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x_n，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X_kは式(1)で与えられる．

$$X_k = \sum_{n=0}^{N-1} x_n \ e^{-j\frac{2\pi}{N}nk} \qquad (1)$$

DFT を式 (1) に忠実に計算した場合，計算量は$O(N^2)$となる．FFT は入力信号x_nの添え字n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$O(NlogN)$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nをビット逆順にして考える．n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，x_a，x_bはバタフライ演算器の入力，X_a，X_bはバタフライ演算機の出力，Wは回転因子と呼ばれ，$W \equiv e^{-j\frac{2\pi}{N}}$である．

$$\begin{cases} X_a = x_a + W^k x_b \\ X_b = x_a - W^k x_b \end{cases} \qquad (2)$$

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．

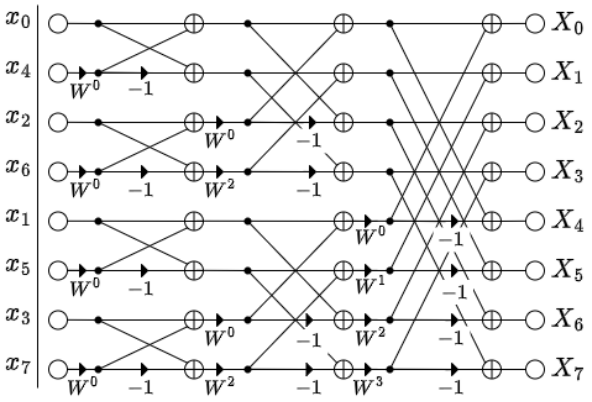

図 1　バタフライ演算

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

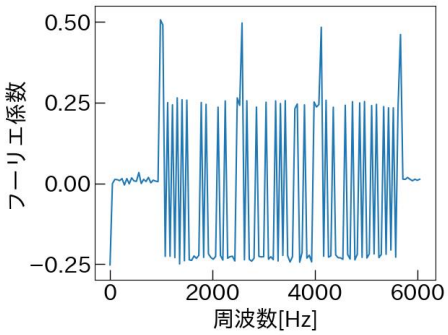

図 2　復調器で計算したフーリエ係数

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 42.8μs，FFT の計算に 587μs，BPSK 及び符号判定処理に 4.46μs，合計 634μs 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.