnCAS, SDnWE, SDDQMH, SDDQML,
   SDBA, SDA, SDDQ,
   DBGOUT);

   input SYSCLK;
   input nRES;
   input [23:0] A;
   input [15:0] DI;
   output [15:0] DO;
   input [2:0] CMD;
   output RDY;

   output SDCKE;
```

```
   output SDnCS;
   output SDnRAS;
   output SDnCAS;
   output SDnWE;
   output SDDQMH;
   output SDDQML;
   output [1:0] SDBA;
   output [12:0] SDA;
   inout [15:0] SDDQ;
//   inout [15:0] SDDQ;

   output [5:0] DBGOUT;

   reg [2:0] state = 3'b000;
   reg [6:0] scnt = 0;
   reg rdy_;
   reg sdncs_ = 1;
   reg sdnras_ = 1;
   reg sdncas_ = 1;
   reg sdnwe_ = 1;
   reg [12:0] sda_ = 0;
   reg [1:0] sdba_ = 0;
   reg [8:0] sdca_ = 0;
   reg [15:0] sdd_ = 0;
   reg [15:0] sddlatch = 0;

   always @(posedge SYSCLK) begin
      if (nRES == 0) begin
         state = 3'b000;
      end else if (state == 3'b000) begin         // IDLE
         if (CMD == 3'b000) begin               // IDLE
            state = 3'b000;
            sdncs_ = 1;
            sdnras_ = 1;
            sdncas_ = 1;
            sdnwe_ = 1;
            state = 3'b000;
            scnt = 0;
         end else if (CMD == 3'b001) begin      // INIT (load mode register)
            sdncs_ = 0;
            sdnras_ = 0;
```

```
    sdncas_ = 0;
    sdnwe_ = 0;
    sdba_ = 2'b00;      //
    sda_ = 13'b000_0_00_011_0_000;      // 000_WB_OP_CL_BT_BL
    sdca_ = 9'h000;   //
    rdy_ = 0;
    state = 3'b001;
    scnt = 1;
end else if (CMD == 3'b010) begin      // READ
    sdncs_ = 0;
    sdnras_ = 0;
    sdncas_ = 1;
    sdnwe_ = 1;
    sda_ = A[21:9];   //
    sdba_ = A[23:22];   //
    sdca_ = A[8:0];   //
    rdy_ = 0;
    state = 3'b010;
    scnt = 1;
end else if (CMD == 3'b011) begin      // WRITE
    sdncs_ = 0;
    sdnras_ = 0;
    sdncas_ = 1;
    sdnwe_ = 1;
    sda_[12:0] = A[21:9];   //
    sdba_[1:0] = A[23:22];   //
    sdca_[8:0] = A[8:0];   //
    rdy_ = 0;
    state = 3'b011;
    scnt = 1;
end else if (CMD == 3'b101) begin      // PRECHARGE
    sdncs_ = 0;
    sdnras_ = 0;
    sdncas_ = 1;
    sdnwe_ = 0;
    sdba_[1:0] = A[23:22];
    sda_[12:11] = 2'b00;
    sda_[10] = 1'b1;
    sda_[9:0] = 10'b00_0000_0000;
    rdy_ = 0;
    state = 3'b101;
```

```
            scnt = 1;
         end else if (CMD == 3'b111) begin      // Long WRITE (fastpage)
            sdncs_ = 0;
            sdnras_ = 0;
            sdncas_ = 1;
            sdnwe_ = 1;
            sda_[12:0] = A[21:9];   //
            sdba_[1:0] = A[23:22];   //
            sdca_[8:0] = A[8:0];   //
            rdy_ = 0;
            state = 3'b111;
            scnt = 1;
         end
      end else if (state == 3'b001) begin      // INIT
         if (scnt == 1) begin                  // IN1
            sdncs_ = 1;
            sdnras_ = 1;
            sdncas_ = 1;
            sdnwe_ = 1;
            rdy_ = 1;
            state = 3'b000;
            scnt = 0;
         end
      end else if (state == 3'b010) begin      // READ
         if (scnt == 1) begin                  // RD1
            sdncs_ = 0;
            sdnras_ = 1;
            sdncas_ = 1;
            sdnwe_ = 1;
            state = 3'b010;
            scnt = scnt + 1;
         end else if (scnt == 2) begin         // RD2
            sdncs_ = 0;
            sdnras_ = 1;
            sdncas_ = 0;
            sdnwe_ = 1;
            sda_[12:11] = 2'b00;
            sda_[10] = 1'b1;          // auto precharge
            sda_[9] = 1'b0;
            sda_[8:0] = sdca_[8:0];   // column address
            state = 3'b010;
```

```
        scnt = scnt + 1;
    end else if (scnt == 3) begin          // RD3
        sdncs_ = 0;
        sdnras_ = 1;
        sdncas_ = 1;
        sdnwe_ = 1;
        state = 3'b010;
        scnt = scnt + 1;
    end else if (scnt == 4) begin          // RD4
        sdncs_ = 0;
        sdnras_ = 1;
        sdncas_ = 1;
        sdnwe_ = 1;
        state = 3'b010;
        scnt = scnt + 1;
    end else if (scnt == 5) begin          // RD5
        sdncs_ = 0;
        sdnras_ = 1;
        sdncas_ = 1;
        sdnwe_ = 1;
        state = 3'b010;
        scnt = scnt + 1;
    end else if (scnt == 6) begin          // RD6
        sdncs_ = 0;
        sdnras_ = 1;
        sdncas_ = 1;
        sdnwe_ = 1;
        state = 3'b010;
        scnt = scnt + 1;
    end else if (scnt == 7) begin          // RD7
        sdncs_ = 1;
        sdnras_ = 1;
        sdncas_ = 1;
        sdnwe_ = 1;
        rdy_ = 1;
        state = 3'b000;
        scnt = 0;
    end
end else if (state == 3'b011) begin        // WRITE
    if (scnt == 1) begin                   // WR1
        sdncs_ = 0;
```

```
            sdnras_ = 1;
            sdncas_ = 1;
            sdnwe_ = 1;
            scnt = scnt + 1;
         end else if (scnt == 2) begin          // WR2
            sdncs_ = 0;
            sdnras_ = 1;
            sdncas_ = 0;
            sdnwe_ = 0;
            sda_[12:11] = 2'b00;
            sda_[10] = 1'b1;          // auto precharge
            sda_[9] = 1'b0;
            sda_[8:0] = sdca_[8:0];   // column address
            sdd_ = DI[15:0];
            scnt = scnt + 1;
         end else if (scnt == 3) begin          // WR3
            sdncs_ = 1;
            sdnras_ = 1;
            sdncas_ = 1;
            sdnwe_ = 1;
            state = 3'b000;
            scnt = 0;
            rdy_ = 1;
         end
      end else if (state == 3'b101) begin       // PRECHARGE
         if (scnt == 1) begin                   // PR1
            sdncs_ = 1;
            sdnras_ = 1;
            sdncas_ = 1;
            sdnwe_ = 1;
            state = 3'b000;
            scnt = 0;
            rdy_ = 1;
         end
      end else if (state == 3'b111) begin       // Long WRITE (fastpage)
         if (scnt == 1) begin
            sdncs_ = 0;
            sdnras_ = 1;
            sdncas_ = 1;
            sdnwe_ = 1;
            state = 3'b111;
```

```
            scnt = scnt + 1;
            rdy_ = 0;
         end else if ((scnt >= 2) & (scnt <= 65)) begin
            sdncs_ = 0;
            sdnras_ = 1;
            sdncas_ = 0;
            sdnwe_ = 0;
            sda_[12:11] = 2'b00;
            if (scnt == 65) begin
               sda_[10] = 1'b1;      // auto precharge at the end
            end else begin
               sda_[10] = 1'b0;      // no auto precharge
            end
            sda_[9] = 1'b0;
            sda_[8:0] = sdca_[8:0];   // column address
            sdca_[8:0] = sdca_[8:0] + 1;
            sdd_[15:0] = DI[15:0];
            state = 3'b111;
            scnt = scnt + 1;
            rdy_ = 0;
         end else if (scnt == 66) begin
            sdncs_ = 1;
            sdnras_ = 1;
            sdncas_ = 1;
            sdnwe_ = 1;
            state = 3'b000;
            scnt = 0;
            rdy_ = 1;
         end
      end
   end

   always @(posedge SYSCLK) begin
      if ((state == 3'b010) & (scnt == 5)) begin
            sddlatch[15:0] = SDDQ[15:0];   // actual read
      end else begin
            sddlatch[15:0] = sddlatch[15:0];
      end
   end

   assign SDDQ[15:0] =
```

```
    ((state == 3'b011) & ~sdnwe_) | ((state == 3'b111) & ~sdnwe_)?
    sdd_[15:0] :
    16'hzzzz; // careful!
   assign DO[15:0] = sddlatch[15:0];      // careful! gate enable

   assign SDBA[1:0] = sdba_[1:0];   //
   assign SDA[12:0] = sda_[12:0];   //
   assign SDDQMH = 0;               // default operation
   assign SDDQML = 0;               // default operation
   assign SDCKE = 1;                // default operation
   assign SDnCS = sdncs_;
   assign SDnRAS = sdnras_;
   assign SDnCAS = sdncas_;
   assign SDnWE = sdnwe_;
   assign RDY = rdy_;

   assign DBGOUT = {state[2:0], scnt[2:0]};
endmodule
```

## C.4 ft2232h_asyncfifo.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Create Date:    01:50:58 01/08/2011
// Module Name:    ft2232h_asyncfifo
//////////////////////////////////////////////////////////////////////////////////

module ft2232h_asyncfifo(SYSCLK, nRES, DI, DO, RD, WR, RDY,
   FTD, FTnRD, FTnWR, FTnRXF, FTnTXE,
   DBGOUT);
    input SYSCLK;
    input nRES;
    input [7:0] DI;
    output [7:0] DO;
    input RD;
    input WR;
    output RDY;
    inout [7:0] FTD;
```

```
output FTnRD;
output FTnWR;
input FTnRXF;
input FTnTXE;
output [3:0] DBGOUT;

reg [3:0] ftstate = 4'b0000;
reg [7:0] wrdlatch = 8'h00;
reg [7:0] rddlatch = 8'h00;
reg ftnrd_ = 1;
reg ftnwr_ = 1;
reg ftden = 0;
reg rdy_ = 0;

always @(posedge SYSCLK) begin
   if (~nRES) begin
       ftnrd_ = 1;
       ftnwr_ = 1;
       ftden = 0;
       rdy_ = 0;
       ftstate = 4'b0000;
   end else if (ftstate == 4'b0000) begin
       if (RD) begin                       // RD0
          ftstate = 4'b0001;
       end else if (WR) begin              // WR1
          ftstate = 4'b1001;
          wrdlatch[7:0] = DI[7:0];
          ftden = 1;
          ftnwr_ = 1;
          rdy_ = 0;
       end else begin
          ftstate = 4'b0000;
          ftden = 0;
          rdy_ = 0;
       end
   end else if (ftstate == 4'b0001) begin // RD1
       rddlatch[7:0] = FTD[7:0];
       ftstate = ftstate + 1;
   end else if (ftstate == 4'b0010) begin // RD2
       rdy_ = 1;
       ftstate = ftstate + 1;
```

```
    end else if (ftstate == 4'b0011) begin // RD3
        rdy_ = 0;
        ftnrd_ = 1;
        ftstate = 4'b0000;
    end else if (ftstate == 4'b1001) begin // WR1
        ftden = 1;
        ftnwr_ = 1;
        rdy_ = 0;
        ftstate = ftstate + 1;
    end else if (ftstate == 4'b1010) begin // WR2
        ftden = 1;
        ftnwr_ = 0;
        rdy_ = 0;
        ftstate = ftstate + 1;
    end else if (ftstate == 4'b1011) begin // WR3
        ftden = 1;
        ftnwr_ = 0;
        rdy_ = 0;
        ftstate = ftstate + 1;
    end else if (ftstate == 4'b1100) begin // WR4
        ftden = 1;
        ftnwr_ = 0;
        rdy_ = 0;
        ftstate = ftstate + 1;
    end else if (ftstate == 4'b1101) begin // WR5
        ftden = 0;
        ftnwr_ = 1;
        rdy_ = 1;
        ftstate = 4'b0000;
    end else begin
        ftden = 0;
        rdy_ = 0;
        ftstate = 4'b0000;
    end
end

assign RDY = rdy_;
assign FTD[7:0] = ftden ? wrdlatch[7:0] : 8'hzz;
assign DO[7:0] = rddlatch[7:0];
assign FTnRD = ftnrd_;
assign FTnWR = ftnwr_;
```

```
    assign DBGOUT[3:0] = ftstate[3:0];
endmodule
```

## C.5 adcpins.v

```
‘timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Create Date:    18:42:36 03/02/2011
// Module Name:    adcpins
//////////////////////////////////////////////////////////////////////////////////

module adcpins(SO, R0, R1, R2, R3, R4, R5, R6, R7);

   output [127:0] SO;
   input [15:0] R0;
   input [15:0] R1;
   input [15:0] R2;
   input [15:0] R3;
   input [15:0] R4;
   input [15:0] R5;
   input [15:0] R6;
   input [15:0] R7;

   assign SO[7'h4F] = R0[4'h0];
   assign SO[7'h5F] = R1[4'h0];
   assign SO[7'h6F] = R0[4'h1];
   assign SO[7'h7F] = R1[4'h1];
   assign SO[7'h4E] = R0[4'h2];
   assign SO[7'h5E] = R1[4'h2];
   assign SO[7'h6E] = R0[4'h3];
   assign SO[7'h7E] = R1[4'h3];
   assign SO[7'h4D] = R0[4'h4];
   assign SO[7'h5D] = R1[4'h4];
   assign SO[7'h6D] = R0[4'h5];
   assign SO[7'h7D] = R1[4'h5];
   assign SO[7'h4C] = R0[4'h6];
   assign SO[7'h5C] = R1[4'h6];
   assign SO[7'h6C] = R0[4'h7];
```

```
assign SO[7'h7C] = R1[4'h7];
assign SO[7'h4B] = R0[4'h8];
assign SO[7'h5B] = R1[4'h8];
assign SO[7'h6B] = R0[4'h9];
assign SO[7'h7B] = R1[4'h9];
assign SO[7'h4A] = R0[4'hA];
assign SO[7'h5A] = R1[4'hA];
assign SO[7'h6A] = R0[4'hB];
assign SO[7'h7A] = R1[4'hB];
assign SO[7'h49] = R0[4'hC];
assign SO[7'h59] = R1[4'hC];
assign SO[7'h69] = R0[4'hD];
assign SO[7'h79] = R1[4'hD];
assign SO[7'h48] = R0[4'hE];
assign SO[7'h58] = R1[4'hE];
assign SO[7'h68] = R0[4'hF];
assign SO[7'h78] = R1[4'hF];

assign SO[7'h08] = R2[4'h0];
assign SO[7'h18] = R3[4'h0];
assign SO[7'h28] = R2[4'h1];
assign SO[7'h38] = R3[4'h1];
assign SO[7'h09] = R2[4'h2];
assign SO[7'h19] = R3[4'h2];
assign SO[7'h29] = R2[4'h3];
assign SO[7'h39] = R3[4'h3];
assign SO[7'h0A] = R2[4'h4];
assign SO[7'h1A] = R3[4'h4];
assign SO[7'h2A] = R2[4'h5];
assign SO[7'h3A] = R3[4'h5];
assign SO[7'h0B] = R2[4'h6];
assign SO[7'h1B] = R3[4'h6];
assign SO[7'h2B] = R2[4'h7];
assign SO[7'h3B] = R3[4'h7];
assign SO[7'h0C] = R2[4'h8];
assign SO[7'h1C] = R3[4'h8];
assign SO[7'h2C] = R2[4'h9];
assign SO[7'h3C] = R3[4'h9];
assign SO[7'h0D] = R2[4'hA];
assign SO[7'h1D] = R3[4'hA];
assign SO[7'h2D] = R2[4'hB];
```

```
assign SO[7'h3D] = R3[4'hB];
assign SO[7'h0E] = R2[4'hC];
assign SO[7'h1E] = R3[4'hC];
assign SO[7'h2E] = R2[4'hD];
assign SO[7'h3E] = R3[4'hD];
assign SO[7'h0F] = R2[4'hE];
assign SO[7'h1F] = R3[4'hE];
assign SO[7'h2F] = R2[4'hF];
assign SO[7'h3F] = R3[4'hF];

assign SO[7'h47] = R4[4'h0];
assign SO[7'h57] = R5[4'h0];
assign SO[7'h67] = R4[4'h1];
assign SO[7'h77] = R5[4'h1];
assign SO[7'h46] = R4[4'h2];
assign SO[7'h56] = R5[4'h2];
assign SO[7'h66] = R4[4'h3];
assign SO[7'h76] = R5[4'h3];
assign SO[7'h45] = R4[4'h4];
assign SO[7'h55] = R5[4'h4];
assign SO[7'h65] = R4[4'h5];
assign SO[7'h75] = R5[4'h5];
assign SO[7'h44] = R4[4'h6];
assign SO[7'h54] = R5[4'h6];
assign SO[7'h64] = R4[4'h7];
assign SO[7'h74] = R5[4'h7];
assign SO[7'h43] = R4[4'h8];
assign SO[7'h53] = R5[4'h8];
assign SO[7'h63] = R4[4'h9];
assign SO[7'h73] = R5[4'h9];
assign SO[7'h42] = R4[4'hA];
assign SO[7'h52] = R5[4'hA];
assign SO[7'h62] = R4[4'hB];
assign SO[7'h72] = R5[4'hB];
assign SO[7'h41] = R4[4'hC];
assign SO[7'h51] = R5[4'hC];
assign SO[7'h61] = R4[4'hD];
assign SO[7'h71] = R5[4'hD];
assign SO[7'h40] = R4[4'hE];
assign SO[7'h50] = R5[4'hE];
assign SO[7'h60] = R4[4'hF];
```

```
   assign SO[7'h70] = R5[4'hF];

   assign SO[7'h00] = R6[4'h0];
   assign SO[7'h10] = R7[4'h0];
   assign SO[7'h20] = R6[4'h1];
   assign SO[7'h30] = R7[4'h1];
   assign SO[7'h01] = R6[4'h2];
   assign SO[7'h11] = R7[4'h2];
   assign SO[7'h21] = R6[4'h3];
   assign SO[7'h31] = R7[4'h3];
   assign SO[7'h02] = R6[4'h4];
   assign SO[7'h12] = R7[4'h4];
   assign SO[7'h22] = R6[4'h5];
   assign SO[7'h32] = R7[4'h5];
   assign SO[7'h03] = R6[4'h6];
   assign SO[7'h13] = R7[4'h6];
   assign SO[7'h23] = R6[4'h7];
   assign SO[7'h33] = R7[4'h7];
   assign SO[7'h04] = R6[4'h8];
   assign SO[7'h14] = R7[4'h8];
   assign SO[7'h24] = R6[4'h9];
   assign SO[7'h34] = R7[4'h9];
   assign SO[7'h05] = R6[4'hA];
   assign SO[7'h15] = R7[4'hA];
   assign SO[7'h25] = R6[4'hB];
   assign SO[7'h35] = R7[4'hB];
   assign SO[7'h06] = R6[4'hC];
   assign SO[7'h16] = R7[4'hC];
   assign SO[7'h26] = R6[4'hD];
   assign SO[7'h36] = R7[4'hD];
   assign SO[7'h07] = R6[4'hE];
   assign SO[7'h17] = R7[4'hE];
   assign SO[7'h27] = R6[4'hF];
   assign SO[7'h37] = R7[4'hF];

endmodule
```

## C.6 dcm4x.v

```
//////////////////////////////////////////////////////////////////////////////
// Copyright (c) 1995-2007 Xilinx, Inc.  All rights reserved.
//////////////////////////////////////////////////////////////////////////////
//   ____  ____
//  /   /\/   /
// /___/  \  /    Vendor: Xilinx
// \   \   \/     Version : 9.1.03i
//  \   \         Application : xaw2verilog
//  /   /         Filename : dcm4x.v
// /___/   /\     Timestamp : 02/27/2014 14:06:17
// \   \  /  \
//  \___\/\___\
//
//Command: xaw2verilog -intstyle D:/WORK/john100/dcm4x.xaw -st dcm4x.v
//Design Name: dcm4x
//Device: xc3s1000-4fg456
//
// Module dcm4x
// Generated by Xilinx Architecture Wizard
// Written for synthesis tool: XST
`timescale 1ns / 1ps

module dcm4x(CLKIN_IN,
             RST_IN,
             CLKFX_OUT,
             CLKIN_IBUFG_OUT,
             CLK0_OUT,
             LOCKED_OUT);

    input CLKIN_IN;
    input RST_IN;
   output CLKFX_OUT;
   output CLKIN_IBUFG_OUT;
   output CLK0_OUT;
   output LOCKED_OUT;

   wire CLKFB_IN;
```

```
wire CLKFX_BUF;
wire CLKIN_IBUFG;
wire CLK0_BUF;
wire GND_BIT;

assign GND_BIT = 0;
assign CLKIN_IBUFG_OUT = CLKIN_IBUFG;
assign CLK0_OUT = CLKFB_IN;
BUFG CLKFX_BUFG_INST (.I(CLKFX_BUF),
                      .O(CLKFX_OUT));
IBUFG CLKIN_IBUFG_INST (.I(CLKIN_IN),
                        .O(CLKIN_IBUFG));
BUFG CLK0_BUFG_INST (.I(CLK0_BUF),
                     .O(CLKFB_IN));
// Period Jitter (unit interval) for block DCM_INST = 0.03 UI
// Period Jitter (Peak-to-Peak) for block DCM_INST = 0.82 ns
DCM DCM_INST (.CLKFB(CLKFB_IN),
              .CLKIN(CLKIN_IBUFG),
              .DSSEN(GND_BIT),
              .PSCLK(GND_BIT),
              .PSEN(GND_BIT),
              .PSINCDEC(GND_BIT),
              .RST(RST_IN),
              .CLKDV(),
              .CLKFX(CLKFX_BUF),
              .CLKFX180(),
              .CLK0(CLK0_BUF),
              .CLK2X(),
              .CLK2X180(),
              .CLK90(),
              .CLK180(),
              .CLK270(),
              .LOCKED(LOCKED_OUT),
              .PSDONE(),
              .STATUS());
defparam DCM_INST.CLK_FEEDBACK = "1X";
defparam DCM_INST.CLKDV_DIVIDE = 2.0;
defparam DCM_INST.CLKFX_DIVIDE = 1;
defparam DCM_INST.CLKFX_MULTIPLY = 2;
defparam DCM_INST.CLKIN_DIVIDE_BY_2 = "FALSE";
defparam DCM_INST.CLKIN_PERIOD = 50.0;
```

```
    defparam DCM_INST.CLKOUT_PHASE_SHIFT = "FIXED";
    defparam DCM_INST.DESKEW_ADJUST = "SYSTEM_SYNCHRONOUS";
    defparam DCM_INST.DFS_FREQUENCY_MODE = "LOW";
    defparam DCM_INST.DLL_FREQUENCY_MODE = "LOW";
    defparam DCM_INST.DUTY_CYCLE_CORRECTION = "TRUE";
    defparam DCM_INST.FACTORY_JF = 16'hC080;
    defparam DCM_INST.PHASE_SHIFT = 0;
    defparam DCM_INST.STARTUP_WAIT = "FALSE";
endmodule
```

## C.7 topbeat.ucf

```
#PACE: Start of Constraints generated by PACE

#PACE: Start of PACE I/O Pin Assignments
NET "AUXCLK"  LOC = "B11" | IOSTANDARD = LVTTL ;
NET "FPGACLK"  LOC = "A11" | IOSTANDARD = LVTTL ;
NET "LED3"  LOC = "C21" | IOSTANDARD = LVTTL ;
NET "LED4"  LOC = "C20" | IOSTANDARD = LVTTL ;
NET "nRESET"  LOC = "C11" | IOSTANDARD = LVTTL ;
NET "P[10]"  LOC = "F10"  | IOSTANDARD = LVTTL ;
NET "P[11]"  LOC = "E10"  | IOSTANDARD = LVTTL ;
NET "P[12]"  LOC = "D10"  | IOSTANDARD = LVTTL ;
NET "P[13]"  LOC = "C10"  | IOSTANDARD = LVTTL ;
NET "P[14]"  LOC = "B10"  | IOSTANDARD = LVTTL ;
NET "P[15]"  LOC = "A10"  | IOSTANDARD = LVTTL ;
NET "P[16]"  LOC = "F11"  | IOSTANDARD = LVTTL ;
NET "P[17]"  LOC = "E11"  | IOSTANDARD = LVTTL ;
NET "P[18]"  LOC = "D11"  | IOSTANDARD = LVTTL ;
NET "P[1]"  LOC = "E8"  | IOSTANDARD = LVTTL ;
NET "P[2]"  LOC = "D8"  | IOSTANDARD = LVTTL ;
NET "P[3]"  LOC = "B8"  | IOSTANDARD = LVTTL ;
NET "P[4]"  LOC = "A8"  | IOSTANDARD = LVTTL ;
NET "P[5]"  LOC = "F9"  | IOSTANDARD = LVTTL ;
NET "P[6]"  LOC = "E9"  | IOSTANDARD = LVTTL ;
NET "P[7]"  LOC = "D9"  | IOSTANDARD = LVTTL ;
NET "P[8]"  LOC = "B9"  | IOSTANDARD = LVTTL ;
NET "P[9]"  LOC = "A9"  | IOSTANDARD = LVTTL ;
```

```
NET "R0[0]"  LOC = "M22"  | IOSTANDARD = LVTTL ;
NET "R0[10]"  LOC = "R18"  | IOSTANDARD = LVTTL ;
NET "R0[11]"  LOC = "T21"  | IOSTANDARD = LVTTL ;
NET "R0[12]"  LOC = "T19"  | IOSTANDARD = LVTTL ;
NET "R0[13]"  LOC = "T17"  | IOSTANDARD = LVTTL ;
NET "R0[14]"  LOC = "U20"  | IOSTANDARD = LVTTL ;
NET "R0[15]"  LOC = "U18"  | IOSTANDARD = LVTTL ;
NET "R0[1]"  LOC = "M20"  | IOSTANDARD = LVTTL ;
NET "R0[2]"  LOC = "M18"  | IOSTANDARD = LVTTL ;
NET "R0[3]"  LOC = "N22"  | IOSTANDARD = LVTTL ;
NET "R0[4]"  LOC = "N20"  | IOSTANDARD = LVTTL ;
NET "R0[5]"  LOC = "N18"  | IOSTANDARD = LVTTL ;
NET "R0[6]"  LOC = "P22"  | IOSTANDARD = LVTTL ;
NET "R0[7]"  LOC = "P19"  | IOSTANDARD = LVTTL ;
NET "R0[8]"  LOC = "P17"  | IOSTANDARD = LVTTL ;
NET "R0[9]"  LOC = "R21"  | IOSTANDARD = LVTTL ;
NET "R1[0]"  LOC = "M21"  | IOSTANDARD = LVTTL ;
NET "R1[10]"  LOC = "T22"  | IOSTANDARD = LVTTL ;
NET "R1[11]"  LOC = "T20"  | IOSTANDARD = LVTTL ;
NET "R1[12]"  LOC = "T18"  | IOSTANDARD = LVTTL ;
NET "R1[13]"  LOC = "U21"  | IOSTANDARD = LVTTL ;
NET "R1[14]"  LOC = "U19"  | IOSTANDARD = LVTTL ;
NET "R1[15]"  LOC = "V22"  | IOSTANDARD = LVTTL ;
NET "R1[1]"  LOC = "M19"  | IOSTANDARD = LVTTL ;
NET "R1[2]"  LOC = "M17"  | IOSTANDARD = LVTTL ;
NET "R1[3]"  LOC = "N21"  | IOSTANDARD = LVTTL ;
NET "R1[4]"  LOC = "N19"  | IOSTANDARD = LVTTL ;
NET "R1[5]"  LOC = "N17"  | IOSTANDARD = LVTTL ;
NET "R1[6]"  LOC = "P21"  | IOSTANDARD = LVTTL ;
NET "R1[7]"  LOC = "P18"  | IOSTANDARD = LVTTL ;
NET "R1[8]"  LOC = "R22"  | IOSTANDARD = LVTTL ;
NET "R1[9]"  LOC = "R19"  | IOSTANDARD = LVTTL ;
NET "R2[0]"  LOC = "AB20"  | IOSTANDARD = LVTTL ;
NET "R2[10]"  LOC = "W15"  | IOSTANDARD = LVTTL ;
NET "R2[11]"  LOC = "AB15"  | IOSTANDARD = LVTTL ;
NET "R2[12]"  LOC = "V14"  | IOSTANDARD = LVTTL ;
NET "R2[13]"  LOC = "U13"  | IOSTANDARD = LVTTL ;
NET "R2[14]"  LOC = "W13"  | IOSTANDARD = LVTTL ;
NET "R2[15]"  LOC = "AA13"  | IOSTANDARD = LVTTL ;
NET "R2[1]"  LOC = "AB19"  | IOSTANDARD = LVTTL ;
NET "R2[2]"  LOC = "W18"  | IOSTANDARD = LVTTL ;
```

```
NET "R2[3]"  LOC = "AA18"  | IOSTANDARD = LVTTL ;
NET "R2[4]"  LOC = "U17"  | IOSTANDARD = LVTTL ;
NET "R2[5]"  LOC = "W17"  | IOSTANDARD = LVTTL ;
NET "R2[6]"  LOC = "AA17"  | IOSTANDARD = LVTTL ;
NET "R2[7]"  LOC = "V16"  | IOSTANDARD = LVTTL ;
NET "R2[8]"  LOC = "Y16"  | IOSTANDARD = LVTTL ;
NET "R2[9]"  LOC = "AB16"  | IOSTANDARD = LVTTL ;
NET "R3[0]"  LOC = "AA19"  | IOSTANDARD = LVTTL ;
NET "R3[10]"  LOC = "AA15"  | IOSTANDARD = LVTTL ;
NET "R3[11]"  LOC = "U14"  | IOSTANDARD = LVTTL ;
NET "R3[12]"  LOC = "W14"  | IOSTANDARD = LVTTL ;
NET "R3[13]"  LOC = "V13"  | IOSTANDARD = LVTTL ;
NET "R3[14]"  LOC = "Y13"  | IOSTANDARD = LVTTL ;
NET "R3[15]"  LOC = "AB13"  | IOSTANDARD = LVTTL ;
NET "R3[1]"  LOC = "V18"  | IOSTANDARD = LVTTL ;
NET "R3[2]"  LOC = "Y18"  | IOSTANDARD = LVTTL ;
NET "R3[3]"  LOC = "AB18"  | IOSTANDARD = LVTTL ;
NET "R3[4]"  LOC = "V17"  | IOSTANDARD = LVTTL ;
NET "R3[5]"  LOC = "Y17"  | IOSTANDARD = LVTTL ;
NET "R3[6]"  LOC = "U16"  | IOSTANDARD = LVTTL ;
NET "R3[7]"  LOC = "W16"  | IOSTANDARD = LVTTL ;
NET "R3[8]"  LOC = "AA16"  | IOSTANDARD = LVTTL ;
NET "R3[9]"  LOC = "V15"  | IOSTANDARD = LVTTL ;
NET "R4[0]"  LOC = "V1"  | IOSTANDARD = LVTTL ;
NET "R4[10]"  LOC = "N6"  | IOSTANDARD = LVTTL ;
NET "R4[11]"  LOC = "N4"  | IOSTANDARD = LVTTL ;
NET "R4[12]"  LOC = "N2"  | IOSTANDARD = LVTTL ;
NET "R4[13]"  LOC = "M6"  | IOSTANDARD = LVTTL ;
NET "R4[14]"  LOC = "M4"  | IOSTANDARD = LVTTL ;
NET "R4[15]"  LOC = "M2"  | IOSTANDARD = LVTTL ;
NET "R4[1]"  LOC = "U4"  | IOSTANDARD = LVTTL ;
NET "R4[2]"  LOC = "U2"  | IOSTANDARD = LVTTL ;
NET "R4[3]"  LOC = "T5"  | IOSTANDARD = LVTTL ;
NET "R4[4]"  LOC = "T3"  | IOSTANDARD = LVTTL ;
NET "R4[5]"  LOC = "T1"  | IOSTANDARD = LVTTL ;
NET "R4[6]"  LOC = "R4"  | IOSTANDARD = LVTTL ;
NET "R4[7]"  LOC = "R1"  | IOSTANDARD = LVTTL ;
NET "R4[8]"  LOC = "P5"  | IOSTANDARD = LVTTL ;
NET "R4[9]"  LOC = "P2"  | IOSTANDARD = LVTTL ;
NET "R5[0]"  LOC = "U5"  | IOSTANDARD = LVTTL ;
NET "R5[10]"  LOC = "N5"  | IOSTANDARD = LVTTL ;
```

```
NET "R5[11]"  LOC = "N3"  | IOSTANDARD = LVTTL ;
NET "R5[12]"  LOC = "N1"  | IOSTANDARD = LVTTL ;
NET "R5[13]"  LOC = "M5"  | IOSTANDARD = LVTTL ;
NET "R5[14]"  LOC = "M3"  | IOSTANDARD = LVTTL ;
NET "R5[15]"  LOC = "M1"  | IOSTANDARD = LVTTL ;
NET "R5[1]"  LOC = "U3"  | IOSTANDARD = LVTTL ;
NET "R5[2]"  LOC = "T6"  | IOSTANDARD = LVTTL ;
NET "R5[3]"  LOC = "T4"  | IOSTANDARD = LVTTL ;
NET "R5[4]"  LOC = "T2"  | IOSTANDARD = LVTTL ;
NET "R5[5]"  LOC = "R5"  | IOSTANDARD = LVTTL ;
NET "R5[6]"  LOC = "R2"  | IOSTANDARD = LVTTL ;
NET "R5[7]"  LOC = "P6"  | IOSTANDARD = LVTTL ;
NET "R5[8]"  LOC = "P4"  | IOSTANDARD = LVTTL ;
NET "R5[9]"  LOC = "P1"  | IOSTANDARD = LVTTL ;
NET "R6[0]"  LOC = "AB11"  | IOSTANDARD = LVTTL ;
NET "R6[10]"  LOC = "U7"  | IOSTANDARD = LVTTL ;
NET "R6[11]"  LOC = "Y6"  | IOSTANDARD = LVTTL ;
NET "R6[12]"  LOC = "V6"  | IOSTANDARD = LVTTL ;
NET "R6[13]"  LOC = "AB5"  | IOSTANDARD = LVTTL ;
NET "R6[14]"  LOC = "Y5"  | IOSTANDARD = LVTTL ;
NET "R6[15]"  LOC = "AB4"  | IOSTANDARD = LVTTL ;
NET "R6[1]"  LOC = "AB10"  | IOSTANDARD = LVTTL ;
NET "R6[2]"  LOC = "Y10"  | IOSTANDARD = LVTTL ;
NET "R6[3]"  LOC = "V10"  | IOSTANDARD = LVTTL ;
NET "R6[4]"  LOC = "W9"  | IOSTANDARD = LVTTL ;
NET "R6[5]"  LOC = "U9"  | IOSTANDARD = LVTTL ;
NET "R6[6]"  LOC = "AA8"  | IOSTANDARD = LVTTL ;
NET "R6[7]"  LOC = "V8"  | IOSTANDARD = LVTTL ;
NET "R6[8]"  LOC = "AA7"  | IOSTANDARD = LVTTL ;
NET "R6[9]"  LOC = "W7"  | IOSTANDARD = LVTTL ;
NET "R7[0]"  LOC = "U11"  | IOSTANDARD = LVTTL ;
NET "R7[10]"  LOC = "AA6"  | IOSTANDARD = LVTTL ;
NET "R7[11]"  LOC = "W6"  | IOSTANDARD = LVTTL ;
NET "R7[12]"  LOC = "U6"  | IOSTANDARD = LVTTL ;
NET "R7[13]"  LOC = "AA5"  | IOSTANDARD = LVTTL ;
NET "R7[14]"  LOC = "W5"  | IOSTANDARD = LVTTL ;
NET "R7[15]"  LOC = "AA4"  | IOSTANDARD = LVTTL ;
NET "R7[1]"  LOC = "AA10"  | IOSTANDARD = LVTTL ;
NET "R7[2]"  LOC = "W10"  | IOSTANDARD = LVTTL ;
NET "R7[3]"  LOC = "U10"  | IOSTANDARD = LVTTL ;
NET "R7[4]"  LOC = "V9"  | IOSTANDARD = LVTTL ;
```

```
NET "R7[5]"  LOC = "AB8"  | IOSTANDARD = LVTTL ;
NET "R7[6]"  LOC = "W8"  | IOSTANDARD = LVTTL ;
NET "R7[7]"  LOC = "AB7"  | IOSTANDARD = LVTTL ;
NET "R7[8]"  LOC = "Y7"  | IOSTANDARD = LVTTL ;
NET "R7[9]"  LOC = "V7"  | IOSTANDARD = LVTTL ;
NET "RXD"  LOC = "A7"  | IOSTANDARD = LVTTL ;
NET "SCLK"  LOC = "AA12"  | IOSTANDARD = LVTTL ;
NET "SD0A[0]"  LOC = "E21"  | IOSTANDARD = LVTTL ;
NET "SD0A[10]"  LOC = "E20"  | IOSTANDARD = LVTTL ;
NET "SD0A[11]"  LOC = "G21"  | IOSTANDARD = LVTTL ;
NET "SD0A[12]"  LOC = "G22"  | IOSTANDARD = LVTTL ;
NET "SD0A[1]"  LOC = "E22"  | IOSTANDARD = LVTTL ;
NET "SD0A[2]"  LOC = "F18"  | IOSTANDARD = LVTTL ;
NET "SD0A[3]"  LOC = "F19"  | IOSTANDARD = LVTTL ;
NET "SD0A[4]"  LOC = "F20"  | IOSTANDARD = LVTTL ;
NET "SD0A[5]"  LOC = "F21"  | IOSTANDARD = LVTTL ;
NET "SD0A[6]"  LOC = "G17"  | IOSTANDARD = LVTTL ;
NET "SD0A[7]"  LOC = "G18"  | IOSTANDARD = LVTTL ;
NET "SD0A[8]"  LOC = "G19"  | IOSTANDARD = LVTTL ;
NET "SD0A[9]"  LOC = "G20"  | IOSTANDARD = LVTTL ;
NET "SD0BA[0]"  LOC = "E18"  | IOSTANDARD = LVTTL ;
NET "SD0BA[1]"  LOC = "E19"  | IOSTANDARD = LVTTL ;
NET "SD0CKE"  LOC = "H19"  | IOSTANDARD = LVTTL ;
NET "SD0CLK"  LOC = "H21"  | IOSTANDARD = LVTTL ;
NET "SD0CLKFB"  LOC = "H22"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[0]"  LOC = "L22"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[10]"  LOC = "K18"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[11]"  LOC = "K17"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[12]"  LOC = "J22"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[13]"  LOC = "J21"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[14]"  LOC = "J19"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[15]"  LOC = "J18"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[1]"  LOC = "L21"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[2]"  LOC = "L20"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[3]"  LOC = "L19"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[4]"  LOC = "L18"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[5]"  LOC = "L17"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[6]"  LOC = "K22"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[7]"  LOC = "K21"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[8]"  LOC = "K20"  | IOSTANDARD = LVTTL ;
NET "SD0DQ[9]"  LOC = "K19"  | IOSTANDARD = LVTTL ;
```

```
NET "SD0DQMH"  LOC = "J17"  | IOSTANDARD = LVTTL ;
NET "SD0DQML"  LOC = "C22"  | IOSTANDARD = LVTTL ;
NET "SD0nCAS"  LOC = "D20"  | IOSTANDARD = LVTTL ;
NET "SD0nCS"  LOC = "D22"  | IOSTANDARD = LVTTL ;
NET "SD0nRAS"  LOC = "D21"  | IOSTANDARD = LVTTL ;
NET "SD0nWE"  LOC = "D19"  | IOSTANDARD = LVTTL ;
NET "SD1A[0]"  LOC = "G4"  | IOSTANDARD = LVTTL ;
NET "SD1A[10]"  LOC = "G3"  | IOSTANDARD = LVTTL ;
NET "SD1A[11]"  LOC = "E4"  | IOSTANDARD = LVTTL ;
NET "SD1A[12]"  LOC = "D1"  | IOSTANDARD = LVTTL ;
NET "SD1A[1]"  LOC = "G5"  | IOSTANDARD = LVTTL ;
NET "SD1A[2]"  LOC = "G6"  | IOSTANDARD = LVTTL ;
NET "SD1A[3]"  LOC = "F2"  | IOSTANDARD = LVTTL ;
NET "SD1A[4]"  LOC = "F3"  | IOSTANDARD = LVTTL ;
NET "SD1A[5]"  LOC = "F4"  | IOSTANDARD = LVTTL ;
NET "SD1A[6]"  LOC = "F5"  | IOSTANDARD = LVTTL ;
NET "SD1A[7]"  LOC = "E1"  | IOSTANDARD = LVTTL ;
NET "SD1A[8]"  LOC = "E2"  | IOSTANDARD = LVTTL ;
NET "SD1A[9]"  LOC = "E3"  | IOSTANDARD = LVTTL ;
NET "SD1BA[0]"  LOC = "G1"  | IOSTANDARD = LVTTL ;
NET "SD1BA[1]"  LOC = "G2"  | IOSTANDARD = LVTTL ;
NET "SD1CKE"  LOC = "D2"  | IOSTANDARD = LVTTL ;
NET "SD1CLK"  LOC = "D4"  | IOSTANDARD = LVTTL ;
NET "SD1CLKFB"  LOC = "D3"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[0]"  LOC = "J5"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[10]"  LOC = "L6"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[11]"  LOC = "L5"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[12]"  LOC = "L4"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[13]"  LOC = "L3"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[14]"  LOC = "L2"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[15]"  LOC = "L1"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[1]"  LOC = "J4"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[2]"  LOC = "J2"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[3]"  LOC = "J1"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[4]"  LOC = "K6"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[5]"  LOC = "K5"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[6]"  LOC = "K4"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[7]"  LOC = "K3"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[8]"  LOC = "K2"  | IOSTANDARD = LVTTL ;
NET "SD1DQ[9]"  LOC = "K1"  | IOSTANDARD = LVTTL ;
NET "SD1DQMH"  LOC = "C1"  | IOSTANDARD = LVTTL ;
```

```
NET "SD1DQML"  LOC = "J6"  | IOSTANDARD = LVTTL ;
NET "SD1nCAS"  LOC = "H2"  | IOSTANDARD = LVTTL ;
NET "SD1nCS"  LOC = "H5"  | IOSTANDARD = LVTTL ;
NET "SD1nRAS"  LOC = "H4"  | IOSTANDARD = LVTTL ;
NET "SD1nWE"  LOC = "H1"  | IOSTANDARD = LVTTL ;
NET "SDCLK"  LOC = "AA11"  | IOSTANDARD = LVTTL ;
NET "SDCLKFB"  LOC = "Y11"  | IOSTANDARD = LVTTL ;
NET "SnCS"  LOC = "AB12"  | IOSTANDARD = LVTTL ;
NET "TRIGIN"  LOC = "C4"  | IOSTANDARD = LVTTL ;
NET "TRIGOUT"  LOC = "C3"  | IOSTANDARD = LVTTL ;
NET "TXD"  LOC = "B7"  | IOSTANDARD = LVTTL ;
NET "USBnTXE"  LOC = "E6"  | IOSTANDARD = LVTTL ;
NET "USBD[0]"  LOC = "A3"  | IOSTANDARD = LVTTL ;
NET "USBD[1]"  LOC = "B4"  | IOSTANDARD = LVTTL ;
NET "USBD[2]"  LOC = "A4"  | IOSTANDARD = LVTTL ;
NET "USBD[3]"  LOC = "E5"  | IOSTANDARD = LVTTL ;
NET "USBD[4]"  LOC = "D5"  | IOSTANDARD = LVTTL ;
NET "USBD[5]"  LOC = "C5"  | IOSTANDARD = LVTTL ;
NET "USBD[6]"  LOC = "B5"  | IOSTANDARD = LVTTL ;
NET "USBD[7]"  LOC = "A5"  | IOSTANDARD = LVTTL ;
NET "USBnRXF"  LOC = "F6"  | IOSTANDARD = LVTTL ;
NET "USBnRD"  LOC = "D6"  | IOSTANDARD = LVTTL ;
NET "USBnWR"  LOC = "C6"  | IOSTANDARD = LVTTL ;
NET "X[0]"  LOC = "B20"  | IOSTANDARD = LVTTL ;
NET "X[10]"  LOC = "D17"  | IOSTANDARD = LVTTL ;
NET "X[11]"  LOC = "C17"  | IOSTANDARD = LVTTL ;
NET "X[12]"  LOC = "B17"  | IOSTANDARD = LVTTL ;
NET "X[13]"  LOC = "F16"  | IOSTANDARD = LVTTL ;
NET "X[14]"  LOC = "E16"  | IOSTANDARD = LVTTL ;
NET "X[15]"  LOC = "D16"  | IOSTANDARD = LVTTL ;
NET "X[16]"  LOC = "C16"  | IOSTANDARD = LVTTL ;
NET "X[17]"  LOC = "B16"  | IOSTANDARD = LVTTL ;
NET "X[18]"  LOC = "A16"  | IOSTANDARD = LVTTL ;
NET "X[19]"  LOC = "E15"  | IOSTANDARD = LVTTL ;
NET "X[1]"  LOC = "C19"  | IOSTANDARD = LVTTL ;
NET "X[20]"  LOC = "D15"  | IOSTANDARD = LVTTL ;
NET "X[21]"  LOC = "B15"  | IOSTANDARD = LVTTL ;
NET "X[22]"  LOC = "A15"  | IOSTANDARD = LVTTL ;
NET "X[23]"  LOC = "F14"  | IOSTANDARD = LVTTL ;
NET "X[24]"  LOC = "E14"  | IOSTANDARD = LVTTL ;
NET "X[25]"  LOC = "D14"  | IOSTANDARD = LVTTL ;
```

```
NET "X[26]"  LOC = "B14"  | IOSTANDARD = LVTTL ;
NET "X[27]"  LOC = "A14"  | IOSTANDARD = LVTTL ;
NET "X[28]"  LOC = "F13"  | IOSTANDARD = LVTTL ;
NET "X[29]"  LOC = "E13"  | IOSTANDARD = LVTTL ;
NET "X[2]"  LOC = "B19"  | IOSTANDARD = LVTTL ;
NET "X[30]"  LOC = "D13"  | IOSTANDARD = LVTTL ;
NET "X[31]"  LOC = "C13"  | IOSTANDARD = LVTTL ;
NET "X[3]"  LOC = "A19"  | IOSTANDARD = LVTTL ;
NET "X[4]"  LOC = "D18"  | IOSTANDARD = LVTTL ;
NET "X[5]"  LOC = "C18"  | IOSTANDARD = LVTTL ;
NET "X[6]"  LOC = "B18"  | IOSTANDARD = LVTTL ;
NET "X[7]"  LOC = "A18"  | IOSTANDARD = LVTTL ;
NET "X[8]"  LOC = "F17"  | IOSTANDARD = LVTTL ;
NET "X[9]"  LOC = "E17"  | IOSTANDARD = LVTTL ;
NET "Y[0]"  LOC = "B13"  | IOSTANDARD = LVTTL ;
NET "Y[1]"  LOC = "A13"  | IOSTANDARD = LVTTL ;
NET "Y[2]"  LOC = "F12"  | IOSTANDARD = LVTTL ;
NET "Y[3]"  LOC = "E12"  | IOSTANDARD = LVTTL ;
NET "Y[4]"  LOC = "D12"  | IOSTANDARD = LVTTL ;
NET "Y[5]"  LOC = "C12"  | IOSTANDARD = LVTTL ;
NET "Y[6]"  LOC = "B12"  | IOSTANDARD = LVTTL ;
NET "Y[7]"  LOC = "A12"  | IOSTANDARD = LVTTL ;

#PACE: Start of PACE Area Constraints

#PACE: Start of PACE Prohibit Constraints

#PACE: End of Constraints generated by PACE
```

# 付録 D

# 信号処理プログラム

本章では、イメージング実験に用いた PC プログラムについて記載する。

## D.1 Beamformer/Matched filter C-program (bfmf.c)

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

#define BUFSIZE 2500
#define CAPTSIZE (65536*128*2)
#define RECVSIZE 256
#define C 340000.0      // sound speed (mm/sec)
#define SF 500000       // sampling frequency (Hz)
#define XBM 100
#define YBM 40
#define XOFS 50
#define YOFS 20
#define XSENS 16
#define YSENS 8
#define FRAMES 5000
#define STEP 10
#define COLORS 3
#define COFFSET (FRAMES * XSENS * YSENS * sizeof(float))

typedef struct XYZ {
```

```
   float x;
   float y;
   float z;
} XYZ;

XYZ s[XSENS][YSENS];
double delay[XSENS][YSENS];
float DT[XBM][YBM][XSENS][YSENS];

#define XP(x) ((double)((x) - XOFS) / 40.0)
#define YP(y) ((double)((y) - YOFS) / 40.0)

#define TXSZ 2500

int getcoord()
{
   int x0,y0;
   double X, Y;

   for (y0=0; y0 < YSENS; y0++) {
      for (x0=0; x0 < XSENS; x0++) {
         X = ((double)x0 - 7.5) * 5.08;
         Y = ((double)y0 - 3.5) * 10.16;
         s[x0][y0].x = X;
         s[x0][y0].y = Y;
         s[x0][y0].z = 0.0;
      }
   }
   return 0;
}

double dist(double xs, double ys, double xp, double yp)
{
   double d;
   d = (xp * xs + yp * ys) / sqrt(xp * xp + yp * yp + 1.0);
   return d;
}

int scanxy(double d[][YSENS], double xp, double yp)
{
   int x1, y1;
```

```
   double xs, ys;

   for (x1=0; x1 < XSENS; x1++) {
      for (y1=0; y1 < YSENS; y1++) {
         xs = s[x1][y1].x;
         ys = s[x1][y1].y;
         d[x1][y1] = (dist(xs, ys, xp, yp) / C) * SF;
      }
   }

   return 0;
}

int fillDT()
{
   int x0, y0;
   int x1, y1;
   double xp, yp;

   for (x0=0; x0 < XBM; x0++) {
      for (y0=0; y0 < YBM; y0++) {
         xp = XP(x0);
         yp = YP(y0);
         scanxy(delay, xp, yp);
         for (x1=0; x1 < XSENS; x1++) {
            for (y1=0; y1 < YSENS; y1++) {
               DT[x0][y0][x1][y1] = delay[x1][y1];
            }
         }
      }
   }
   return 0;
}

double matchf(short *sens, int t, int sx, int sy, float *tx)
{
   int u;
   int tt;
   double mm;

   mm = 0;
```

```
   for (u=0; u < TXSZ; u++) {
      tt = t + u;
      if (tt >= 0 && tt < FRAMES) {
         mm += (double)*(sens + tt * XSENS * YSENS + sy * XSENS + sx)
          * tx[u];
      }
   }

   return mm;
}

double beamf(float *sens, int t, int x, int y)
{
   int xs, ys;
   double dtime;
   int dei;
   double der;
   double bm;
   int t0, t1;

   bm = 0.0;
   for (ys = 0; ys < YSENS; ys++) {
      for (xs = 0; xs < XSENS; xs++) {
         dtime = DT[x][y][xs][ys];
         dei = floor(dtime);
         der = dtime - dei;
         t0 = t - dei;
         t1 = t0 + 1;
         if (t0 >= 0 && t1 < FRAMES) {
            bm += (double)*(sens + t0 * XSENS * YSENS + ys * XSENS + xs)
             * (1.0 - der)
             + (double)*(sens + t1 * XSENS * YSENS + ys * XSENS + xs)
             * der;
         } else {
            bm += 0.0;
         }
      }
   }
   return bm;
}
```

```
int read_tx(float *tx, char *filename)
{
   FILE *fp;
   char buf[256];
   int pos[256];
   int c;
   int i;
   int k;
   int n;
   int p;
   int v;
   double fv;

   k = 0;
   fp = fopen(filename, "r");
   while (fgets(buf, 256, fp)) {
      pos[0] = 0;
      i = 1;
      for (p=0; c = buf[p]; p++) {
         if (c == ',' && buf[p+1] >= '0' && buf[p+1] <= '9') {
             pos[i++] = p + 1;
         }
      }
      n = i;
      for (i=0; i < n; i++) {
         v = atoi(&buf[pos[i]]);
         fv = ((double)v - 32768.0) / 32768.0;
         if (k < BUFSIZE * 2 && k % 2 == 0) {
            tx[k/2] = fv;
         }
         k++;
      }
   }
   fclose(fp);
   return 0;
}

int main(int argc, char *argv[])
{
   unsigned char *p;
   int t;
```

```
int i;
int x, y;
FILE *fh;
float *beam_red;
float *beam_grn;
float *beam_blu;
float *mtch_red;
float *mtch_grn;
float *mtch_blu;
short *sens;
double bm;
double mf;
double maxabs = 0.0;
int maxframe = 0;
char logfile[256];
float *tx_red;
float *tx_grn;
float *tx_blu;
float *hsens;
float *hmtch;

tx_red = (float *)calloc((size_t)BUFSIZE, sizeof(float));
tx_grn = (float *)calloc((size_t)BUFSIZE, sizeof(float));
tx_blu = (float *)calloc((size_t)BUFSIZE, sizeof(float));

read_tx(tx_red, "tx_red.txt");
read_tx(tx_grn, "tx_grn.txt");
read_tx(tx_blu, "tx_blu.txt");

if (argc > 1) {
   strcpy(logfile, argv[1]);
   fprintf(stderr, "\tLOG: %s for Reading.\n", logfile);
} else {
   strcpy(logfile, "logfile.log");
   fprintf(stderr, "\tLOG: logfile.log.\n");
}

p = (unsigned char *)malloc((size_t)CAPTSIZE);
sens = (short *)p;
mtch_red = (float *)malloc((size_t)FRAMES * XSENS * YSENS * sizeof(float));
mtch_grn = (float *)malloc((size_t)FRAMES * XSENS * YSENS * sizeof(float));
```

```
   mtch_blu = (float *)malloc((size_t)FRAMES * XSENS * YSENS * sizeof(float));
   beam_red = (float *)malloc((size_t)FRAMES * XBM * YBM * sizeof(float));
   beam_grn = (float *)malloc((size_t)FRAMES * XBM * YBM * sizeof(float));
   beam_blu = (float *)malloc((size_t)FRAMES * XBM * YBM * sizeof(float));

   fh = fopen(logfile, "rb");
   for (i=0; i < CAPTSIZE; i += RECVSIZE) {
      fread(p+i, 1, RECVSIZE, fh);
   }
   fprintf(stderr, "\tReading Data Finished\n");
   fclose(fh);

   getcoord();
   fillDT();

   for (t=0; t < FRAMES; t++) {
      for (y=0; y < YSENS; y++) {
         for (x=0; x < XSENS; x++) {
            *(sens + t * XSENS * YSENS + y * XSENS + x) -= 2048;
         }
      }
   }
   for (t=0; t < FRAMES; t++) {
      for (y=0; y < YSENS; y++) {
         for (x=0; x < XSENS; x++) {
            mf = matchf(sens, t, x, y, tx_red);
            *(mtch_red + t * XSENS * YSENS + y * XSENS + x) = (float)mf;
            mf = matchf(sens, t, x, y, tx_grn);
            *(mtch_grn + t * XSENS * YSENS + y * XSENS + x) = (float)mf;
            mf = matchf(sens, t, x, y, tx_blu);
            *(mtch_blu + t * XSENS * YSENS + y * XSENS + x) = (float)mf;
         }
      }
   }

fprintf(stderr, "Matched filter finished.\n");

for (t=0; t < FRAMES; t++) {
for (y=0; y < YBM; y++) {
for (x=0; x < XBM; x++) {
bm = beamf(mtch_red, t, x, y);
```

```
*(beam_red + t * XBM * YBM + y * XBM + x) = (float)bm;
bm = beamf(mtch_grn, t, x, y);
*(beam_grn + t * XBM * YBM + y * XBM + x) = (float)bm;
bm = beamf(mtch_blu, t, x, y);
*(beam_blu + t * XBM * YBM + y * XBM + x) = (float)bm;
}
}
}
fprintf(stderr, "Beamforming finished.\n");

   for (t=0; t < FRAMES; t+=STEP) {
      for (y=0; y < YBM; y++) {
         for (x=0; x < XBM; x++) {
            printf("%f, %f, %f\n",
               *(beam_red + t * XBM * YBM + y * XBM + x),
               *(beam_grn + t * XBM * YBM + y * XBM + x),
               *(beam_blu + t * XBM * YBM + y * XBM + x));
         }
      }
   }

   free(beam_blu);
   free(beam_grn);
   free(beam_red);
   free(mtch_blu);
   free(mtch_grn);
   free(mtch_red);
   free(p);
}
```

# 参考文献


[Aki10] 秋山いわき. アコースティックイメージング. コロナ社, 2010.

[Alt73] R. A. Altes. Some invariance properties of the wide-band ambiguity function. *Journal of Acoustical Society of America*, Vol. 53, No. 4, pp. 1154–60, 1973.

[Asa11] 浅野太. 音のアレイ信号処理. コロナ社, 2011.

[Col91] C. R. Cole. Properties of swept fm waveforms in medical ultrasound imaging. In *Proceedings of IEEE Ultrasonics Symposium*, December 1991.

[CQ05] CQ 出版. FPGA/PLD 設計スタートアップ 2005/2006 年版. CQ 出版, 2005.

[Har03] 原田豊. VHDL, Verilog, AHDL による デジタルシステム設計. 丸善, 2003.

[HKK08] 平田慎之介, 黒澤実, 片桐崇. 線形周期変調信号を用いたパルス圧縮による超音波距離・速度計測. 日本音響学会講演論文集, pp. 1531–1532, 九州大学, September 2008.

[HKK09] 平田慎之介, 黒澤実, 片桐崇. 2 波の線形周期変調信号を用いた超音波距離・速度計測における精度・分解能の検討. 日本音響学会講演論文集, pp. 1351–1352, March 2009.

[ISH10] 伊藤俊夫, 杉本雅則, 橋爪宏達. 最適化したマルチキャリア信号と合成送信開口による高画質音響イメージング. 電子情報通信学会論文誌 A, Vol. J93-A, No. 5, pp. 341–352, 2010.

[ISH11] T. Ito, M. Sugimoto, and H. Hashizume. Evaluation of acoustic imaging system using correlation division in synthetic transmit aperture with multicarrier signals. *IEICE Trans. Fundamentals*, Vol. E94-A, No. 10, pp. 1907–1919, October 2011.

[Ita05] 伊丹誠. わかりやすい OFDM 技術. オーム社, 2005.

[ITMM02] 岩下和之, 田川憲男, 皆川明洋, 守屋正. アップチャープとダウンチャープの併用によるドップラ計測. 超音波エレクトロニクスの基礎と応用に関するシンポジウム講演予稿集, No. 23, pp. 31–32, 金沢工業大学, 金沢, November 2002.

[KJM98] R. Kažys, L. Jakevčius, and L. Mažeika. Beamforming by means of 2D phased ultrasonic arrays. *Ultragarsas*, No. 1(29), pp. 12–15, 1998.

[KNKO85] C. Kasai, K. Namekawa, A. Koyano, and R. Omoto. Real-time two-dimensional blood flow imaging using an autocorrelation technique. *IEEE Transactions on Sonics and Ultrasonics*, Vol. SU-32, No. 3, pp. 458–464, May 1985.

[KSNS04] 亀岡弘和, 斎藤翔一郎, 西本卓也, 嵯峨山茂樹. Specmurt における準最適共通調波パターンの反復推定による多声音楽信号の可視化と midi 変換. 情報処理学会研究報告［音楽情報科学］, 第 84 巻, pp. 41–48, Augest 2004.

[LNB+11] M. Legros, A. Novell, A. Bouakaz, G. Ferin, R. Dufait, and D. Certon. Tissue harmonic imaging with cmuts. In *Proceedings of IEEE International Ultrasonics Symposium 2011 (IUS2011)*, pp. 2249–2252, Orland, USA, October 2011.

[LZqBD12] Ying Luo, Qun Zhang, You quing Bal, and Yan-Li Duan. High-resolution ISAR imaging with sparse-spectrum OFDM-LFM waveforms. In *PIERS Processing*, pp. 230–234, March 2012.

[Mik05] 三上直樹. 初めて学ぶディジタル・フィルタと高速フーリエ変換. CQ 出版, 2005.

[Mit90] D. G. Mitchell. Color doppler imaging: principles, limitations, and artifacts. *Radiology*, Vol. 177, No. 1, pp. 1–10, October 1990.

[Mor76] 森本光生. 佐藤超函数入門. 共立出版, 1976.

[MSH10] 前田泰成, 杉本雅則, 橋爪宏達. マルチキャリア波を用いた超音波イメージングの基礎検討. 第 Vol.2010 巻, pp. 34–39, 東京農工大学, 東京, June 2010. BT2010-6.

[MSH11a] 前田泰成, 杉本雅則, 橋爪宏達. 平面 mems センサアレイと高速ビームフォームアルゴリズムによる実時間超音波イメージング装置の実現. pp. 45–50, 産総研，つくば, February 2011.

[MSH11b] Y. Maeda, M. Sugimoto, and H. Hashizume. Mems microphone array

and signal processor for realtime object detection. In *Proceedings of 31st International Acoustical Imaging Symposium (AI-31)*, Warsaw, Poland, April 2011.

[MSH11c] Y. Maeda, M. Sugimoto, and H. Hashizume. A robust doppler ultrasonic 3d-imaging system with mems microphone array and configurable processor. In *Proceedings of IEEE International Ultrasonics Symposium 2011 (IUS2011)*, Orland, USA, October 2011.

[MSH12a] 前田泰成, 杉本雅則, 橋爪宏達. ログステップマルチキャリア波による超音波イメージング技法の速度検出性能. 平成 24 年度第 2 回アコースティックイメージング研究会, 北海道大学, 札幌, August 2012.

[MSH12b] Y. Maeda, M. Sugimoto, and H. Hashizume. A novel algorithm for doppler imaging based on wavelet analysis. In *Proceedings. of IEEE International Ultrasonics Symposium 2012 (IUS2012)*, Dresden, Germany, October 2012.

[MSH13a] 前田泰成, 杉本雅則, 橋爪宏達. ログステップマルチキャリア波を用いた超音波ドップラー速度推定法におけるノイズ耐性の改善. 平成 25 年度第 2 回アコースティックイメージング研究会, 北海道大学, 札幌, August 2013.

[MSH13b] Y. Maeda, M. Sugimoto, and H. Hashizume. A robust doppler imaging method using log-step multicarrier ultrasonic signals. In *Proceedings of IEEE International Ultrasonics Symposium 2013 (IUS2013)*, Prague, Czech Republic, July 2013. (To appear).

[MSH14] 前田泰成, 杉本雅則, 橋爪宏達. 音響イメージングにおけるログステップマルチキャリヤ波によるドップラー速度推定法. 電子情報通信学会論文誌 A, January 2014. (To appear).

[NH06] 中村駿介, 降旗健治. 木魚音のヒルベルト変換による瞬時周波数解析. 信学技報, pp. 1–6, 2006.

[NSG98] Steven E. Noel, Harold H. Szu, and Yogesh J. Gehel. Doppler frequency estimation with wavelets and neural networks. No. 3391 Wavelet Applications V, 150, March 1998.

[PH06] David A. Patterson and John L. Hennessy. コンピュータの構成と設計, 下. 日経 BP, 第 3 版, 2006.

[RK99] P. Robertson and S. Kaiser. Analysis of the loss of orthogonality through doppler spread in ofdm systems. In *Proceedings of Global*

*Telecommunications Conference (GLOBECOM'99)*, Vol. 1b, pp. 701–706, Rio de Janeiro, Brazil, December 1999.

[RMP12] Vishal Riche, Stephane Meric, and Eric Potter. OFDM signal design for range ambiguity supression in SAR configuration. In *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, pp. 2156–2159, Munich, 2012.

[RYBI02] A. M. Rollins, S. Yazdanfar, J. K. Barton, and J. A. Izatt. Real-time in vivo color doppler optical coherence tomography. *Journal of Biomedical Optics*, Vol. 7, No. 1, pp. 123–129, January 2002.

[Sak09] 坂井修一. 実践 コンピュータアーキテクチャ. コロナ社, 2009.

[SFnk11] 佐野明, 福井大, コウモリの会. コウモリ識別ハンドブック改訂版. 文一総合出版, 2011.

[SH06] 清水一弘, 平井慎一. Cmos + fpga vision を用いた matched filter の高速化. 計測自動制御学会システムインテグレーション部門学術学術講演会, pp. 908–909, 2006.

[SKNS05] Shoichiro Saito, Hirokazu Kameoka, Takuya Nishimoto, and Shigeki Sagayama. Specmurt analysis of multi-pitch music signals with adaptive estimation of common harmonic structure. In *International Conference on Music Information Retrieval (ISMIR 2005)*, pp. 84–91, September 2005.

[SKOK10] 斎藤信弥, 黒澤実, 折野裕一郎, 片桐崇. 2 波の線形周期変調信号を用いた超音波距離・速度計測における計測精度の検討. 日本音響学会講演論文集, pp. 1433–1434, September 2010.

[SKOS06a] 齊藤翔一郎, 亀岡弘和, 小野順貴, 嵯峨山茂樹. 事後確率最大化 specmurt 分析による多重ピッチの反復推定アルゴリズム. 情報処理学会研究報告［音楽情報科学］, 第 90 巻, pp. 85–92, August 2006.

[SKOS06b] 齊藤翔一郎, 亀岡弘和, 小野順貴, 嵯峨山茂樹. 事後確率最大化 specmurt 分析による音楽音響信号の多重ピッチ推定. 日本音響学会講演論文集, pp. 581–582, September 2006.

[SKOS06c] 齊藤翔一郎, 亀岡弘和, 小野順貴, 嵯峨山茂樹. 凸射影法に基づく specmurt 分析の共通調波構造推定. 情報処理学会研究報告［音楽情報科学］, 第 45 巻, pp. 13–18, May 2006.

[SKOS06d] 齊藤翔一郎, 亀岡弘和, 小野順貴, 嵯峨山茂樹. 凸射影法に基づく specmurt 分析の共通調波構造推定アルゴリズムとその収束性に関する考察. 日本音

響学会講演論文集, pp. 553–554, March 2006.

[SNM02] 村山司, 中原史生, 森恭一. イルカ・クジラ学—イルカとクジラの謎に挑む. 東海大学出版会, 2002.

[STKN04] Shigeki Sagayama, Keigo Takahashi, Hirokazu Kameoka, and Takuya Nishimoto. Specmurt anasylis: A piano-roll-visualization of polyphonic music signals by deconvolution of log-frequency spectrum. In *Workshop on Statistical and Perceptual Audio Processing SAPA-2004*, Jeju, Korea, October 2004.

[Tak06] 高橋陽一郎. 実関数とフーリエ変換. 岩波書店, 2006.

[Tak10] 高木直史. 論理回路. オーム社, 2010.

[TNS03] 高橋佳吾, 西本卓也, 嵯峨山茂樹. 対数周波数逆畳み込みによる多重音の基本周波数解析. 情報処理学会研究報告［音楽情報科学］, 第 127 巻, pp. 61–66, December 2003.

[UG04] Elif Derya Ubeyli and Inan Guler. Spectral broadening of ophthalmic arterial doppler signals using stft and wavelet transform. *Computers in Biology and Medicine*, No. 34, pp. 345–354, 2004.

[Wil93a] Jens. E. Wilhjelm. Target velocity estimation with fm and pw echo ranging doppler systems—part i: Signal analysis. In *IEEE Transactions of Ultrasonics, Ferroelectrics, and Frequency Control*, No. 4 in 40, pp. 366–372, July 1993.

[Wil93b] Jens. E. Wilhjelm. Target velocity estimation with fm and pw echo ranging doppler systems—part ii: System analysis. In *IEEE Transactions of Ultrasonics, Ferroelectrics, and Frequency Control*, No. 4 in 40, pp. 373–380, July 1993.

[Xil00] Xilinx. *Synthesizable High Performance SDRAM Controller (XAPP134)*, 2000.

[Xil09] Xilinx. *Spartan-3 FPGA Family Data Sheet (DS099)*, 2009.

[You97] R. K. ヤング. ウェーブレット 信号処理とシステム推定への応用. トッパン, 1997.

[YRI03] S. Yazdanfar, A. M. Rollins, and J. A. Izatt. In vivo imaging of human retinal flow dynamics by color doppler optical coherence tomography. *Archives of Ophthalmology*, Vol. 121, pp. 235–239, February 2003.

[YS06] Jie Yang and Tapan K. Sarkar. A novel doppler-tolerant polyphase codes for pulse compression based on hyperbolic frequency modula-

tion. *Digital Signal Process*, 2006. doi:10.1016/j.dsp.2006.09.006.