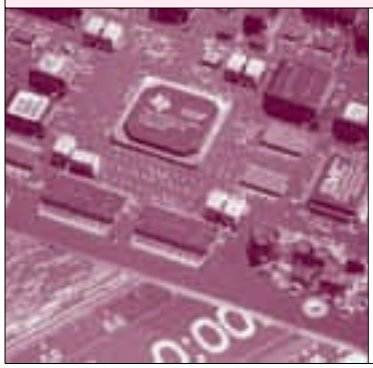


第4章　シミュレーションや DSPスターター・キットで学ぶ

FIRフィルタの基礎とフィルタ・プログラミング

山口　晶大
Akio Yamaguchi

本章では，簡易なディジタル・フィルタの設計手法を紹介します．非常に直截な手法で設計しますが，DSKを使った実験を行うには十分な特性です．

一般のフィルタ設計ソフトウェアで使われているような，仕様を満たす最も次数の小さい(タップ長の小さい) フィルタ係数を求める，という最適設計は行いません．しかし，数式を使わずにFIRフィルタ係数とフーリエ変換の性質の議論だけから設計手法を導き出しているので，理解しやすいと思います．

第4章では大きく分けて次の三つを取り上げます．

- FIRフィルタの基礎と構造
- FIRフィルタ係数の設計手順
- 実用的なFIRフィルタ設計方法

さらに本文中に出てくる実験課題を体験していただければ，一層理解が深まると思います．

なお本文中で取り上げるCCS用のFIRフィルタ設計プログラム**fir001**は，DSKがなくても，評価版CCS(付録CD-ROMに収録)のシミュレーション機能を利用して実行することが可能です．本文中の実験課題3，5，6，7についても評価版CCSで体験できます．

シミュレータ上でプログラムを実行する場合の評価版CCS環境設定の方法については章末のコラムを参照ください．

FIRフィルタの基礎と構造

ディジタル・フィルタにはFIR(Finite Impulse Response)フィルタとIIR(Infinite Impulse Response)フィルタの2種類があります．IIRフィルタはアナログのアクティブ・フィルタと同様にフィードバック処理を使っているため，FIRフィルタと比較すると設計が少々複雑です．

ここではより扱いの簡単なFIRフィルタを取り上げます．希望の特性のフィルタを実現するにはフィルタ係数をどのようにして求めれば良いかという設計の話は少し後回しにして，まずFIRフィルタの構造についての説明から始めたいと思います．

● 入出力の関係式とシグナル・フロー・グラフ

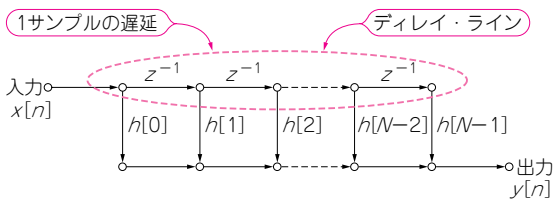
FIRフィルタの入力信号$x[n]$と出力$y[n]$の関係は次式のように表されます．

$$y[n] = \sum_{i=0}^{N-1} h[i]\, x[n-i]$$
$$= h[0]\, x[n] + h[1]\, x[n-1] + h[2]\, x[n-2] + \cdots + h[N-2]\, x[n-N+2] + h[N-1]\, x[n-N+1] \quad \cdots\cdots (1)$$

ただし，$[n]$：フィルタ係数，N：フィルタ次数

上式のFIRフィルタの演算をシグナル・フロー・グラフで表すと，**図1**のようになります．**図1**に示すようにFIRフィルタはディレイ・ラインの中間タップの出力にフィルタ係数を乗ずる処理として実現できます．

〈図1〉FIRフィルタのシグナル・フロー・グラフ



Keywords
体験版CCS，DSK，DSPスターター・キット，FIRフィルタ，タップ長，シミュレータ，Finite Impulse Response，FIR，IIR，Infinite Impulse Response，シグナル・フロー・グラフ，フィルタ係数，リング・バッファ，フーリエ変換，振幅スペクトル，バンドパス・フィルタ，ローパス・フィルタ，ハイパス・フィルタ，サンプリング周波数，BPF，LPF，HPF，ホワイト・ノイズ，FFTアナライザ，WaveGene，WaveSpectra，カットオフのパラメータ，フィルタ・スケーリング，ゲート・タイム，周波数カウンタ，窓関数，ハニング窓，三角窓，ハミング窓．

▶**図1**を別の図で表現する

FIRフィルタのシグナル・フロー・グラフの表現はただ一つではありません．表現を変えた**図2**のシグナル・フロー・グラフを見れば，FIRフィルタの演算のしくみをよく理解できると思います．

図1と**図2**はまったく等価なシグナル・フロー・グラフですが，**図1**中のディレイ(遅延)の値はすべて1であるのに対して，**図2**では各遅延の値がみな異なります．

▶**図1**と**図2**をさらにわかりやすく表現する

図2に示すシグナル・フローのFIRフィルタ演算をさらにわかりやすく説明したものが**図3**になります．

図3からFIRフィルタにインパルス(単発パルス)を入力したときの出力，すわなちインパルス・レスポンスはフィルタ係数そのものとなることがわかります．

● **リング・バッファを使ったフィルタ・プログラム**

FIRフィルタ演算のプログラムは，第3章で説明したリング・バッファを使えば効率的に実現できます．リング・バッファを使ってC言語で制作したFIRフィルタ関数を**リスト1**に示します．

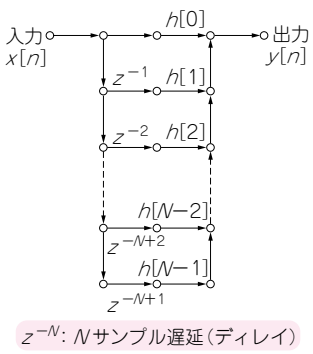
関数**fir()**には五つの引き数があり，詳細を**表1**に示します．この関数は1回呼ばれてから，入力信号xの値を1サンプルずつ入力するたびに，計算したFIRフィルタ出力の値を一つ返します．

▶インパルス・レスポンスの逆転に注意

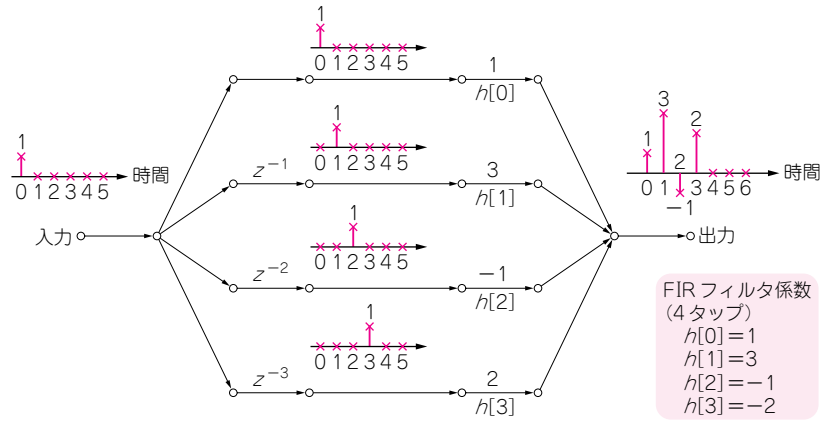
リスト1の関数で注意が必要なのはリング・バッファを使ったFIRフィルタ演算の手順が**図1**や式(1)の定義とは少し異なっていることです．**図1**や式(1)の定義どおりの演算は**図4**のように表されますが，**リスト1**のFIRフィルタ関数は**図5**に示す手順で演算を行っています．

二つの違いは，**リスト1**の関数**fir()**ではインパルス・レスポンスが，フィルタ係数**h[]**ではなく，フ

〈図2〉図1と等価なFIRフィルタのシグナル・フロー・グラフ



〈図3〉図2に示すシグナル・フローのFIRフィルタ演算の詳細



〈表1〉FIRフィルタ関数fir()の引き数の説明

引き数	説　明
float x	FIRフィルタ入力信号
float buf[]	リング・バッファ(1次元配列**buf[0]**～**buf[length-1]**)
float h[]	フィルタ係数(1次元配列**h[0]**～**h[length-1]**)
unsigned int *ix	リング・バッファ書き込み/読み出しのためのポインタとして使っている変数
unsigned int length	フィルタ・タップ長

〈リスト1〉FIRフィルタ関数fir()(%は除算の余りを求める演算子)

```
float fir(float x, float buf[], float h[], unsigned int *ix, unsigned int length) {
  float        sum;
  unsigned int i;

  buf[*ix]=x;                              // write ring buffer
  *ix=(*ix+1)%length;                      // update index
  sum=0.0;                                 // clear sum
  for (i=0; i<length; i=i+1) {             // FIR filter calcualtion
    sum=sum+h[i]*buf[(*ix+i)%length];      // multiply and accumlate
  }
  return sum;                              // return FIR filter output
}
```