FPGAを用いたOFDM復調器の製作

Development of OFDM Demodulator using FPGA

T5-36　山口雄大

指導教員　髙﨑和之

1．はじめに

流星バースト通信に直交周波数分割多重方式（Orthogonal Frequency Division Multiplexing）を用いると，ビット誤り率を低減できることが知られている[1]．先行研究[1]では，PCを用いてOFDMの復調が行われていたが，リアルタイムの復調は行われていなかった．本研究ではFPGAを用いて，OFDMのリアルタイム復調を行い，その性能を評価した．

2．理論

2．1　直交周波数分割多重方式（OFDM）

OFDMは多数の直交した周波数の搬送波を用いて，並列伝送する二次変調方式の一つである[2]．変調時は送信信号を逆離散フーリエ変換（Inverse Discrete Fourier Transform）し，復調時は離散フーリエ変換（Discrete Fourier Transform）する．変復調にDFTやIDFTを用いるため，複数の搬送波を扱うマルチキャリア方式の中でも，ディジタル信号処理との親和性が高く，回路規模やコストの面でも優れている．

　本研究に用いるOFDMの信号仕様を表1に示す．この信号仕様は実験装置の仕様に合わせて決定した．OFDM信号は同一のOFDMシンボルを9回送信，1回休止を1セットとし，10セット繰り返したものを用いた．

表1　OFDMの信号仕様

|  |  |
| --- | --- |
| 一次変調方式 | BPSK |
| 二次変調方式 | OFDM |
| 帯域幅[Hz] | 984.375~5671.875 |
| サブキャリア数 | 101（パイロット信号含む） |
| サブキャリアの振幅 | 1 |
| サブキャリア間隔[Hz] | 46.875 |
| パイロット信号数 | 5（一定間隔で挿入） |
| パイロット信号の振幅 | 2 |
| パイロット信号の位相[] | 0 |
| 1シンボルの時間[ms] | 21.3 |
| データ量[Byte] | 12 |

2．2　高速フーリエ変換（Fast Fourier Transform）

FFTはDFTを高速で行うアルゴリズムであり，入力信号を，入力信号のサンプル数を*N*とした場合，離散フーリエ係数は式(1)で与えられる．

　DFTを式（1）に忠実に計算した場合，計算量はとなる．FFTは入力信号の添え字を偶数と奇数の二つのグループに分割し，分割統治法を用いることで，入力信号のサンプル数*N*が2の冪乗のとき，計算量をに削減するアルゴリズムである[3]．分割・計算手順を図に示すと，図1に示すように蝶のような形になることからバタフライ演算とも呼ばれる．図1中のは加算を意味し，矢印は矢印の下の数との乗算を意味している．バタフライ演算は，演算の過程でデータの順序が変化するため，入力か出力のどちらかのインデックスをビット逆順にする必要がある．本研究では，入力側のインデックスをビット逆順にして考える．は時間領域のインデックスにあたることから，この手法は時間間引き法と呼ばれている．

バタフライ演算は式(2)に示す演算を行うハードウェアのバタフライ演算機を用いることで，効率よく計算できる．ここで，はバタフライ演算器の入力，はバタフライ演算機の出力，*W*は回転因子と呼ばれ，である．

FFTはバタフライ演算の最終段以外はインデックス*N*/2を境に，上下で独立して演算可能である．本研究では，バタフライ演算器を二つ搭載することで，演算速度の向上を図った．FFTの演算用のRAMが一つの場合，二つのバタフライ演算器が同時にRAMにアクセスすると待ち時間が発生してしまう．そのため，演算用のRAMもインデックス*N*/2を境に，二つ搭載した．回転因子はROMに保存したsinテーブルを用いた．バタフライ演算器を一つから二つにした結果，*N*=1024では演算速度が約1.81倍向上した．

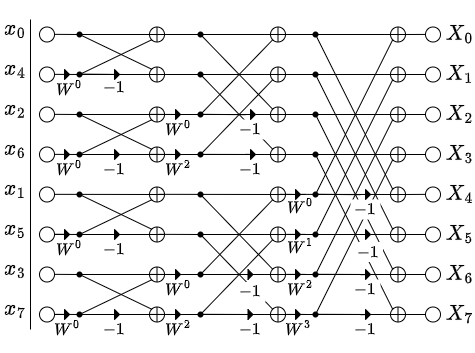
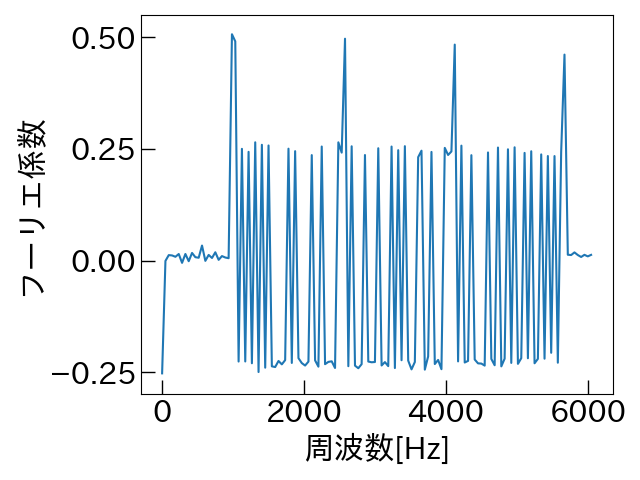


図1　バタフライ演算

3．FPGAを用いたOFDM復調器

FPGAを用いて製作した復調器の仕様を表2に示す．

表2　復調器の仕様

|  |  |
| --- | --- |
| FPGA | GW1NR-9(Gowin) |
| 評価ボード | Tang Nano 9K(Sipeed) |
| 10ビットADC | MCP3002(Microchip) |
| FPGAの動作周波数[MHz] | 24 |
| サンプリング周波数[kHz] | 48 |
| ADCのクロック周波数[kHz] | 800 |
| FFTのサンプル数*N* | 1024 |

3．1　FPGAの動作

ADCのサンプリング結果用のRAMをRAM\_ADC，FFTのインデックスが0~(*N*/2-1)用のRAMをRAM\_FFT0，*N*/2~(*N*-1)用のRAMをRAM\_FFT1とする．FPGAはサンプリング周波数48kHzで連続して電圧のサンプリングを行い，サンプリング結果はRAM\_ADCに保存する．RAM\_ADCを監視し，閾値を超えたらOFDMシンボルのスタート点と判断する．スタートの瞬間を含めて1024サンプルが完了した後，RAM\_ADCの内容をRAM\_FFT0とRAM\_FFT1に転送する．ここで，RAM\_ADCのインデックスをビット逆順したものを，RAM\_FFT0または1に書き込む．バタフライ演算器を用いてFFTを行い，FFT完了後は演算結果の符号を用いてBPSKを復調して，ビット列の最初と最後のデータが0x55である場合，復調成功と判断する．成功時は結果の12バイトをシリアル通信で送信する．失敗時は，スタート検出の際に，検出位置がわずかにずれてしまった可能性があるため，RAM\_ADCのデータを一つシフトして一連の処理を試行する．シフトを5回繰り返して成功しない場合は，失敗と判断する．復調に成功した場合，同じOFDMシンボルを繰り返し送信しているため，次のOFDMシンボルは前回の検出位置から1024サンプル離れた場所から始まることを利用して，復調する．復調器の結果の出力方法には二種類のモードを用意した．一つ目は連続で信号の復調を行い，結果のみをシリアル通信で送信するモードである．二つ目はデバッグや性能評価を目的とした，1回分のOFDMシンボルのみを復調する代わりに，FFTの結果をシリアル通信で送信するモードである．パイロット信号を用いて振幅の補正を行う予定だったが，補正には除算器が必要であり，除算器の作成が間に合わなかったため，パイロット信号を用いた補正は現時点では行っていない．

3．2　演算時間の評価

　PC上で生成したOFDM信号をオーディオインターフェイスを介して製作した復調器に入力し，復調器で計算したフーリエ係数を図2に示す．

図2より，パイロット信号の振幅が0.5程度であること，サブキャリアの振幅が0.25程度であり，その比が2:1で

図2　復調器で計算したフーリエ係数

あることから，正しく演算できていると考えられる．0Hzのフーリエ係数が0でないことから，DCバイアス等の雑音があると考えられる．

サンプリング終了後，RAM\_ADCからRAM\_FFTへのデータ転送に42.8s，FFTの計算に587s，BPSK及び符号判定処理に4.46s，合計634sでOFDMの復調処理を行うことができた．シフト回数は最大5回であるため，最大シフト時の処理時間は3.17msとなり，OFDMのシンボル時間（21.3ms）内に処理が終わることを確認した．

また，使用したFPGAのリソースは表3に示すとおりであり，他の処理を組み込む余裕があることも確認できた．

表3　使用したFPGAのリソース

|  |  |
| --- | --- |
| Register | 834 / 6480 (12.9%) |
| LUT4 | 2095 / 8640 (24.2%) |
| 16Kbit BSRAM | 8 / 26 (30.8%) |
| MULT18X18 | 8 / 20 (40%) |

4．考察と今後の課題

　目標のリアルタイム復調は，シリアル通信による伝送時間を含めずに考えた場合，OFDMの1シンボルの3%程度の時間で復調できているため，目標を達成できていると考えられる．しかし，製作した復調器は相関を用いた同期処理を行っていないため，途中から受信されたOFDMシンボルを復調することができない．実際の環境では信号が途中から受信されることは十分起こり得るため，この機能の実装が必要である．また，パイロット信号を用いた補正を行っていないため，振幅の減衰や位相のずれに弱くなっている．実際の流星バースト通信環境で復調を行うためにはこれらの機能の実装が今後の課題である．

参考文献

[1] 髙﨑和之，若林良二，亀井利久，高塚徹，三寺史夫，“OFDMを用いた流星バースト通信に関する検討”， 信学ソ大，2016年，B-1-19.

[2] 伊丹誠，“OFDMの基礎と応用技術”，電子情報通信学会Vol. 1, No.2 (2007), pp.35-43.

[3] James W. Cooley, John W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series”, Mathematics of Computation, Vol.19, No.90(1965), pp.297-301.