卒 業 研 究 報 告

題　　目

VHDL によるデジタルフィルタの実現と動作検証

指 導 教 員

橘昌良助教授

報 告 者

中西 潤

平成　14 年　2 月　　4 日

高知工科大学 電子･光システム工学科

目次

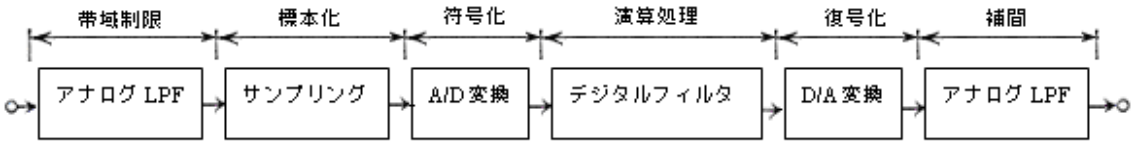
１．はじめに

私たちが世の中で扱う情報には音声や画像、その他様々なものがあるがそれらは大抵アナログ情報である。
一般に、アナログ信号の情報を処理するにはアナログ的に処理するのが望ましいがアナログ信号は伝送時における雑音・歪み等の問題で容易ではない。そこでアナログ信号をデジタル化し信号処理を行う方法が考えられるようになった。
この研究ではそのデジタル信号処理技術のなかでも基礎であり重要な要素である FIR・IIR デジタルフィルタについてソフトウェア・ハードウェア両面における特性・動作を検証しそれぞれがどのような用途に適しているかを考えるのが目的である。

流れとしては、ＦＩＲフィルタとＩＩＲフィルタの概念を説明し、次に実際に設計値を決めて低域通過フィルタの動作検証を行い最後に高次数･高サンプリング周波数時のフィルタの動作について確認しそれぞれの最適用途について考察を行う。

２．デジタル信号処理の特徴

“信号処理”とはろ波・増幅・雑音除去・信号発生・信号検出・信号の特徴抽出などのことを意味する。今回はその中でも信号の特徴抽出・雑音除去等の役目をするデジタルフィルタについて取り扱う。
一般的にアナログ信号をデジタル化してフィルタリングするというのは下記のような一連の流れのことを言う



入力アナログ信号を A/D 変換器で標本化及び量子化してデジタル信号（時間軸上・振幅軸上も離散的な信号）を得る。このデジタル信号を所要のフィルタリング処理を行いその出力を D/A 変換器によってアナログ信号に戻す。これが一般的なデジタル信号処理の流れである。
デジタル信号を処理する方法としてはソフトウェアによる演算によってフィルタ処理を行うのと NMOS、CMOS や LSI などのハードウェア素子を演算子に利用する方法がある
ここでは基本的なフィルタである FIR と IIR フィルタを検証することでデジタルフィルタの基本的な特性を調べる。

３．FIR フィルタ

3－1．FIRフィルタの性質

FIR フィルタは

$$y_k = \sum_{m=0}^{M} a_m x_{k-m}$$

のように、現在及び過去の入力値 x_{k-m} のみで現在の出力 y_k が求められるものであり、インパルス応答 $\{h_m\}_{m=0}^{\infty}$ は係数 a_m と同じと表現され

$$h_m = \begin{cases} a_m; 0 \le m \le M \\ 0; M < m \end{cases}$$

となる。その伝達関数 $H(z)$ は

$$H(z) = \sum_{m=0}^{M} a_m z^{-m} = \sum_{m=0}^{M} h_m z^{-m}$$

となる。

FIR フィルタの周波数特性は $z = e^{j\omega T}$ を伝達関数の式に代入することにより

$$H(e^{j\omega T}) = H(z)|_{z=e^{j\omega T}} = \sum_{m=0}^{M} a_m e^{-jm\omega T} = \sum_{m=0}^{M} a_m \left[\cos(m\omega T) - j\sin(m\omega T)\right]$$

$$= \sum_{m=0}^{M} a_m \cos(m\omega T) - j\sum_{m=1}^{M} a_m \sin(m\omega T)$$

という関係が成立する。ここで

$$H_R(e^{j\omega T}) = \sum_{m=0}^{M} a_m \cos(m\omega T)$$

$$H_I(e^{j\omega T}) = -\sum_{m=1}^{M} a_m \sin(m\omega T)$$

と実数部と虚数部に分けられる。この場合振幅特性と位相特性はそれぞれ

$$\text{振幅特性：} |H(e^{j\omega T})| = \sqrt{\{H_R(e^{j\omega T})\}^2 + \{H_I(e^{j\omega T})\}^2}$$

$$\text{位相特性：} \angle H(e^{j\omega T}) = \tan^{-1}\left[\frac{H_I(e^{j\omega T})}{H_R(e^{j\omega T})}\right]$$

と表現される。これはそれぞれ入出力の正弦波の振幅比と位相差を表している

振幅特性は $20\log_{10}|H(e^{j\omega T})|$ $[dB]$ のデシベル表記を用いることもある。さらに-の符号を

つけて減衰量表記とすることもある

3－2. FIR フィルタの設計法

ここでは実際に検証に使用するフィルタ設計の手順を説明する

FIR フィルタには

・　直線位相特性を実現できるので位相の歪が生じない

・　安定性が常に保証されているので乗算係数の設定にフィルタの安定を考慮する必要がない

という特性がある

そこで、希望する周波数特性を IDFT を用いて直接近似し設計する方法をとる

3－2－1. 直線位相 FIR フィルタの設計

概略説明

次数2M の FIR フィルタのインパルス応答 $h(nTs)$ を MTs を中心とする線対称

$$h(nTs)=\begin{cases} h\{(2M-n)Ts\} & (n=0,1,2,\cdots\cdots,2M) \\ 0 & (n<0,n>2M) \end{cases} \quad \cdots\cdots\text{A}$$

とすると $h(nTs)$ のz変換は

$$H(z)=Z[h(nTs)]$$

$$=\sum_{n=0}^{2M} h(nTs)z^{-n}$$

$$=\sum_{n=0}^{M-1} h(nTs)z^{-n}+h(MTs)z^{-M}+\sum_{n=M+1}^{2M} h(nTs)z^{-n}$$

$$=\sum_{m=1}^{M} h\{(M-m)Ts\}z^{-(M-m)}+h(MTs)z^{-M}+\sum_{m=1}^{M} h\{(M+m)Ts\}z^{-(M+m)}$$

$$=\alpha(0)z^{-M}+\sum_{m=1}^{M}\alpha(mTs)\{z^{-(M-m)}+z^{-(M+m)}\}$$

$$=\left\{\alpha(0)+\sum_{m=1}^{M}\alpha(mTs)(z^{m}+z^{-m})\right\}z^{-M}$$

となる。ここで

$$h\{(M+m)Ts\}=h\{(M-m)Ts\}=\alpha(mTs) \quad (m=0,1,\cdots,M)$$

としている

このフィルタのシステム関数$H(\omega)$は

$$H(\omega) = H(z)\,|_{z=e^{j\omega Ts}}$$
$$= \left\{ \alpha(0) + 2\sum_{m=1}^{M} \alpha(mTs)\cos m\omega Ts \right\} e^{-j\omega MTs}$$
$$= \left| \alpha(0) + 2\sum_{m=1}^{M} \alpha(mTs)\cos m\omega Ts \right| \angle -\omega MTs$$

これは、システム関数$H(\omega)$が周期$2\pi/Ts$で繰り返す周期関数であることをあらわしている
また、この場合位相は直線位相となる。
希望する周波数特性を$H_d(\omega)$とすると直線位相 FIR フィルタの最適な乗算係数$\alpha(mTs)$は
平均 2 乗誤差

$$\mathrm{E} = \frac{Ts}{2\pi} \int_{-\frac{\pi}{Ts}}^{\frac{\pi}{Ts}} \left| H_d(\omega) - H(\omega) \right|^2 d\omega$$

を乗算係数$\alpha(mTs)$で偏微分して零に等しくすると

$$\alpha(mTs) = \frac{Ts}{2\pi} \int_{-\frac{\pi}{Ts}}^{\frac{\pi}{Ts}} \left| H_d(\omega) \right| e^{jm\omega Ts} d\omega$$

となる。乗算係数$\alpha(mTs)$は実数であるので、振幅特性は偶関数

$$A(-\omega) = A(\omega)$$

位相特性は奇関数

$$\phi(-\omega) = \phi(\omega)$$

となる。ここでそれぞれ

$$A(\omega) = \left| \alpha(0) + 2\sum_{m=1}^{M} \alpha(mTs)\cos m\omega Ts \right|$$

$$\phi(\omega) = -\omega MTs$$

である

まず、$0 \le f \le 1/Ts$の周波数帯域において、希望する振幅特性$\left|H_d(\omega)\right|$を N 等分すると

$$\left|H_d(kf_0)\right| = \left|H_d(\omega)\right|\,|_{\omega=\frac{2\pi}{NTs}k} \quad (k = 0,1,\cdots,N-1)$$

となる。このとき$f_0 = 1/(NTs)$である。そして$\omega = 2\pi\kappa/(NTs)$とおくと最適な乗算係数は

$$\alpha(mTs) = \frac{1}{N}\sum_{k=0}^{N-1} \left|H_d(kf_0)\right| e^{j\frac{2\pi}{N}km} \quad (m = 0,1,\cdots,M)$$

で求められる。これが IDFT によって最適な乗算係数を求めるときに使用する式となる

このとき、乗算係数$\alpha(mTs)$は実数であるので振幅特性と位相特性には

$$A\{(N-k)f_0\} = A(kf_0) \quad (k = 0,1,\cdots, N-1)$$
$$\phi\{(N-k)f_0\} = -\phi(kf_0) \quad (k = 0,1,\cdots N-1)$$

という関係が成り立つ。また$M+1$個の乗算係数を決定するために、分割数は

$$N \geq 2M+1$$

とならなければならない

4. IIR フィルタ

4－1. IIRフィルタの性質

IIR フィルタは、FIR フィルタの一般式

$$y_k = \sum_{m=0}^{M} a_m x_{k-m}$$

と比較して、過去の出力値 y_{k-n} が現在の出力 y_k の計算に影響を与えるような

$$y_k = \sum_{n=1}^{N} b_n y_{k-n} + \sum_{m=0}^{M} a_m x_{k-m} \quad \cdots\cdots\cdots \mathrm{B}$$

という式であらわされる

また、IIR フィルタの伝達関数 $H(z)$ は B の両辺を z 変換して

$$H(z) = \frac{A(z)}{B(z)} \qquad ; A(z) = \sum_{m=0}^{M} a_m z^{-m}$$

$$; B(z) = 1 - \sum_{n=1}^{N} b_n z^{-n}$$

となる、これは有理関数（分母と分子がそれぞれ多項式で構成されているもの）とよばれるものである

有理関数となったことで特に IIR フィルタの安定性に問題が出てくる

ここで $A(z) = 0$ となる z の値をフィルタの零点、$B(z) = 0$ となる z の値をフィルタの極という

一般に IIR フィルタは低い次数で急峻なフィルタを作成するのに適しているといわれているが

FIR フィルタが絶対安定なのにたいして IIR フィルタではその安定性が問題となる

4－1－1. IIR フィルタの安定性

ここでは IIR フィルタの安定性解析のために1次の差分方程式

$$y_k = b_1 y_{k-1} + x_k \quad ; b_1 \neq 0$$

を考える

入力 x_k を単位インパルス σ_k、$y_{-1} = 0$ として、出力系列（インパルス応答）を求める。なお、

単位インパルスは

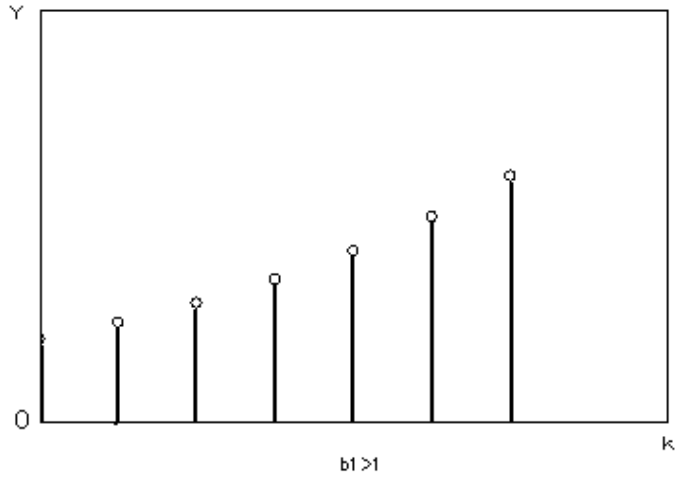
$$x_k = \sigma_k = \begin{cases} 1 & ; k = 0 \\ 0 & ; k \neq 0 \end{cases}$$

である。計算すると

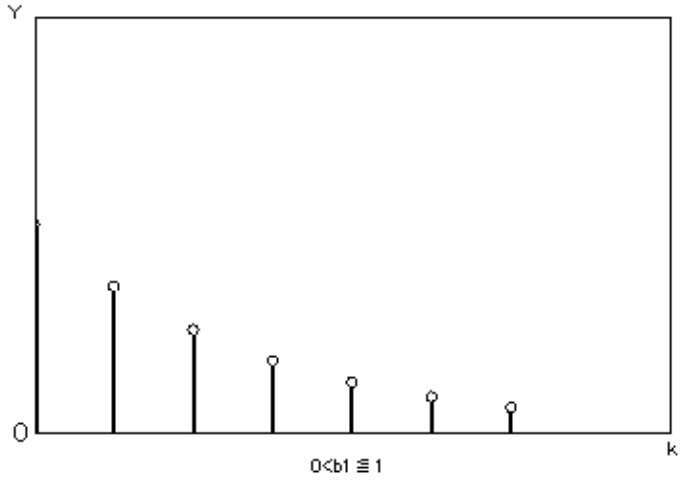
$$y_k = b_1^{\,k}$$

という一般式が求まる

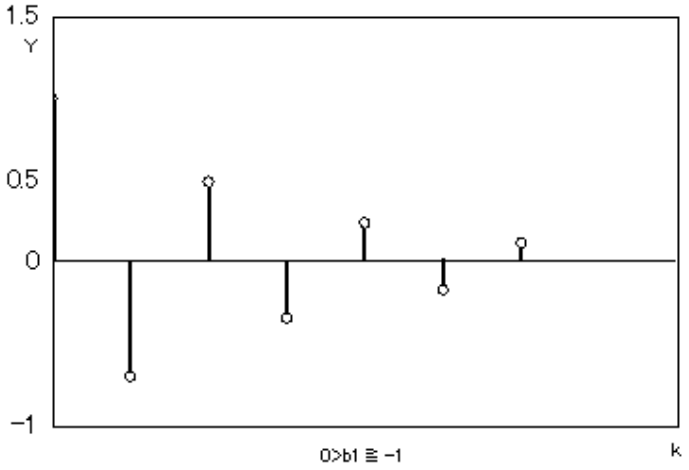
b_1 の値による出力系列の違いは以下のようになる



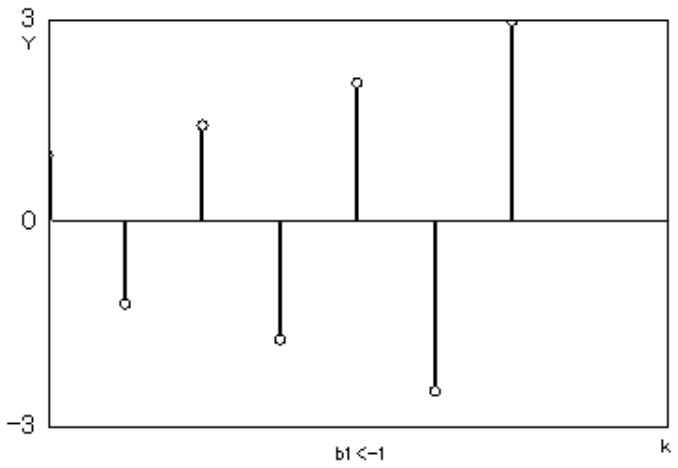
b1 >1



0<b1 ≦ 1



0>b1 ≧ −1



これはb_1の値によって出力系列が発散したり収束したりしている。IIR ではこのように FIR とちがってインパルス応答が常に収束するとは限らないことがわかる。
IIR フィルタの安定性の条件は

$$|b_1| < 1$$

であった場合にインパルス応答が0に収束するのでこれが安定条件となることがわかる
また、1次のフィルタの伝達関数は

$$H(z) = \frac{1}{1 - b_1 z^{-1}}$$

となり分母の多項式$B(z) = 1 - b_1 z^{-1} = 0$となる$z$の値を$z_p$としたら

$$z_p = b_1$$

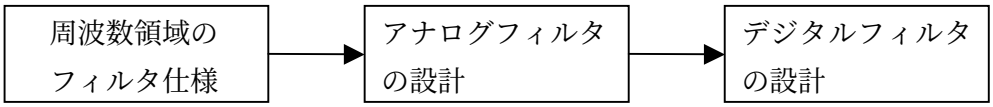
となる、よって一次 IIR フィルタの安定条件は

$$|z_p| < 1$$

となる。これは「極の絶対値が1より小さい」いいかえると極の乗算係数値b_nの絶対値の総計が1より小さいと安定するということである

4－2. IIR フィルタの設計法

今回の場合は IIR フィルタの設計には「最初にアナログフィルタの設計を行いそこよりデジタルフィルタを作成する」間接設計を利用する
その流れは



となる。まず、アナログフィルタには主に

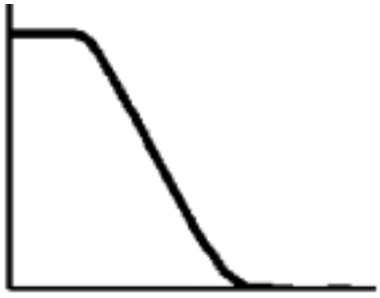
- バタワース
- チェビシェフ
- 逆チェビシェフ
- 連立チェビシェフ（楕円）

などがあり、通常はこのなかよりどれかひとつを使用してアナログフィルタの設計を行う

次に、アナログフィルタの伝達関数 $H(s)$ をもとにデジタルフィルタの伝達関数 $H(z)$ を求める
このとき $H(s) \rightarrow H(z)$ の変換方法には

- 標準 z 変換
- 双一次変換
- 整合 z 変換

の3つがあげられる（これを s-z 変換と呼ぶ。）
実際の設計の流れには、ここではバタワースアナログフィルタを利用する
まず、バタワースアナログフィルタの特性を示す



特徴としては通過域と阻止域が滑らかである（リップルが存在していない）ということである
このような状態を最大平坦特性という
左図のような特性を満たすフィルタを作成する場合、
バタワースフィルタの設計には

$$|H(j\Omega)|^2 = \frac{1}{1+\left(\frac{\Omega}{\Omega_c}\right)^{2N}} \cdots\cdots \mathrm{C}$$

という式が用いられる。これはバタワースフィルタ設計の伝達関数を求める式であり「振幅二重特性」といわれる N はフィルタの次数をあらわし Ω_c はカットオフ周波数である。

ここでカットオフ周波数を $\Omega_c = 1rad/\sec$ とした場合のローパスフィルタを特に「プロトタイプフィルタ」と呼ぶ。このプロトタイプフィルタから式変換を行うことによって基本となるアナログフィルタを作成できる

実際に $\Omega_c = 1rad/\sec$ を代入し、$s = j\omega$　より $\omega = s/j$ を式 C に代入すると

$$|H(j\Omega)|^2 = \left|\frac{1}{1+(-1)^N s^{2N}}\right|_{s=j\Omega}$$

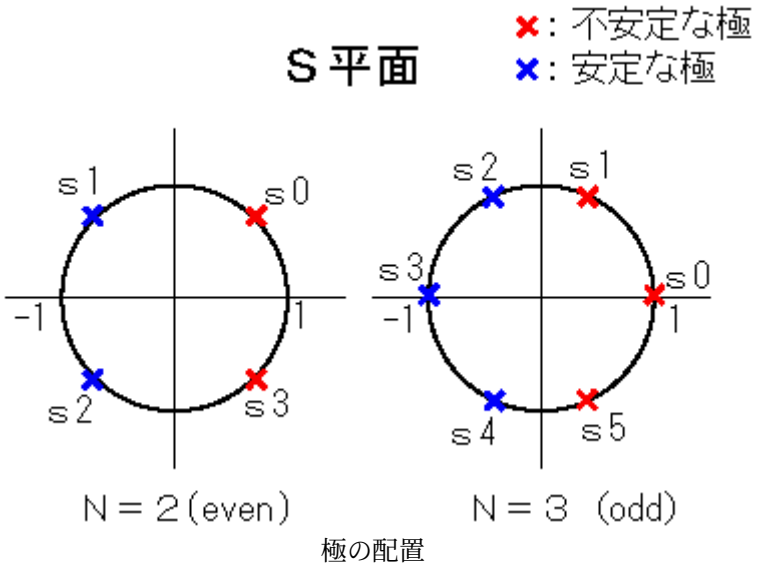
という式になる。上式の分母が0の根（フィルタの極 s_k ）は s 平面の単位円上に配置される。

その位相は

$$\phi_k = \begin{cases} \dfrac{k\pi}{N} & (N = \text{奇数}) \\ \dfrac{\left(k+\frac{1}{2}\right)\pi}{N} & (N = \text{偶数}) \end{cases} \qquad k = 0,1,\cdots 2N-1$$

という式で与えられる

この関係式より s 平面の極配置を行うと s 平面の右半面に極が配置された場合は不安定な根となるので極が s 平面の左半面にあるように選び伝達関数をもとめるとシステムとして安定したバタワースフィルタをもとめることができる



極の配置

ここで N=3 のときのプロトタイプフィルタの伝達関数を求めてみる

極配置は上図より左半面の s2,s3,s4 となり、それは

$$s_2 = -\frac{1}{2} + \frac{\sqrt{3}}{2} j$$

$$s_3 = -1 + (0\,j)$$

$$s_4 = -\frac{1}{2} - \frac{\sqrt{3}}{2} j$$

という式で与えられる。したがってこれより伝達関数は

$$H(s) = \frac{1}{(s+1)\left(s + \frac{1}{2} + \frac{\sqrt{3}}{2} j\right)\left(s + \frac{1}{2} - \frac{\sqrt{3}}{2} j\right)}$$

$$= \frac{1}{(s+1)(s^2 + s + 1)}$$

となる

上の例をもとに各次数におけるプロトタイプフィルタの伝達関数を求めると

次数N	バタワースフィルタの伝達関数
1	$\frac{1}{s+1}$
2	$\frac{1}{(s^2 + \sqrt{2}s + 1)}$
3	$\frac{1}{(s^2 + s + 1)(s+1)}$
4	$\frac{1}{(s^2 + 0.7653s + 1)(s^2 + 1.848s + 1)}$
5	$\frac{1}{(s+1)(s^2 + 0.6180s + 1)(s^2 + 1.618s + 1)}$
6	$\frac{1}{(s^2 + 0.5176s + 1)(s^2 + \sqrt{2}s + 1)(s^2 + 1.932s + 1)}$

という式になる

これより、アナログバタワースフィルタの一般式をもとめると

N：偶数の場合

$$H_N(s)=\prod_{k=1}^{N/2}\frac{1}{s^2+2\cos(\phi_k)s+1}$$

$$\phi_k=\left[(2k-1)\pi/2N\right]\quad(k=1,2,\cdots,N/2)$$

N：奇数($N\geq 3$)の場合

$$H_N(s)=\frac{1}{s+1}\prod_{k=1}^{(N-1)/2}\frac{1}{s^2+2\cos(\phi_k)s+1}$$

$$\phi_0=0,\qquad \phi_k=\left[k\pi/N\right]\quad(k=1,2,\cdots,(N-1)/2)$$

この式はカットオフ周波数（遮断周波数ω_0）を$1rad/\sec$とした式なので一般式は

N：偶数の場合

$$H_N(s)=\prod_{k=1}^{N/2}\frac{1}{s^2+2\cos(\phi_k)(s/\omega_0)+1}$$

$$\phi_k=\left[(2k-1)\pi/2N\right]\quad(k=1,2,\cdots,N/2)$$

N：奇数($N\geq 3$)の場合

$$H_N(s)=\frac{1}{s+1}\prod_{k=1}^{(N-1)/2}\frac{1}{s^2+2\cos(\phi_k)(s/\omega_0)+1}$$

$$\phi_0=0,\qquad \phi_k=\left[k\pi/N\right]\quad(k=1,2,\cdots,(N-1)/2)$$

となる。その位相特性は

$$\angle H_N(j\omega)=-\tan^{-1}\left[\frac{I_m\{B_N(j\omega)\}}{R_e\{B_N(j\omega)\}}\right]$$

という式で与えられる

振幅二重特性と通過域、阻止域双方の許容減衰量より

通過域 $\left(\frac{\omega_p}{\omega_0}\right)^{2N}=10^{A_p/10}-1\cdots\cdots\mathrm{D}$

阻止域 $\left(\frac{\omega_r}{\omega_0}\right)^{2N}=10^{A_r/10}-1\cdots\cdots\mathrm{E}$

となる

これを N と ω_0 との連立方程式とすると

$$\left(\frac{\omega_r}{\omega_p}\right)^{2N} = \frac{10^{A_r/10}-1}{10^{A_p/10}-1}$$

が得られ、これの対数をとることにより次数 N を決定する式

$$N = \frac{\log\left[\left(10^{A_r/10}-1\right)/\left(10^{A_p/10}-1\right)\right]}{2\log\left(\omega_r/\omega_p\right)}$$

が与えられる。このとき計算で出された値の小数点以下の端数はきりあげられ次数 N は整数となる。
式 D,E と次数 N より遮断周波数 ω_0 を算出する

$$\frac{\omega_p}{\left(10^{A_p/10}-1\right)^{1/2N}} = \omega_0{}^{(p)}$$

$$\frac{\omega_r}{\left(10^{A_r/10}-1\right)^{1/2N}} = \omega_0{}^{(r)}$$

$$\omega_0{}^{(p)} \le \omega_0 \le \omega_0{}^{(r)}$$

今回は $\omega_0 = \frac{\omega_0{}^{(p)} + \omega_0{}^{(r)}}{2}$ として扱うことにする
次にアナログフィルタから S-Z 変換を用いてデジタルフィルタの伝達関数 $H(z)$ を導く
ここでは双一次変換を利用する
双一次変換とは直接 z 変換を行うのではなく、ω をいったん $\omega s/2$ 以内の周波数に変換して（それを s1とする）それをz変換するこのためエリアジングによる誤差は発生しなくなる

$$s = \frac{2}{T}\tanh\left(\frac{s_1 T}{2}\right)$$

S → s1 → Z

$$e^{-s_1 T} = z^{-1}$$

$$s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$$

5. 研究内容

FIR と IIR フィルタには下記のような特性的差異がある

	FIR	IIR
システム応答	有限時間	無限時間
フィードバック	なし	あり
誤差	累積しない	累積する
安定	絶対安定	条件により変動
位相特性	線形位相	近似的
次数	大きい	小さい

上記の表によると FIR のほうが安定して動作し、フィルタの誤差もないので有用とおもわれるが
実際にはサンプリング周波数が低いとき（CDDA や DVDAUDIO クラス）でしか FIR フィルタは
用いられることがなく、サンプリング周波数が高いときは多くの場合 IIR フィルタが用いられている。
今回の研究では上記の表にある特性を比較的低い次数、サンプリング周波数で数式演算、
ソフトウェアによる処理、VHDL によるシミュレートで確認・検証しその後に次数・サンプリング周波数の
変化がフィルタの動作にどう影響するかを調べてなぜ高いサンプリング周波数になると FIR フィルタは
あまり使用されないのかを考える。設計するフィルタの形式は低域通過フィルタとする。
理由は、ほかの帯域通過、帯域阻止、高域通過フィルタはすべて低域通過フィルタをもとにして
作られるからである。

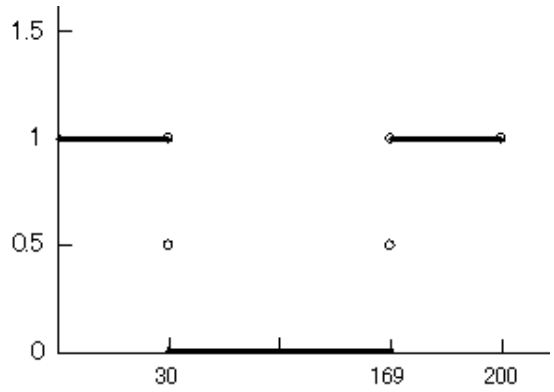
5－1. 28 次直線位相FIRフィルタの設計

まず、FIR フィルタの動作特性から検証する。

標本化周期（サンプリング周期）を96000Hz、フィルタの次数を28（2M）次として理想振幅特性を200等分した標本値を希望特性とした

$$|H_k| = \begin{cases} 1 & (0 \le k \le 30, 169 \le k \le 199) \\ 0.5 & (k = 31, 168) \\ 0 & (32 \le k \le 167) \end{cases}$$

の直線位相 FIR フィルタを作成する。尚その際の数式の演算には C 言語を利用する



まず、最適な乗算係数をもとめる。上図より理想振幅特性が中心の$14Ts$で対称となるので
乗算係数は$\alpha_0 Ts$～$\alpha_{14} Ts$まで求めることとなる。これには IDFT を利用した式

$$\alpha(mTs) = \frac{1}{N}\sum_{k=0}^{N-1} |H_d(kf_0)| e^{j\frac{2\pi}{N}km} \quad (m = 0,1,\cdots,M)$$

を用いる。その値は

$\alpha_0 = 0.315000$
$\alpha_1 = 0.265991$
$\alpha_2 = 0.145945$
$\alpha_3 = 0.018209$
$\alpha_4 = -0.057819$
$\alpha_5 = -0.061585$
$\alpha_6 = -0.017838$
$\alpha_7 = 0.027028$
$\alpha_8 = 0.039190$
$\alpha_9 = 0.017233$
$\alpha_{10} = -0.014155$
$\alpha_{11} = -0.028052$
$\alpha_{12} = -0.016412$
$\alpha_{13} = 0.006952$
$\alpha_{14} = 0.020963$

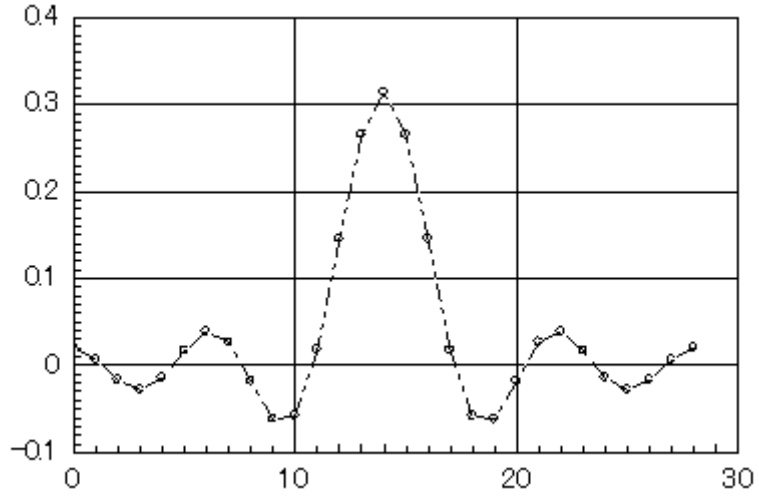
となる。係数値は偶関数で次数が 2M なので 15 以降は a_{14} を中心として 13.12.11……と繰り返していくことになる。求めた乗算係数と

$$h\{(M+m)Ts\} = h\{(M-m)Ts\} = \alpha(mTs) \quad (m = 0,1,\cdots,M)$$

よりインパルス応答 $h\{(M \pm m)Ts\}$ を求める

$h_0 = h_{28} = \alpha_{14}$
$h_1 = h_{27} = \alpha_{13}$
$h_2 = h_{26} = \alpha_{12}$
$h_3 = h_{25} = \alpha_{11}$
$h_4 = h_{24} = \alpha_{10}$
$h_5 = h_{23} = \alpha_9$
$h_6 = h_{22} = \alpha_8$
$h_7 = h_{21} = \alpha_7$
$h_8 = h_{20} = \alpha_6$
$h_9 = h_{19} = \alpha_5$
$h_{10} = h_{18} = \alpha_4$
$h_{11} = h_{17} = \alpha_3$
$h_{12} = h_{16} = \alpha_2$
$h_{13} = h_{15} = \alpha_1$
$h_{14} = \alpha_0$

となる。これを図示すると

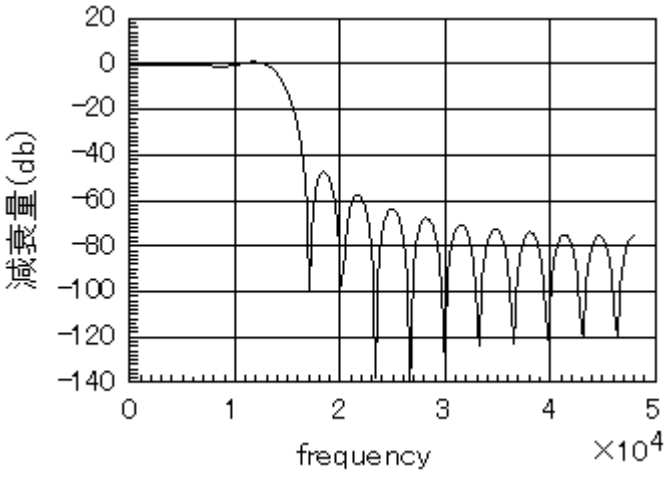


となる。これは 14Ts(MTs)を中心に線対称になっている。また、この関係よりインパルス応答が
乗算係数に等しく、システム応答が有限であることがわかる。
次に、作成したフィルタの振幅特性を求める。これによって設計したフィルタの特性がわかる

振幅特性は

$$\left|\alpha(0)+2\sum_{m=1}^{M}\alpha(mTs)\cos m\omega Ts\right|$$

より算出される。横軸に規格化周波数、縦軸に入出力の振幅比である減衰量（利得）をとったグラフを
表記すると



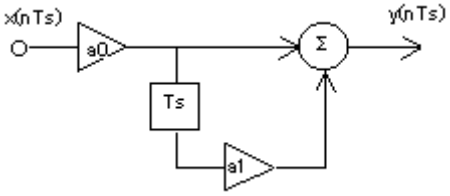
となる。減衰量の推移よりこのフィルタは低域通過フィルタ（ローパスフィルタ）として
機能していることがわかる
この図より、通過帯域の周波数が 14000、阻止帯域が 16000 あたりにあることがわかる
次にこのフィルタに正弦波を入力してフィルタリングした後の出力波形の変化をみる

まず、FIR フィルタの入出力の関係式は

$$y_k = \sum_{m=0}^{M} \alpha_m x_{k-m}$$

であり、現在および過去の入力値 x_{k-m} のみで現在の出力 y_k が計算されるものである

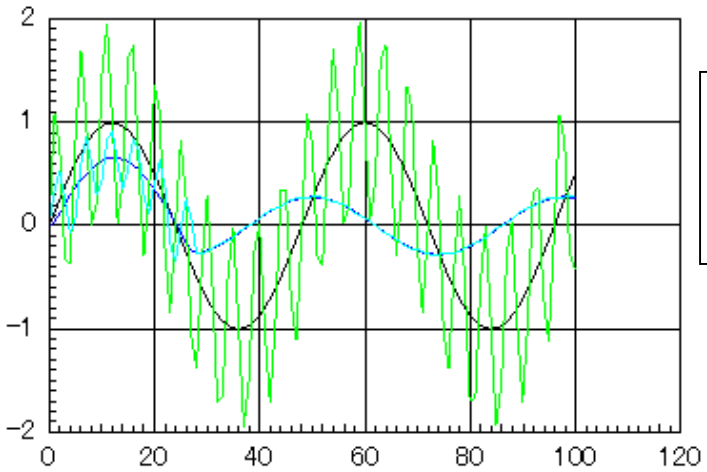
下図は FIR フィルタの最小構成数学的モデルである。



これらより、FIR フィルタにはフィードバック（帰還路）がなくまた誤差が累積することもないので

係数値が実数である場合は安定であることがわかる

実際の波形の入出力を対比した図は以下のようになる



黒：2kHz の正弦波

緑：2kHz と 20kHz の合成正弦波

青：2kHz のフィルタ出力

水色：合成正弦波のフィルタ出力

これは、2KHzと20kHzの合成正弦波 FIR フィルタで処理した場合の入出力波形の推移である

合成正弦波の出力が28Ts を過ぎたところで 2kHz の正弦波の出力と一致しているのがわかる。

このことは、20kHzの信号が取り除かれ2kHzの信号が取り出されていることを示しこのフィルタが

低域通過フィルタとして機能していることがわかる。

5－2. バタワース特性を利用したIIRフィルタの設計

次にFIRフィルタの設計要綱を流用して類似したIIRフィルタを作成する
サンプリング周期を96000、アナログフィルタ設計に使用する周波数を通過域14400、阻止域15360
許容偏差を通過域14、阻止域15[db]とする

$$Ts = 0.1041\times10^{-4}(\text{サンプリング周波数} = 1/Ts = 96000Hz)$$

$$f_p = 14400Hz, \qquad f_r = 15360Hz$$

$$A_p = 14dB, \qquad A_r = 15dB$$

を満たすバタワース特性を有するフィルタを作成する
最初に想定したデジタルフィルタの仕様に対応するアナログフィルタを作成する

$$\omega_A = \frac{2}{T}\tan\left(\frac{\Omega_D T}{2}\right) \text{より} \quad \begin{aligned}\omega_p &= (2/Ts)\tan(2\pi\times14400\times Ts/2) = 97818rad/s\\ \omega_r &= (2/Ts)\tan(2\pi\times15360\times Ts/2) = 105539rad/s\end{aligned}$$

これがアナログフィルタの角周波数となる。これよりアナログフィルタの次数Nを求める

$$N = \frac{\log\left\{\left(10^{A_r/10}-1\right)/\left(10^{A_p/10}-1\right)\right\}}{2\log\left(\omega_r/\omega_p\right)}$$

より次数はN＝2となるので、2次のフィルタを作成することになる
次に遮断周波数ω_0を求めるが、これは$\omega_0[p], \omega_0[r]$をもとめてそこから算出することになる

$$\omega_0[p] = \frac{\omega_p}{\left(10^{A_p/10}-1\right)^{1/2N}}$$

$$\omega_0[r] = \frac{\omega_r}{\left(10^{a_r/10}-1\right)^{1/2N}}$$

$$\omega_0 = \frac{\omega_0[p]+\omega_0[r]}{2}$$

これより

$$\omega_0 = 44502$$

となる。これをバタワースの一般式

$$H_2(s) = \frac{1}{(s/\omega_0)^2 + \cos(\phi_k)(s/\omega_0)+1}$$

$$\phi_k = (2\times k-1)\pi/2\times N$$

$$(k = 1.2.3.\cdots N/2)$$

に代入して値を求める。

$$\frac{(44502)^2}{s^2+\sqrt{2}(44502)s+(44502)^2}=\frac{1.9804\times10^9}{s^2+6.2935\times10^4 s+1.9804\times10^9}=G_2(s)$$

これがデジタルフィルタの基本となるアナログフィルタの一般式となる

s で表現されている領域を z で表現されている領域へと変換すればよいので

$$s=\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$$

を利用して

$$s_{ana\log}=2\times9.6\times10^4\frac{1-z^{-1}}{1+z^{-1}}$$

を $G_2(s)$ に代入して $s-z$ 変換することによりデジタルフィルタの式を求める

$$H_{digital}=\frac{0.012284(1+z^{-1})^2}{1+(-0.45181)z^{-1}+(-0.49905)z^{-2}}=\frac{0.012284+0.024568z^{-1}+0.012284z^{-2}}{1+(-0.45181)z^{-1}+(-0.49905)z^{-2}}$$

これが設計した IIR デジタルフィルタの式となる。

$$H(z)=\frac{a_0+a_1z^{-1}+a_2z^{-2}}{1-b_1z^{-1}-b_2z^{-2}}$$

より、このフィルタの乗算係数値は

$a_0=0.012284$

$a_1=0.024568$

$a_2=0.012284$

$b_1=0.45181$

$b_2=0.49905$

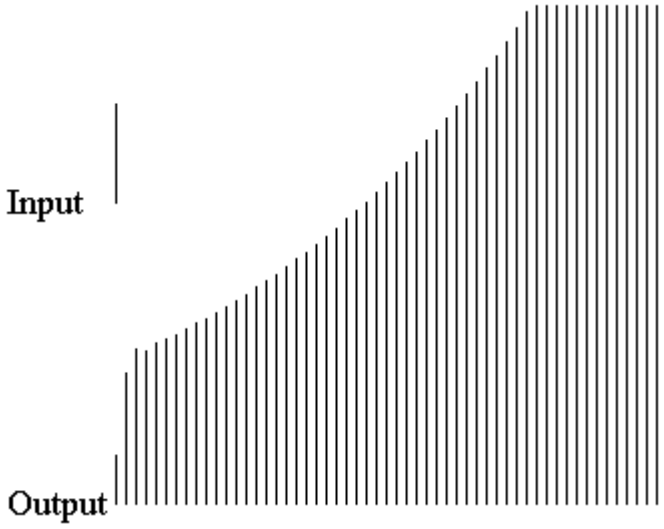
となる。

IIR フィルタの安定する条件（フィルタが成立する条件）は乗算係数 b_n の絶対値を取った総和が 1 未満であることなので、設計したフィルタの安定性を解析するために b_n の絶対値をとった総和が 1 未満になるかを確認すると

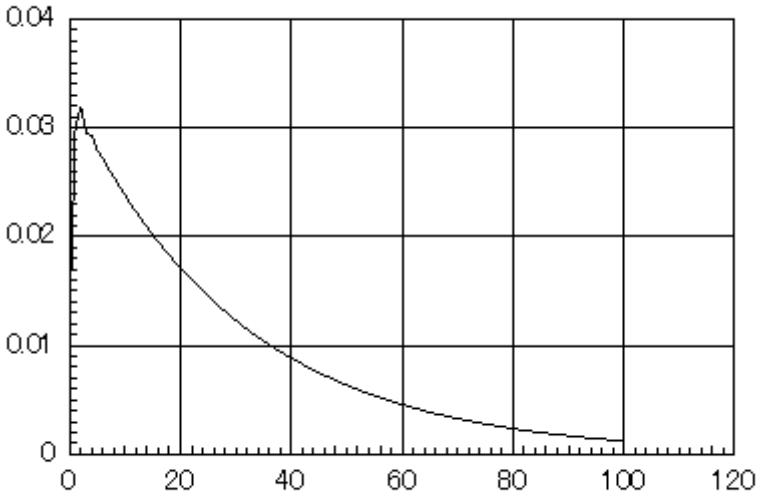
$b_1+b_2=0.45181+0.49905=0.95086\le1$

となるので、条件を満たし安定したシステム応答を持つということになる

もし総和が 1 を超えてしまった場合そのシステム応答は



上記のような信号の推移となり、時間の経過とともに指数関数的に増加したあとに表示範囲を超えてしまう､つまり値が発散してしまう｡これだと安定しないのでフィルタとして成立しない。
今回設計したフィルタは安定性の条件を満たし値が最終的には 0 に収束する方向にある。
その経過をグラフにあらわすと



となる。設計したフィルタの特性を調べるために FIR のときと同様に振幅特性と位相特性を算出する。IIR フィルタの振幅特性と位相特性は

$$A(\omega) = \sqrt{\left(H_R(\omega)\right)^2 + \left(H_I(\omega)\right)^2}$$

$$\phi(\omega) = \tan^{-1}\frac{H_I(\omega)}{H_R(\omega)}$$

という式より求められる。ここではシステム関数$H(\omega)$の実数部を$H_R(\omega)$、虚数部を $H_I(\omega)$
と分割して

$$H(\omega) = H_R(\omega) + H_I(\omega)$$

としている。また、システム関数$H(\omega)$は伝達関数$H(z)$を利用して

$$H(\omega) = H(z)\big|_{z=e^{j\omega Ts}}$$

$$\frac{\sum_{k=0}^{M} a_k z^{-k}}{1-\sum_{l=1}^{M} b_l z^{-l}} = \frac{\sum_{k=0}^{M} a_k e^{-jk\omega Ts}}{1-\sum_{l=1}^{M} b_l e^{-jl\omega Ts}}$$

と求めることができる
これを利用して設計したフィルタの振幅特性、位相特性を算出する。伝達関数は

$$H_{digital} = \frac{0.012284 + 0.024568z^{-1} + 0.012284z^{-2}}{1 + (-0.45181)z^{-1} + (-0.49905)z^{-2}}$$

なのでこれに離散正弦波$z = e^{j\omega Ts}$を入力し

$$H_{digital} = \frac{0.012284 + 0.024568e^{-j\omega Ts} + 0.012284e^{-2j\omega Ts}}{1 + (-0.45181)e^{-j\omega Ts} + (-0.49905)e^{-2j\omega Ts}}$$

オイラーの公式$e^{-j\omega Ts} = \cos\omega Ts - j\sin\omega Ts$を利用して

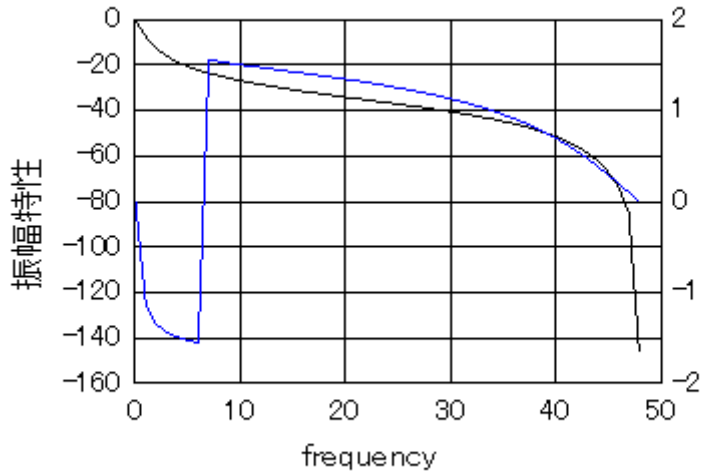
$$\frac{0.012284 + 0.024568(\cos\omega Ts - j\sin\omega Ts) + 0.012284(\cos 2\omega Ts - j\sin 2\omega Ts)}{1 - \{0.45181(\cos\omega Ts - j\sin\omega Ts) + 0.49905(\cos 2\omega Ts - j\sin 2\omega Ts)\}}$$

$$\frac{0.012284 + 0.024568\cos\omega Ts - j0.024568\sin\omega Ts + 0.012284\cos 2\omega Ts - j0.012284\sin 2\omega Ts}{1 - \{0.45181\cos\omega Ts - j0.45181\sin\omega Ts + 0.49905\cos 2\omega Ts - j0.49905\sin 2\omega Ts\}}$$

$$A(\omega) = \frac{\sqrt{(0.012284 + 0.024568\cos\omega Ts + 0.012284\cos 2\omega Ts)^2 + (-0.024568\sin\omega Ts - 0.012284\sin 2\omega Ts)^2}}{\sqrt{(1 - 0.45181\cos\omega Ts - 0.49905\cos 2\omega Ts)^2 + (0.45181\sin\omega Ts + 0.49905\sin 2\omega Ts)^2}}$$

$$\phi(\omega) = \angle - \tan^{-1}\frac{0.024568\sin\omega Ts + 0.012284\sin 2\omega Ts}{0.012284 + 0.024568\cos\omega Ts + 0.012284\cos 2\omega Ts} - \tan^{-1}\frac{0.45181\sin\omega Ts + 0.49905\sin 2\omega Ts}{1 - 0.45181\cos\omega Ts - 0.49905\cos 2\omega Ts}$$

となる。これを計算して数値を取りグラフ化すると



となる。減衰量の推移、振幅特性と位相特性の近似関係によりこのフィルタが想定どうりに IIR 低域通過フィルタとして機能していることがわかる

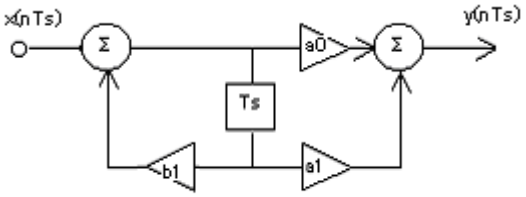
次にこのフィルタの入出力波形の比較を行う。FIR のときと同じく２kHｚの正弦波を入力しその変化を調べる。

IIR の入出力関係式は

$$y_k = \sum_{n=1}^{N} b_n y_{k-n} + \sum_{m=0}^{M} a_m y_{k-m}$$

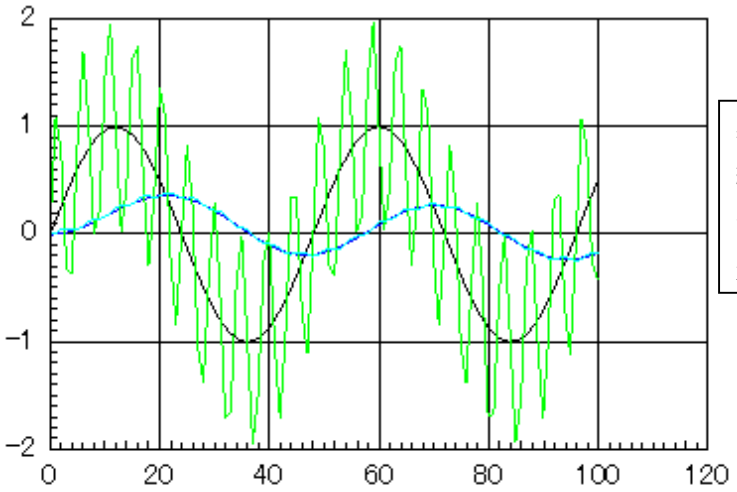
であらわされる。

下の図はこの式を最小構成数学的モデル化したものである。



これは FIR フィルタと比較して過去の出力値 y_{k-n} が現在の出力 y_k の計算に影響を与える点が異なる。

実際の入出力波形の比較は下図のようになる



黒：2kHz の正弦波
緑：2kHz と 20kHz の合成正弦波
青：2kHz のフィルタ出力
水色：合成正弦波のフィルタ出力

FIR フィルタのときと同じく２kＨｚと２０kＨｚの合成正弦波を入力しその出力をみると２kＨｚのフィルタ出力と一致していることがわかる。
つまり、２０ｋＨｚの信号成分が取り除かれており、このことからＩＩＲフィルタも設計したＦＩＲフィルタと同様に低域通過フィルタとして機能していることがわかる。

６．VHDL による構成とシミュレート

ハードウェアによるデジタルフィルタの実装（VHDL シミュレート）によって
FIR と IIR の特性的な差異を調べる

６－１．ハードウェア構成のシンボル表記

デジタルフィルタは、ソフトウェアにしろハードウェアにしろ下記に示す三つの要素
入力信号 $x_1(nTs), x_2(nTs), \cdots, x_N(nTs)$ の和

$$y(nTs) = \sum_{i=1}^{N} x_i(nTs)$$

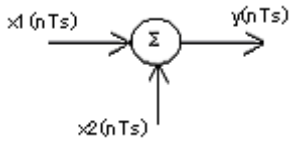
を出力する加算器
入力信号 $x(nTs)$ と乗算係数 a の積

$$y(nTs) = ax(nTs)$$

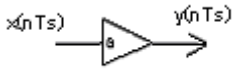
を出力する乗算器
Ts 秒前に入力された信号を出力する

$$y(nTs) = x\{(n-1)Ts\}$$

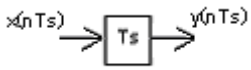
遅延器（メモリ）
によって成り立っている。
そのシンボル表記は
加算器



乗算器



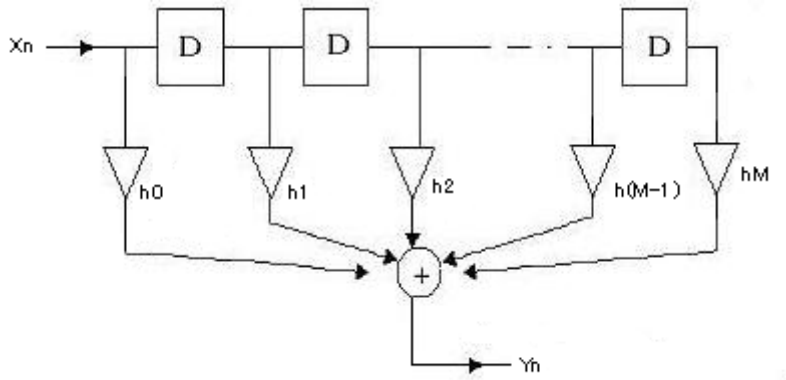
遅延器



とあらわされる。基本的にハードウェアで構成されるときも数式をシンボルによって表記化し
構成を決定する。

６－２．FIR フィルタの構成とシミュレート

FIR フィルタをシンボルによって表記したハードウェア構成モデルは下記のようになる。



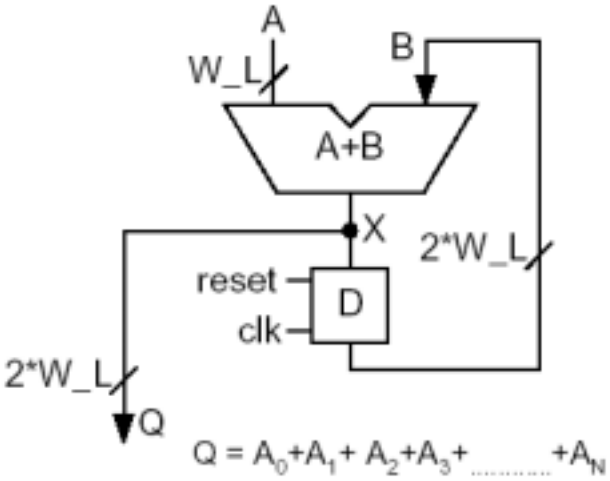
これは、式

$$y_k = \sum_{m=0}^{M} a_m x_{k-m}$$

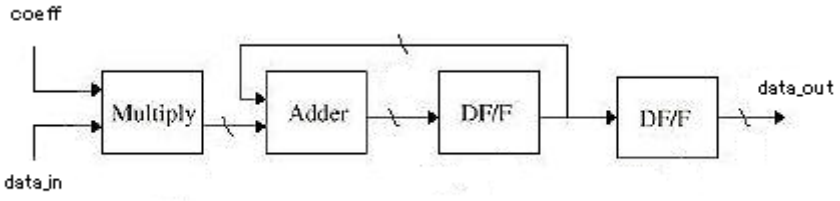
$$H(z) = \sum_{m=0}^{M} a_m z^{-m} = \sum_{m=0}^{M} h_m z^{-m}$$

よりつくられた構成で、特定の条件でサンプリング処理され入力された信号が乗算係数をかけあわされ、その総和をとり出力することによってフィルタリング処理を実現させている
この構成モデルで使われている遅延器（Ｄフリップフロップ）は現在、過去の複数の入力値 x_{k-m} が処理されることを意味している。
上記のような図の形式のフィルタ構成を直接構成といいフィルタの次数の増加（乗算係数の総数の増加）に応じて乗算器、遅延器が増加していく構成方法となっている。
今回の研究ではこの直接構成をつかってハードウェアを構成するのではなく別の構成方法を利用する。そして作成した回路をシミュレートしてその出力の値を数式計算で求めたものと比較し、フィルタリング処理が行われているかどうかを確認する。

その構成とは、アキュームレータを利用した構成法である
アキュームレータとは累積、積算などをといった意味を持つ算術回路でハードウェアモデルは



上記のような形になり、これによって前の状態を保持して入力系に持っていきそれを演算に利用することができるので前までの演算結果の入力系への再利用を行って過去の入力を
実現することでフィルタを作成すると



となる。この方式をとることによって乗算係数の増加に応じて乗算器と遅延器を増やす必要がなくなる。つまりハードウェア規模の増加の問題が解消できる。

実際にＶＨＤＬでこの回路を作成し実際の動作を検証する。
まず、回路全体を必要最小限なパーツ単位に分割してそれぞれを作成する。
その構成は
掛け算器
足し算器
遅延器
となる。
この組み合わせを上位構成で配線、つなぎ合わせて実際の回路を構築する
そうしてできた回路に入力情報を設定し回路を通してフィルタリング動作がなされているかを

確認する。入力データには乗算係数は、前段階で算出した値（ソフトウェアや数式計算）を利用し、入力する信号値には単位信号を利用する。

そのときの２進データ化したときのビット長は係数値を 12bit,単位信号を 16bit 出力信号のビット長は 28bit とする。小数の値を２進数化すると 0.0101110……というように表記するがこのときに負数は２の補数をとり最上位ビットを符号ビットとして扱う。

実際にＶＨＤＬによるシミュレートを行いでてきた値は

$y_0 = 0.3148.....$ $y_1 = 0.2658.....$ $y_2 = 0.1458.....$ $y_3 = 0.01817.....$

$y_4 = -0.05777.....$ $y_5 = -0.06155.....$ $y_6 = -0.01781.....$ $y_7 = 0.02699.....$

$y_8 = 0.03916.....$ $y_9 = 0.017203.....$ $y_{10} = -0.01412.....$ $y_{11} = -0.02803.....$

$y_{12} = -0.01637.....$ $y_{13} = 0.006924.....$ $y_{14} = 0.02092.....$ $y_{15} = 0.006924.....$

$y_{16} = -0.01637.....$ $y_{17} = -0.02803.....$ $y_{18} = -0.01412.....$ $y_{19} = 0.17203.....$

$y_{20} = 0.03916.....$ $y_{21} = 0.02699.....$ $y_{22} = -0.01781.....$ $y_{23} = -0.06155.....$

$y_{24} = -0.05777.....$ $y_{25} = 00.01817.....$ $y_{26} = 0.01458.....$ $y_{27} = 0.2658.....$

$y_{28} = 0.3148.....$

となる。

これは数式計算よりもとめた y_n の値

$y_0 = 0.315000$ $y_1 = 0.265991$ $y_2 = 0.145945$ $y_3 = 0.018209$

$y_4 = -0.057819$ $y_5 = -0.061585$ $y_6 = -0.017838$ $y_7 = 0.027028$

$y_8 = 0.039190$ $y_9 = 0.017233$ $y_{10} = -0.014155$ $y_{11} = -0.028052$

$y_{12} = -0.016412$ $y_{13} = 0.006952$ $y_{14} = 0.020963$ $y_{15} = 0.006952$

$y_{16} = -0.016412$ $y_{17} = -0.028052$ $y_{18} = -0.014155$ $y_{19} = 0.017233$

$y_{20} = 0.039190$ $y_{21} = 0.027028$ $y_{22} = -0.017838$ $y_{23} = -0.061585$

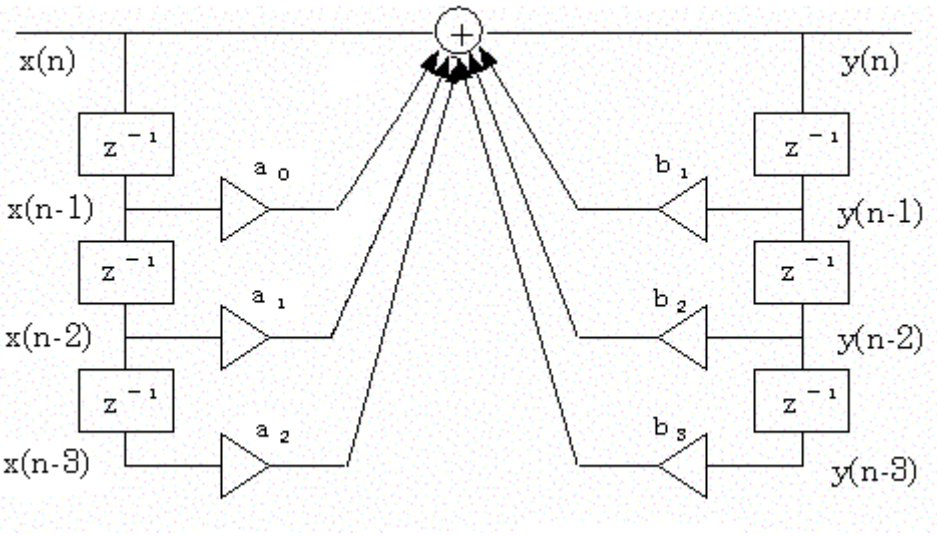
$y_{24} = -0.057819$ $y_{25} = 0.018209$ $y_{26} = 0.145945$ $y_{27} = 0.265991$

$y_{28} = 0.315000$

を２進数表記したものと比較してみると、誤差が生じているがこれは演算誤差で発生したものではない。この誤差は小数を 2 進化したときに発生したものなのでこの誤差のことを考慮してみるとこの回路は FIR フィルタとして動作しているといえる。

６－３．IIR フィルタの構成とシミュレート

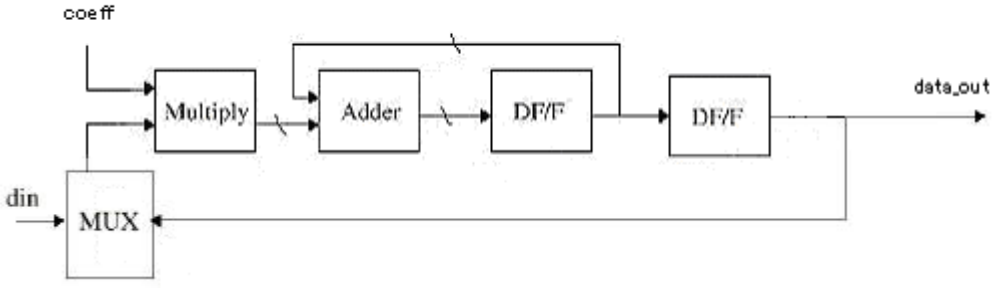
IIR フィルタをシンボル表記すると



のようなハードウェア構成になる。これは、式

$$y_k = \sum_{n=1}^{N} b_n y_{k-n} + \sum_{m=0}^{M} a_m x_{k-m}$$

よりつくられた構成で FIR と比較して過去の出力系が現在の入力系に再帰して影響を与えている点が異なる。この場合も遅延器や乗算係数の増加に伴って各素子（D フリップフロップ等）が増加していくが IIR の場合も FIR のときと同じくアキュームレータを利用して回路の増大を防ぐ。アキュームレータを利用した IIR のハードウェア構成は



となる。過去の出力系をあらわす y_{n-k} と入力 x をマルチプレクサを使い多重化してデータ入力としている。これは $\sum_{n=1}^{N} b_n y_{k-n}$ と $\sum_{m=0}^{M} a_m x_{k-m}$ の演算をひとつの演算回路で行うためである。これによって FIR と比較してほぼ変わらないくらいの規模の回路構成とすることができる。

実際にこの回路を VHDL によって作成しシミュレートを行う。IIR の場合も動作の検証に単位信号を入力系として使用するが IIR フィルタはインパルス応答が無限に続くためきりがいいところでデータの計測を止めることにする。その基準として FIR の出力の計測終了時間を利用する。乗算係数は前に求めたソフトウェアや数式演算でもとめたものを利用する。
入出力データは FIR のときと同じく 2 進数化し、そのデータ長も同じとする。
その出力データは

$y_0 = 0.012262$ $y_1 = 0.030069$ $y_2 = 0.031810$ $y_3 = 0.029202$
$y_4 = 0.028853$ $y_5 = 0.027391$ $y_6 = 0.026729$ $y_7 = 0.025557$
$y_8 = 0.024628$ $y_9 = 0.023700$ $y_{10} = 0.022771$ $y_{11} = 0.021843$
$y_{12} = 0.020914$ $y_{13} = 0.019986$ $y_{14} = 0.019057$ $y_{15} = 0.018349$
$y_{16} = 0.017664$ $y_{17} = 0.016957$ $y_{18} = 0.016272$.....

となる。
これは数式計算で算出された値

$y_0 = 0.012284$ $y_1 = 0.030118$ $y_2 = 0.032022$ $y_3 = 0.029498$
$y_4 = 0.029308$ $y_5 = 0.027963$ $y_6 = 0.027260$ $y_7 = 0.026271$
$y_8 = 0.025474$ $y_9 = 0.024620$ $y_{10} = 0.023836$ $y_{11} = 0.023056$
$y_{12} = 0.022312$ $y_{13} = 0.021587$ $y_{14} = 0.020888$ $y_{15} = 0.020211$
$y_{16} = 0.019556$ $y_{17} = 0.018922$ $y_{18} = 0.018308$.....

とは若干数値に誤差がある。
それは、このハードウェア回路では出力 y_n が 2 進数 28bit 長で出力されるのに対して再帰して入力される過去の出力 y_{n-k} を入力 x のビット長 12 ビットにあわせてそれ以下のビット長を切り捨てて入力し、演算を行っているからである。そのため演算が繰り返されるたびに誤差が蓄積されていきずれが生じる。よってこの回路ではそこの点を考慮して出力系がある程度近似であればフィルタ回路として正当性があるものとしている。
そのずれは最大で約 0.0005 ほどである。ただしこれは出力の値それぞれにおいて生じるものなのでそれを考えて誤差を調べるとこの回路では最大誤差の許容範囲内に収まっている。
上記の出力値の結果とそれが正確な数値計算の値と近似していることにより、この回路は誤差を含んだ IIR フィルタ回路として動作していることがわかる。

７．考察

これまでの数値計算・ソフトウェアによる処理、VHDL によって求められたものをまとめるとまず一般的にいわれている FIR と IIR の特性の違いというのを検証確認することはほぼ終えることができた。残りの問題として、ビット長の切り詰めによる数値の誤差があるがこれは値が許容範囲内で推移していることとする。

７－１．フィルタの処理時間の差

また、回路構成を固定してしまうことでハードウェアの規模的な違いがあまりなくなった（というよりはむしろ FIR よりも IIR のほうが増大している）が、フィルタの同一時間あたりの信号の処理を比較してみると

FIR フィルタの y_{28} 付近の入出力信号の推移

IIR フィルタの y_{65} 付近の入出力信号の推移

となり、これにより FIR フィルタが信号の処理を 28Ts まで行う時間で IIR フィルタは 65Ts まで処理が行われているのがわかる。

このことにより FIR フィルタの処理には多大な時間がかかることがわかる。

仮にハードウェア構成を直接構成で実現していたとしたら次数に比例して遅延器と乗算器が増加しその結果処理時間も比例して増加していたことと考えられる。

すなわち、ハードウェア構成を工夫してリソースの増加を防げたとしても時間的な制限は解消できないということになる。

７－２．フィルタの次数と減衰量の関係

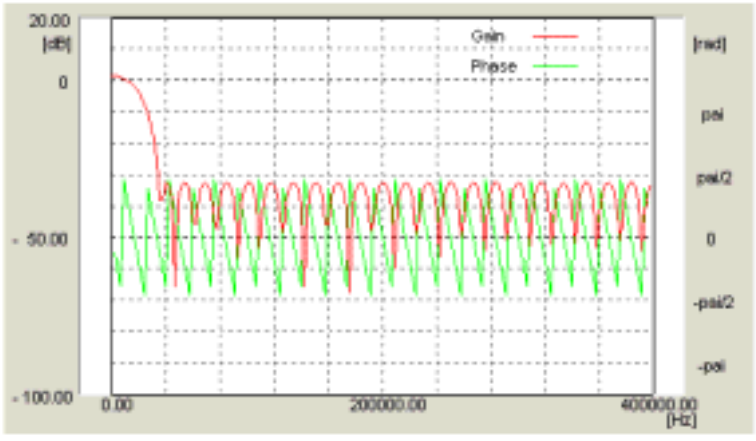
FIR のサンプリング信号の周波数帯を 800kＨｚとか１MHz などのフィルタの動作検証をした 96000kHz より高くとったとしてその阻止域と通過域を低めに取る・・・急峻で比較的通過域が少ないフィルタを作成するとしたらそのフィルタの振幅特性と次数の関係は次のようになる

サンプリングレート８００ｋＨｚ

通貨域の周波数（これ以下の周波数の信号を通す）17ｋHz

阻止域の周波数（これ以上の周波数の信号を通さない）35ｋHz

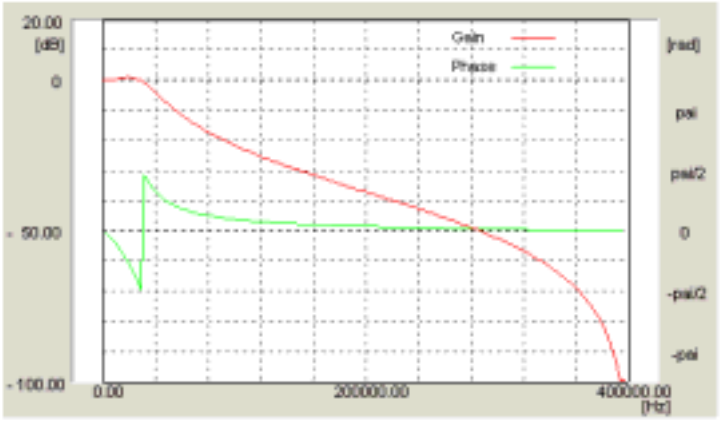
として

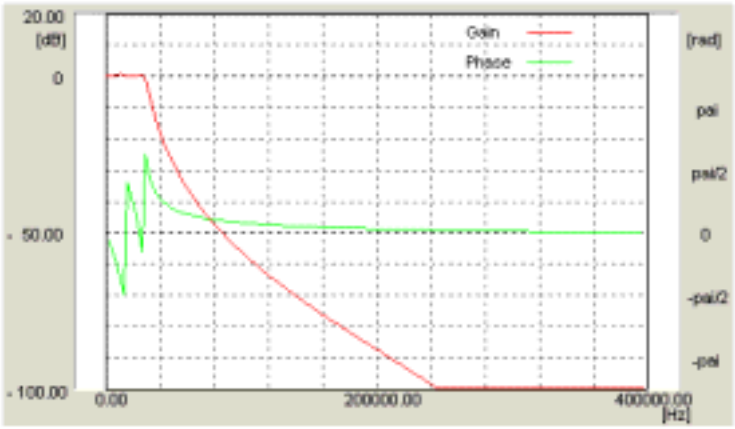


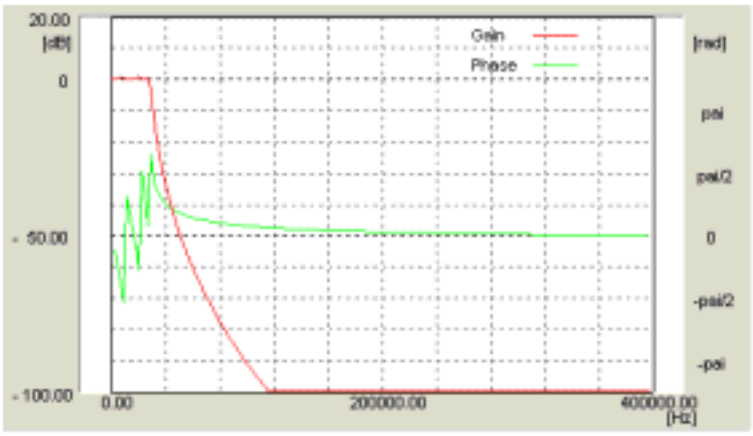
上が次数１００、下が次数５０のときの減衰量と位相の波形である。

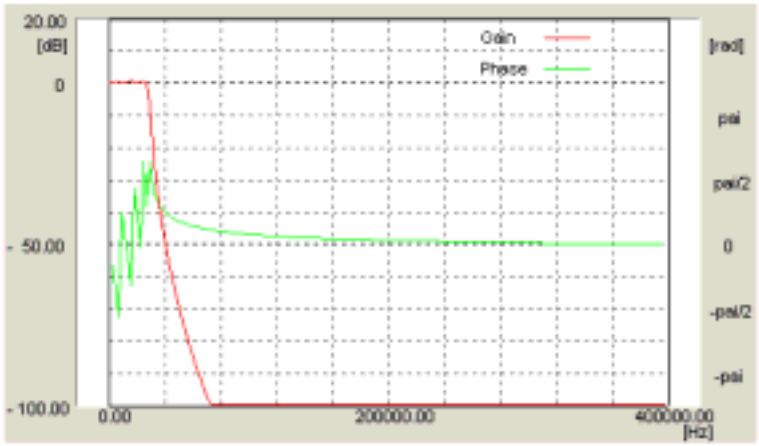
減衰量を多くとるのなら上図の比較より次数が多くなければならないことがわかる。

次に IIR フィルタでも同一の処理を行うとする。このとき IIR フィルタでは阻止域と通過域の周波数を $\omega_r + \omega_p / 2$ によって遮断周波数 ω_0 にまとめているのと、IIR フィルタの次数を求める式 $N = \dfrac{\log\left[\left(10^{A_r/10} - 1\right) / \left(10^{A_p/10} - 1\right)\right]}{2\log\left(\omega_r / \omega_p\right)}$ は、通過域と阻止域の周波数だけでなく通過域と阻止域の信号の揺れ（リップル）も考慮しなければならない。
これを考慮してフィルタの次数ごとの振幅比を見ると









上から順に次数１、２、３、４となっている。
これより IIR では高周波数なサンプリング周波数をもつフィルタを作る場合でも次数の増加は圧倒的に低くなっていることがわかる。

8. まとめ

本研究によって
FIR・IIR デジタルフィルタの各構成とその動作特性について確認することができた。
それより FIR と IIR の次数の振幅特性への影響と処理時間の差の比較によって
FIR フィルタは、IIR フィルタに比べて安定性に優れており、入出力特性も安定しており
小数点の切捨て誤差も累積しない。その反面、１つの処理体系にかかる時間が莫大になり
よい性能を得るのに高い次数を必要とするフィルタリング処理にはむいていない。
IIR フィルタは安定させれば、ＦＩＲフィルタで次数が多くなるような特性を低い次数で
実現でき、低次数で性能のよいフィルタを作るのに適している
しかし、フィルタとして動作するのには安定させるのが容易ではなく、過去の出力系が現在の
入力系のフィルタリング処理にかかわってきて 1 実行あたりの演算が複雑になりそのときの小
数点の切捨てによる誤差も累積するので正確なフィルタを作り難いことがわかった
この結果より安定性と総合的な処理時間を考慮すると CDDA 相当の低い次数でよい特性を実現
できるフィルタリング処理には FIR フィルタのほうが適しているといえるが FIR フィルタで
よい特性を実現するには高い次数を必要とするフィルタリング処理では安定させた IIR フィ
ルタを使用したほうが適しているということが判明し、それぞれのフィルタの適切な用途が理解
できた。

9. 謝辞

今回の研究を行うにあたって様々なアドバイスを頂いた橘昌良助教授に感謝いたします。

10.　参考文献

１．伊達玄訳　「ディジタル信号処理（上・下)」、コロナ社
２．前田渡著　「ディジタル信号処理の基礎」、オーム社
３．Jayaram Bhasker 著　　「ＶＨＤＬ言語入門」、ＣＱ出版社

付録

ここには FIR フィルタと IIR フィルタの動作を検証したときに用いた C 言語のプログラムのソースと VHDL のソース、テストベンチを掲載している。

28 次直線位相 FIR フィルタのプログラムソース

乗算係数

```
#include <stdio.h>
#include <math.h>
#include <conio.h>

#define n 200 /*  希望振幅特性の標本点数  */
#define m 14 /* FIR フィルタの次数を２M としたときの値 M */

float sfr[m+1],sfi[m+1];
float lfr[n],lfi[n];
float pi;
void idft( int, int );
void ilpf( int );
void LPW( void );

void main( void )
{
   LPW();
   getch();
   }

   void idft( int N, int M )
   {
      int i,k;
            float omega;

            for (i=0; i<=M-1; ++i) {
               sfr[i]=0.0; /*  実数部の初期化  */
               sfi[i]=0.0; /*  虚数部の初期化  */
```

```
            for (k=0; k<=N-1; ++k) {
               omega=2.0*pi*(float)i*(float)k/(float)N; /* 回転子の角度計算 */
sfr[i]+=lfr[k]*cos(omega)-lfi[k]*sin(omega); /* IDFT の実数部の計算 */
sfi[i]*=lfi[k]*cos(omega)+lfr[k]*sin(omega); /* IDFT の虚数部の計算 */
                     }
            }
          for (i=0; i<=M-1; ++i) /* 1/N の乗算 */
          {
            sfr[i]/=(float)N;
            sfi[i]/=(float)N;
            }
          }

  void ilpf( int N ) /* 希望振幅特性の分割(LPF) */
  {
    int i;

          for (i=0; i<=30; ++i) {
            lfr[i]=1.0;
          }
    for (i=169; i<=N-1; ++i) {
            lfr[i]=1.0;
          }
          lfr[31]=0.5;
          lfr[168]=0.5;
          for (i=32; i<=167; ++i) {
            lfr[i]=0.0;
          }
          for (i=0; i<=N-1; ++i) { /* 虚数部を零に設定 */
            lfi[i]=0.0;
          }
  }
  void LPW( void )
  {
   int i;
```

```
    pi=acos(-1.0); /* 円周率 */
    ilpf( n ); /* 希望帯域通過特性を N 等分する */
    idft( n, m+1 );/* M+1 個の最適な係数を求める */
    for (i=0; i<=m; ++i) {
    printf("a(%2dTs) = %f = a(%2dTs)¥n",i,sfr[i],2*m-i);
    }
  }
```

振幅特性

```
for (i=0; i<=m; ++i) {
    printf("a(%2dTs) = %f = a(%2dTs)¥n",i,sfr[i],2*m-i);
    }
```

を

```
omega=2.0*pi*(float)i;
            amp=sfr[0];
            for (k=1; k<=m; ++k) {
            amp+=2.0*sfr[k]*cos((float)k*omega*ts);
            }
               amp=sqrt(amp*amp);
               printf("|H(5%d)| = %f ¥n",i,20.0*log(amp));
```

へと置き換える。

FIR フィルタの入出力信号の推移

```
#include <math.h>
#include <stdio.h>
#include <conio.h>


#define M 28

void main( void ){
```

```
int i,k;
double y,x[M+1],a[M+1];
double pi,ts=0.1041e-4;
pi=acos(-1.0);
 for (k=0; k<=M; ++k) {
   x[k]=0.0;
        }
 a[0]=0.315000;
 a[1]=0.265991;
 a[2]=0.145945;
 a[3]=0.018209;
 a[4]=-0.057819;
 a[5]=-0.061585;
 a[6]=-0.017838;
 a[7]=0.027028;
 a[8]=0.039190;
 a[9]=0.017233;
 a[10]=-0.014155;
 a[11]=-0.028052;
 a[12]=-0.016412;
 a[13]=0.006952;
 a[14]=0.020963;
 a[15]=0.006952;
 a[16]=-0.016412;
 a[17]=-0.028052;
 a[18]=-0.014155;
 a[19]=0.017233;
 a[20]=0.039190;
 a[21]=0.027028;
 a[22]=-0.017838;
 a[23]=-0.061585;
 a[24]=-0.057819;
 a[25]=0.018209;
 a[26]=0.145945;
 a[27]=0.265991;
 a[28]=0.315000;
```

```
    for (i=0; i<=100; ++i) {
       x[0]=1.0*sin(2.0*pi*2000*(double)i*ts);
       x[0]+=1.0*sin(2.0*pi*20000*(double)i*ts);
             y=0.0;
             for (k=0; k<=M; ++k) {
             y+=a[k]*x[k];
             }
             for (k=M; k>=1; --k){
             x[k]=x[k-1];
             }
       printf("i = %d x = %f y = %f¥n",i,x[0],y);

getch();
}
}
```

FIR フィルタのインパルス応答をもとめるプログラム

```
#include <math.h>
#include <stdio.h>
#include <conio.h>


#define M 28

void main( void ){

   int i,k;
   double y,x[M+1],a[M+1];
   double pi,ts=0.1041e-4,freq=2000.0;
   pi=acos(-1.0);
    for (k=0; k<=M; ++k) {
      x[k]=0.0;
            }
    a[0]=0.315000;
    a[1]=0.265991;
    a[2]=0.145945;
    a[3]=0.018209;
    a[4]=-0.057819;
    a[5]=-0.061585;
    a[6]=-0.017838;
    a[7]=0.027028;
    a[8]=0.039190;
    a[9]=0.017233;
    a[10]=-0.014155;
    a[11]=-0.028052;
    a[12]=-0.016412;
    a[13]=0.006952;
    a[14]=0.020963;
    a[15]=0.006952;
    a[16]=-0.016412;
    a[17]=-0.028052;
```

```
    a[18]=-0.014155;
    a[19]=0.017233;
    a[20]=0.039190;
    a[21]=0.027028;
    a[22]=-0.017838;
    a[23]=-0.061585;
    a[24]=-0.057819;
    a[25]=0.018209;
    a[26]=0.145945;
    a[27]=0.265991;
    a[28]=0.315000;



    for (i=0; i<=100; ++i) {
       if( i==0){
         x[0]=1.0;
       }
       else {
         x[0]=0.0;
       }
            y=0.0;
            for (k=0; k<=M; ++k) {
            y+=a[k]*x[k];
            }
            for (k=M; k>=1; --k){
            x[k]=x[k-1];
            }
       printf("i = %d x = %f y = %f¥n",i,x[0],y);

getch();
}
}
```

バタワース特性の IIR フィルタのプログラムソース

アナログフィルタの角周波数

```
#include <stdio.h>
#include <math.h>
#include <conio.h>


void main( void ) {
double pi=3.141592;
double outp,outr,ts,frequencyp,frequencyr;

printf(" Ts [ms]= ");
scanf("%lf",&ts);
printf(" p[Hz] = ");
scanf("%lf",&frequencyp);
printf(" r[Hz] = ");
scanf("%lf",&frequencyr);

outp=(2/ts)*tan(2*pi*frequencyp*ts/2);
outr=(2/ts)*tan(2*pi*frequencyr*ts/2);
printf("omega[p] = %f, omega[r] = %f ¥n",outp,outr);

getch();
}
```

アナログフィルタの次数

```
#include <stdio.h>
#include <math.h>
#include <conio.h>

void main( void ){
double n,ar,ap,wp,wr,x,y;

printf(" a[r] = ");
scanf("%lf",&ar);
printf(" a[p] = ");
scanf("%lf",&ap);
printf(" w[r] = ");
scanf("%lf",&wr);
printf(" w[p] = ");
scanf("%lf",&wp);
x=pow(10,ar/10);
y=pow(10,ap/10);
n=log((x-1)/(y-1))/(2*log(wr/wp));
printf(" N = %f ¥n",n);
getch();
}
```

デジタルフィルタの通過帯と阻止帯の角周波数

```
#include <stdio.h>
#include <math.h>
#include <conio.h>

void main( void ){
double ap,y,n,omega,wp;
printf("wp=");
scanf("%lf",&wp);
printf("ap=");
scanf("%lf",&ap);
printf("n=");
scanf("%lf",&n);
y=wp;
y/=pow((pow(10,ap/10)-1),1/(2*n));
omega=y;
printf("omega= %f ¥n",omega);
getch();
}
```

IIR フィルタのインパルス応答

```
#include <math.h>
#include <stdio.h>
#include <conio.h>

#define M 2

void main( void ){

   int i,k;
   double x,y,v[M+1],a[M+1],b[M+1];
   double pi,freq,ts=0.1041e-4;
   pi=acos(-1.0);
```

```
    for (k=0; k<=M; ++k) {
    v[k]=0.0;
          }
   a[0]=0.012284;
   a[1]=0.024568;
   a[2]=0.012284;
   b[1]=0.45181;
   b[2]=0.49905;
   printf(" freq = ");
   scanf("%lf",&freq);
   for (i=0; i<=100; ++i) {
     if (i==0 ) {
        x=1.0;
      }
       else {
          x=0.0;
      }
           v[0]=x;
           for (k=1; k<=M; ++k) {
              v[0]+=b[k]*v[k];

           }
           y=0.0;
           for (k=0; k<=M; ++k){
              y+=a[k]*v[k];
           }
           for (k=M; k>=1; --k){
           v[k]=v[k-1];
           }
           printf("i = %d x = %f y =%f ¥n",i,x,y);
getch();
  }
}
```

IIR フィルタの振幅特性と位相特性

```
#include <stdio.h>
#include <math.h>
#include <conio.h>

void main( void ){
   int i;
   double omega,denomi;
   double denomi_real,denomi_imagi;
   double nume;
   double amp;
   double pi,freq,ts,phase;

   pi=acos(-1.0);
   ts=0.1041e-4;

   for (i=0; i<=48; ++i) {
      freq=(double)i*1000.0;
      omega=2.0*pi*freq;
            denomi_real=0.50095*cos(omega*ts)-0.45181;
            denomi_imagi=1.49905*sin(omega*ts);
            denomi=pow(denomi_real,2.0);
            denomi+=pow(denomi_imagi,2.0);
            nume=2+2*cos(omega*ts);
            amp=0.012284*(nume/sqrt(denomi));
            phase=-atan(denomi_imagi/denomi_real);
      printf("|H(%5d)| = %f PHASE= %f ¥n",i,20*log10(amp),phase);
            getch();
            }
```

IIR フィルタの入出力の推移

```
#include <math.h>
#include <stdio.h>
#include <conio.h>
```

```
#define M 2

void main( void ){
int i,k;
   double x,y,v[M+1],
a[M+1],b[M+1];
   double pi,ts=0.1041e-4;
   pi=acos(-1.0);
      for (k=0; k<=M; ++k) {
      v[k]=0.0;
            }
     a[0]=0.012284;
     a[1]=0.024568;
     a[2]=0.012284;
     b[1]=0.45181;
     b[2]=0.49905;
     for (i=0; i<=100; ++i) {
        x=1.0*sin(2.0*pi*2000*(double)i*ts);
            x+=1.0*sin(2.0*pi*22000*(double)i*ts);
              v[0]=x;
              for (k=1; k<=M; ++k) {
                 v[0]+=b[k]*v[k];
                  }
              y=0.0;
              for (k=0; k<=M; ++k){
                 y+=a[k]*v[k];
              }
              for (k=M; k>=1; --k){
              v[k]=v[k-
1];
              }
              printf("i = %d x = %f y =%f ¥n",i,x,y);
getch();
   }
}
```

FIR フィルタの VHDL ソース

```
library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;

entity multiplier_16x12 is
    Port (
            RST : In std_logic; -- reset
            ACCCLK : In std_logic; -- 192Fs
            A_IN : In std_logic_vector( 15 downto 0 );
            B_IN : In std_logic_vector( 11 downto 0 );
            MUL_OUT : Out std_logic_vector( 27 downto 0 ) );
end multiplier_16x12;

architecture STRUCT of multiplier_16x12 is

  signal coeff_in : signed( 15 downto 0 );
  signal data_in : signed( 11 downto 0 );
  signal mul_tmp : signed( 27 downto 0 );

begin
gen_coeff_in : process( RST, ACCCLK )
begin

if(RST = '1') then
  coeff_in <= "0000000000000000";
elsif( ACCCLK'event and ACCCLK = '1' ) then
  for i in 15 downto 0 loop
    coeff_in(i) <= A_IN(i);
  end loop;
end if;

end process gen_coeff_in;
```

```
gen_data_in : process( RST, ACCCLK )
begin

if(RST = '1') then
   data_in <= "000000000000";
elsif( ACCCLK'event and ACCCLK = '1' ) then
   for i in 11 downto 0 loop
      data_in(i) <= B_IN(i);
   end loop;
end if;

end process gen_data_in;

   mul_tmp <= coeff_in * data_in;

gen_MUL_OUT : process( RST, ACCCLK )
begin -- process gen_MULTIPLIED_OUT

if(RST = '1') then
   MUL_OUT <= "000000000000000000000000000000";
elsif( ACCCLK'event and ACCCLK = '1' ) then
   for i in 27 downto 0 loop
      MUL_OUT(i) <= mul_tmp(i);
   end loop;
end if;

end process gen_MUL_OUT;
end;

library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_signed.all;

entity accumulator is
    Port (
            RST : In std_logic; -- reset
```

```
            ACCCLK : In std_logic; -- 192Fs
            C_IN : In std_logic_vector(27 downto 0);
            ACC_OUT : Out std_logic_vector(27 downto 0) );
end accumulator;

architecture STRUCTURE of accumulator is


  signal B_tmp,C_tmp : std_logic_vector( 27 downto 0 );

begin -- architecture accumulator
process(C_IN,B_tmp)
variable A_tmp:std_logic_vector(27 downto 0);
begin
A_tmp(27 downto 0):=C_IN;
C_tmp<=A_tmp + B_tmp;
end process;
process(ACCCLK)
begin
if(ACCCLK='1')and(ACCCLK'EVENT)then
if(RST='1')then
B_tmp<=(others=>'0');
else
B_tmp<=C_tmp;
end if;
end if;
end process;

process(C_tmp)
begin
ACC_OUT<=C_tmp;
end process;
end;


library IEEE;
```

```
use IEEE.std_logic_1164.all;

entity DFF is
port( D : in std_logic_vector(27 downto 0);
        D_OUT : out std_logic_vector(27 downto 0);
        RST : in std_logic;
        OEN:in std_logic;
        ACCCLK : in std_logic);
    end DFF;

architecture struct of DFF is
signal Q:std_logic_vector(27 downto 0);
begin
process(ACCCLK)
begin
if(ACCCLK='1')and(ACCCLK'EVENT)then
if(RST='1')then
Q<=(others=>'0');
else
Q<=D;
end if;
end if;
end process;
process(Q,OEN)
begin
case OEN is
when '0'=>
D_OUT<=(others=>'Z');
when others =>
D_OUT<=Q(26 downto 0)&'0';
end case;
end process;

end struct;
```

```
library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_arith.all;

entity fir_top is
    Port (
              RST : In std_logic;
              OEN : In std_logic;
              A_IN: In std_logic_vector(15 downto 0);
               B_IN : In std_logic_vector(11 downto 0);
              ACCCLK : In std_logic; -- 384Fs (master clock)
              D_OUT : Out std_logic_vector(27 downto 0) );
end fir_top;

architecture STRUCTURE of fir_top is

   signal mul_signal : std_logic_vector(27 downto 0);
   signal acc_signal : std_logic_vector(27 downto 0);

   component multiplier_16x12
Port (
              RST : In std_logic;
              ACCCLK : In std_logic;
              A_IN : In std_logic_vector( 15 downto 0 );
              B_IN : In std_logic_vector( 11 downto 0 );
              MUL_OUT : Out std_logic_vector( 27 downto 0 ) );
end component;

component accumulator
Port (
              RST : In std_logic;
              ACCCLK : In std_logic;
              C_IN : In std_logic_vector(27 downto 0);
              ACC_OUT : Out std_logic_vector(27 downto 0) );
end component;
```

```
component DFF
Port (
        D : in std_logic_vector(27 downto 0);
        D_OUT : out std_logic_vector(27 downto 0);
        RST : in std_logic;
        OEN : in std_logic;
        ACCCLK : in std_logic);
end component;

begin -- architecture fir_top

multiplier_16x12_1: multiplier_16x12 port map (
        RST => RST,
        ACCCLK => ACCCLK,
        A_IN => A_IN,
        B_IN => B_IN,
        MUL_OUT => mul_signal);

accumulator_1: accumulator port map (
        RST => RST,
        ACCCLK => ACCCLK,
        C_IN => mul_signal,
        ACC_OUT => acc_signal);

DFF_1: DFF port map (
      D => acc_signal,
      D_OUT => D_OUT,
      RST => RST,
      OEN => OEN,
      ACCCLK => ACCCLK);

end; -- architecture fir_top
```

FIR フィルタの VHDL テストベンチ

尚、テストベンチは処理終了までのソースファイルが莫大になり且つ途中より反復演算処理となるのできりのいいところで省略してある。

```
LIBRARY   IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

LIBRARY ieee;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY testbench IS
END testbench;

ARCHITECTURE testbench_arch OF testbench IS
-- If you get a compiler error on the following line,
-- from the menu do Options->Configuration select VHDL 93
FILE RESULTS: TEXT IS OUT "results.txt";
        COMPONENT fir_top
                PORT (
                        RST : In    std_logic;
                        OEN : In    std_logic;
                        A_IN : In    std_logic_vector (15 DOWNTO 0);
                        B_IN : In    std_logic_vector (11 DOWNTO 0);
                        ACCCLK : In    std_logic;
                        D_OUT : Out    std_logic_vector (27 DOWNTO 0)
                );
        END COMPONENT;

        SIGNAL RST : std_logic;
        SIGNAL OEN : std_logic;
        SIGNAL A_IN : std_logic_vector (15 DOWNTO 0);
        SIGNAL B_IN : std_logic_vector (11 DOWNTO 0);
        SIGNAL ACCCLK : std_logic;
        SIGNAL D_OUT : std_logic_vector (27 DOWNTO 0);

BEGIN
        UUT : fir_top
        PORT MAP (
```

```
                RST => RST,
                OEN => OEN,
                A_IN => A_IN,
                B_IN => B_IN,
                ACCCLK => ACCCLK,
                D_OUT => D_OUT
        );

        PROCESS
                VARIABLE TX_OUT : LINE;
                VARIABLE TX_ERROR : INTEGER := 0;

                PROCEDURE CHECK_D_OUT(
                        next_D_OUT : std_logic_vector (27 DOWNTO 0);
                        TX_TIME : INTEGER
                ) IS
                        VARIABLE TX_STR : String(1 to 512);
                        VARIABLE TX_LOC : LINE;
                BEGIN
                        -- If compiler error ("/=" is ambiguous) occurs in the next line of
code
                        -- change compiler settings to use explicit declarations only
                        IF (D_OUT /= next_D_OUT) THEN
                                write(TX_LOC,string'("Error at time="));
                                write(TX_LOC, TX_TIME);
                                write(TX_LOC,string'("ns D_OUT="));
                                write(TX_LOC, D_OUT);
                                write(TX_LOC, string'(", Expected = "));
                                write(TX_LOC, next_D_OUT);
                                write(TX_LOC, string'(" "));
                                TX_STR(TX_LOC.all'range) := TX_LOC.all;
                                writeline(results, TX_LOC);
                                Deallocate(TX_LOC);
                                ASSERT (FALSE) REPORT TX_STR SEVERITY
ERROR;
                                TX_ERROR := TX_ERROR + 1;
```

```
        END IF;
END;

BEGIN

ACCCLK <= transport '0';
RST <= transport '1';
OEN <= transport '0';
A_IN <= transport std_logic_vector'("0000000000000000"); --0
B_IN <= transport std_logic_vector'("000000000000"); --0
WAIT FOR 10 ns; -- Time=10 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=20 ns
WAIT FOR 90 ns; -- Time=110 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=200 ns
RST <= transport '0';
A_IN <= transport std_logic_vector'("0010100001010001"); --2851
B_IN <= transport std_logic_vector'("011111111111"); --7FF
WAIT FOR 10 ns; -- Time=210 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=220 ns
WAIT FOR 90 ns; -- Time=310 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=400 ns
WAIT FOR 10 ns; -- Time=410 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=420 ns
WAIT FOR 90 ns; -- Time=510 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=600 ns
OEN <= transport '1';
WAIT FOR 10 ns; -- Time=610 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=620 ns
WAIT FOR 90 ns; -- Time=710 ns
```

```
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=800 ns
RST <= transport '1';
OEN <= transport '0';
WAIT FOR 10 ns; -- Time=810 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=820 ns
WAIT FOR 90 ns; -- Time=910 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=1000 ns
WAIT FOR 10 ns; -- Time=1010 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1020 ns
WAIT FOR 90 ns; -- Time=1110 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=1200 ns
WAIT FOR 10 ns; -- Time=1210 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1220 ns
WAIT FOR 90 ns; -- Time=1310 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=1400 ns
RST <= transport '0';
A_IN <= transport std_logic_vector'("0010100001010001"); --2851
B_IN <= transport std_logic_vector'("000000000000"); --0
WAIT FOR 10 ns; -- Time=1410 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1420 ns
WAIT FOR 90 ns; -- Time=1510 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=1600 ns
A_IN <= transport std_logic_vector'("0010001000001011"); --220B
B_IN <= transport std_logic_vector'("011111111111"); --7FF
WAIT FOR 10 ns; -- Time=1610 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1620 ns
```

```
WAIT FOR 90 ns; -- Time=1710 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=1800 ns
WAIT FOR 10 ns; -- Time=1810 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1820 ns
WAIT FOR 90 ns; -- Time=1910 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=2000 ns
OEN <= transport '1';
WAIT FOR 10 ns; -- Time=2010 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=2020 ns
WAIT FOR 90 ns; -- Time=2110 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=2200 ns
RST <= transport '1';
OEN <= transport '0';
WAIT FOR 10 ns; -- Time=2210 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=2220 ns
WAIT FOR 90 ns; -- Time=2310 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=2400 ns
WAIT FOR 10 ns; -- Time=2410 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=2420 ns
IF (TX_ERROR = 0) THEN
        write(TX_OUT,string'("No errors or warnings"));
        writeline(results, TX_OUT);
        ASSERT (FALSE) REPORT
                "Simulation successful (not a failure).  No problems
detected. "
                SEVERITY FAILURE;
ELSE
        write(TX_OUT, TX_ERROR);
```

```
                write(TX_OUT, string'(
                        " errors found in simulation"));
                writeline(results, TX_OUT);
                ASSERT (FALSE) REPORT
                        "Errors found during simulation"
                        SEVERITY FAILURE;
            END IF;
        END PROCESS;
END testbench_arch;

CONFIGURATION fir_top_cfg OF testbench IS
        FOR testbench_arch
        END FOR;
END fir_top_cfg;
```

IIR フィルタの VHDL ソース

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;
entity mux is
port (
        mux_a: in std_logic_vector(11 downto 0);
        mux_b: in std_logic_vector(27 downto 0);
        mux_out: out std_logic_vector(11 downto 0);
        s: in std_logic);
end mux;

architecture struct of mux is
signal mux_a_tmp: std_logic_vector(27 downto 0);
signal mux_b_tmp: std_logic_vector(27 downto 0);
signal mux_tmp_out: std_logic_vector(27 downto 0);

begin

gen_mux_a_tmp : process(mux_a)
begin
for i in 11 downto 0 loop
        mux_a_tmp(i+16) <= mux_a(i);
     end loop;
end process gen_mux_a_tmp;

gen_mux_b_tmp: process(mux_b)
begin
mux_b_tmp<=mux_b;
end process gen_mux_b_tmp;


process(mux_a_tmp,mux_b_tmp,s)
begin
if(s='1')then
```

```
mux_tmp_out<=mux_a_tmp;
else
mux_tmp_out<=mux_b_tmp;

end if;
end process;
gen_mux_out:process(mux_tmp_out)
begin
for i in 11 downto 0 loop
        mux_out(i) <= mux_tmp_out(i+16);
     end loop;
            end process;
end struct;


library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;

entity multiplier_16x12 is
    Port (
            RST : In std_logic; -- reset
            ACCCLK : In std_logic; -- 192Fs
            A_IN : In std_logic_vector( 15 downto 0 );
            B_IN : In std_logic_vector( 11 downto 0 );
            MUL_OUT : Out std_logic_vector( 27 downto 0 ) );
end multiplier_16x12;

architecture STRUCT of multiplier_16x12 is

  signal coeff_in : signed( 15 downto 0 );
  signal data_in : signed( 11 downto 0 );
  signal mul_tmp : signed( 27 downto 0 );

begin -- architecture multiplier_16x12
```

```
gen_coeff_in : process( RST, ACCCLK )
begin -- process gen_coeff_tmp

if(RST = '1') then
   coeff_in <= "0000000000000000";
elsif( ACCCLK'event and ACCCLK = '1' ) then
   for i in 15 downto 0 loop
      coeff_in(i) <= A_IN(i);
   end loop;
end if;

end process gen_coeff_in;

gen_data_in : process( RST, ACCCLK )
begin -- process gen_data_tmp

if(RST = '1') then
   data_in <= "000000000000";
elsif( ACCCLK'event and ACCCLK = '1' ) then
   for i in 11 downto 0 loop
      data_in(i) <= B_IN(i);
   end loop;
end if;

end process gen_data_in;

   mul_tmp <= coeff_in * data_in;

gen_MUL_OUT : process( RST, ACCCLK )
begin -- process gen_MULTIPLIED_OUT

if(RST = '1') then
   MUL_OUT <= "000000000000000000000000000000";
elsif( ACCCLK'event and ACCCLK = '1' ) then
   for i in 27 downto 0 loop
      MUL_OUT(i) <= mul_tmp(i);
```

```
   end loop;
end if;

end process gen_MUL_OUT;
end;


library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_signed.all;

entity accumulator is
    Port (
            RST : In std_logic; -- reset
            ACCCLK : In std_logic; -- 192Fs
            C_IN : In std_logic_vector(27 downto 0);
            ACC_OUT : Out std_logic_vector(27 downto 0) );
end accumulator;

architecture STRUCTURE of accumulator is


   signal B_tmp,C_tmp : std_logic_vector( 27 downto 0 );

begin
process(C_IN,B_tmp)
variable A_tmp:std_logic_vector(27 downto 0);
begin
A_tmp(27 downto 0):=C_IN;
C_tmp<=A_tmp + B_tmp;
end process;
process(ACCCLK)
begin
if(ACCCLK='1')and(ACCCLK'EVENT)then
if(RST='1')then
B_tmp<=(others=>'0');
```

```
else
B_tmp<=C_tmp;
end if;
end if;
end process;

process(C_tmp)
begin
ACC_OUT<=C_tmp;
end process;
end;


library IEEE;
use IEEE.std_logic_1164.all;

entity DFF is
port( D : in std_logic_vector(27 downto 0);
        D_OUT : out std_logic_vector(27 downto 0);
        RST : in std_logic;
        OEN:in std_logic;
        ACCCLK : in std_logic);
    end DFF;

architecture struct of DFF is
signal Q:std_logic_vector(27 downto 0);
begin
process(ACCCLK)
begin
if(ACCCLK='1')and(ACCCLK'EVENT)then
if(RST='1')then
Q<=(others=>'0');
else
Q<=D;
end if;
end if;
```

```
end process;
process(Q,OEN)
begin
case OEN is
when '0'=>
D_OUT<=(others=>'Z');
when others =>
D_OUT<=Q(26 downto 0)&'0';
end case;
end process;

end struct;


library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;
entity mux2 is
port (
        mux_in_a:in std_logic_vector(27 downto 0);
        s2:in std_logic;
        mux2_out:out std_logic_vector(27 downto 0));
end mux2;

architecture arch of mux2 is
signal mux_tmp_a: std_logic_vector(27 downto 0);
signal mux_tmp_b: std_logic_vector(27 downto 0);
signal mux_out_tmp: std_logic_vector(27 downto 0);
begin
gen_mux_tmp_a:process(mux_in_a)
begin
mux_tmp_a<=mux_in_a;
end process gen_mux_tmp_a;

mux_tmp_b<=mux_out_tmp;
```

```
process (mux_tmp_a,mux_tmp_b,s2)
begin
if(s2='1')then
mux_out_tmp<=mux_tmp_a;
else
mux_out_tmp<=mux_tmp_b;
end if;
mux2_out<=mux_out_tmp;
end process;

end arch;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;
entity mux3 is
port (
        mux_in_c:in std_logic_vector(27 downto 0);
        s3:in std_logic;
        mux3_out:out std_logic_vector(27 downto 0));
end mux3;

architecture arch of mux3 is
signal mux_tmp_c: std_logic_vector(27 downto 0);
signal mux_tmp_d: std_logic_vector(27 downto 0);
signal mux3_out_tmp: std_logic_vector(27 downto 0);
begin
gen_mux_tmp_c:process(mux_in_c)
begin
mux_tmp_c<=mux_in_c;
end process gen_mux_tmp_c;

mux_tmp_d<=mux3_out_tmp;
```

```
process (mux_tmp_c,mux_tmp_d,s3)
begin
if(s3='1')then
mux3_out_tmp<=mux_tmp_c;
else
mux3_out_tmp<=mux_tmp_d;
end if;
mux3_out<=mux3_out_tmp;
end process;

end arch;


library IEEE;
   use IEEE.std_logic_1164.all;
   use IEEE.std_logic_arith.all;

entity iir_top is
    Port (
              s: in std_logic;
              s2: in std_logic;
              s3: in std_logic;
              RST : In std_logic;
              OEN : In std_logic;
              A_IN: In std_logic_vector(15 downto 0);
               G_IN : In std_logic_vector(11 downto 0);
              ACCCLK : In std_logic; -- 384Fs (master clock)
              DATA_OUT : Out std_logic_vector(27 downto 0) );
end iir_top;

architecture STRUCTURE of iir_top is

   signal mul_signal : std_logic_vector(27 downto 0);
   signal acc_signal : std_logic_vector(27 downto 0);
   signal mux1_signal : std_logic_vector(11 downto 0);
   signal mux2_signal : std_logic_vector(27 downto 0);
```

```
   signal mux3_signal : std_logic_vector(27 downto 0);
   signal D_signal : std_logic_vector(27 downto 0);
component mux
port (
          mux_a: in std_logic_vector(11 downto 0);
         mux_b: in std_logic_vector(27 downto 0);
         mux_out: out std_logic_vector(11 downto 0);
         s: in std_logic);
end component;

component multiplier_16x12
Port (
              RST : In std_logic;
              ACCCLK : In std_logic;
              A_IN : In std_logic_vector( 15 downto 0 );
              B_IN : In std_logic_vector( 11 downto 0 );
              MUL_OUT : Out std_logic_vector( 27 downto 0 ) );
end component;

component accumulator
Port (
              RST : In std_logic;
              ACCCLK : In std_logic;
              C_IN : In std_logic_vector(27 downto 0);
              ACC_OUT : Out std_logic_vector(27 downto 0) );
end component;

component DFF
Port (
              D : in std_logic_vector(27 downto 0);
              D_OUT : out std_logic_vector(27 downto 0);
              RST : in std_logic;
              OEN : in std_logic;
              ACCCLK : in std_logic);
end component;
```

```
component mux2
port (
        mux_in_a:in std_logic_vector(27 downto 0);
        s2:in std_logic;
        mux2_out:out std_logic_vector(27 downto 0));
end component;

component mux3
port (
        mux_in_c:in std_logic_vector(27 downto 0);
        s3:in std_logic;
        mux3_out:out std_logic_vector(27 downto 0));
end component;



begin -- architecture iir_top
mux_1 : mux port map (
           mux_a => G_IN,
           mux_b =>mux3_signal,
                        s => s,
           mux_out =>mux1_signal);


multiplier_16x12_1: multiplier_16x12 port map (
              RST => RST,
              ACCCLK => ACCCLK,
              A_IN => A_IN,
              B_IN => mux1_signal,
              MUL_OUT => mul_signal);

accumulator_1: accumulator port map (
              RST => RST,
              ACCCLK => ACCCLK,
              C_IN => mul_signal,
              ACC_OUT => acc_signal);
```

```
DFF_1: DFF port map (
        D => acc_signal,
        D_OUT => D_signal,
        RST => RST,
        OEN => OEN,
        ACCCLK => ACCCLK);

mux_2: mux2 port map(
        mux_in_a=>D_signal,
        s2=>s2,
        mux2_out=>mux2_signal);
mux_3 : mux3 port map(
         mux_in_c=>mux2_signal,
         s3=>s3,
         mux3_out=>mux3_signal);


gen_DATA_OUT: process(D_signal)
begin
DATA_OUT<=D_signal;
end process;
end; -- architecture iir_top
```

IIR フィルタの VHDL テストベンチ

```
LIBRARY  IEEE;
USE IEEE.std_logic_1164.all;
USE IEEE.std_logic_arith.all;

LIBRARY ieee;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE STD.TEXTIO.ALL;

ENTITY testbench IS
```

```
END testbench;

ARCHITECTURE testbench_arch OF testbench IS
-- If you get a compiler error on the following line,
-- from the menu do Options->Configuration select VHDL 93
FILE RESULTS: TEXT IS OUT "results.txt";
        COMPONENT iir_top
                PORT (
                        s : in    std_logic;
                        s2 : in    std_logic;
                        s3 : in    std_logic;
                        RST : In    std_logic;
                        OEN : In    std_logic;
                        A_IN : In    std_logic_vector (15 DOWNTO 0);
                        G_IN : In    std_logic_vector (11 DOWNTO 0);
                        ACCCLK : In    std_logic;
                        DATA_OUT : Out    std_logic_vector (27 DOWNTO 0)
                );
        END COMPONENT;

        SIGNAL s : std_logic;
        SIGNAL s2 : std_logic;
        SIGNAL s3 : std_logic;
        SIGNAL RST : std_logic;
        SIGNAL OEN : std_logic;
        SIGNAL A_IN : std_logic_vector (15 DOWNTO 0);
        SIGNAL G_IN : std_logic_vector (11 DOWNTO 0);
        SIGNAL ACCCLK : std_logic;
        SIGNAL DATA_OUT : std_logic_vector (27 DOWNTO 0);

BEGIN
        UUT : iir_top
        PORT MAP (
                s => s,
                s2 => s2,
                s3 => s3,
```

```
                RST => RST,
                OEN => OEN,
                A_IN => A_IN,
                G_IN => G_IN,
                ACCCLK => ACCCLK,
                DATA_OUT => DATA_OUT
        );

        PROCESS
                VARIABLE TX_OUT : LINE;
                VARIABLE TX_ERROR : INTEGER := 0;

                PROCEDURE CHECK_DATA_OUT(
                        next_DATA_OUT : std_logic_vector (27 DOWNTO 0);
                        TX_TIME : INTEGER
                ) IS
                        VARIABLE TX_STR : String(1 to 512);
                        VARIABLE TX_LOC : LINE;
                BEGIN
                        -- If compiler error ("/=" is ambiguous) occurs in the next line of
code
                        -- change compiler settings to use explicit declarations only
                        IF (DATA_OUT /= next_DATA_OUT) THEN
                                write(TX_LOC,string'("Error at time="));
                                write(TX_LOC, TX_TIME);
                                write(TX_LOC,string'("ns DATA_OUT="));
                                write(TX_LOC, DATA_OUT);
                                write(TX_LOC, string'(", Expected = "));
                                write(TX_LOC, next_DATA_OUT);
                                write(TX_LOC, string'(" "));
                                TX_STR(TX_LOC.all'range) := TX_LOC.all;
                                writeline(results, TX_LOC);
                                Deallocate(TX_LOC);
                                ASSERT (FALSE) REPORT TX_STR SEVERITY
ERROR;
                                TX_ERROR := TX_ERROR + 1;
```

```
        END IF;
END;

BEGIN

ACCCLK <= transport '0';
s <= transport '1';
s2 <= transport '0';
s3 <= transport '0';
RST <= transport '1';
OEN <= transport '0';
A_IN <= transport std_logic_vector'("0000000000000000"); --0
G_IN <= transport std_logic_vector'("000000000000"); --0
WAIT FOR 10 ns; -- Time=10 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=20 ns
WAIT FOR 90 ns; -- Time=110 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=200 ns
RST <= transport '0';
A_IN <= transport std_logic_vector'("0000000110010010"); --192
G_IN <= transport std_logic_vector'("011111111111"); --7FF
WAIT FOR 10 ns; -- Time=210 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=220 ns
WAIT FOR 90 ns; -- Time=310 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=400 ns
RST <= transport '0';
WAIT FOR 10 ns; -- Time=410 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=420 ns
WAIT FOR 90 ns; -- Time=510 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=600 ns
s2 <= transport '1';
```

```
RST <= transport '0';
OEN <= transport '1';
WAIT FOR 10 ns; -- Time=610 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=620 ns
WAIT FOR 90 ns; -- Time=710 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=800 ns
s2 <= transport '0';
s3 <= transport '1';
RST <= transport '1';
OEN <= transport '0';
WAIT FOR 10 ns; -- Time=810 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=820 ns
WAIT FOR 90 ns; -- Time=910 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=1000 ns
s <= transport '0';
s3 <= transport '0';
RST <= transport '0';
A_IN <= transport std_logic_vector'("0011100111010100"); --39D4
WAIT FOR 10 ns; -- Time=1010 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1020 ns
WAIT FOR 90 ns; -- Time=1110 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=1200 ns
s <= transport '1';
A_IN <= transport std_logic_vector'("0000001100100101"); --325
G_IN <= transport std_logic_vector'("011111111111"); --7FF
WAIT FOR 10 ns; -- Time=1210 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1220 ns
WAIT FOR 90 ns; -- Time=1310 ns
ACCCLK <= transport '0';
```

```
WAIT FOR 90 ns; -- Time=1400 ns
A_IN <= transport std_logic_vector'("0000000110010010"); --192
G_IN <= transport std_logic_vector'("000000000000"); --0
WAIT FOR 10 ns; -- Time=1410 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1420 ns
WAIT FOR 90 ns; -- Time=1510 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=1600 ns
WAIT FOR 10 ns; -- Time=1610 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1620 ns
WAIT FOR 90 ns; -- Time=1710 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=1800 ns
s2 <= transport '1';
OEN <= transport '1';
WAIT FOR 10 ns; -- Time=1810 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=1820 ns
WAIT FOR 90 ns; -- Time=1910 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=2000 ns
s2 <= transport '0';
RST <= transport '1';
OEN <= transport '0';
WAIT FOR 10 ns; -- Time=2010 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=2020 ns
WAIT FOR 90 ns; -- Time=2110 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=2200 ns
s <= transport '0';
RST <= transport '0';
A_IN <= transport std_logic_vector'("0011111111100000"); --3FE0
WAIT FOR 10 ns; -- Time=2210 ns
```

```
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=2220 ns
WAIT FOR 90 ns; -- Time=2310 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=2400 ns
s <= transport '1';
s3 <= transport '1';
A_IN <= transport std_logic_vector'("0000000000000000"); --0
WAIT FOR 10 ns; -- Time=2410 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=2420 ns
WAIT FOR 90 ns; -- Time=2510 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=2600 ns
s <= transport '0';
s3 <= transport '0';
A_IN <= transport std_logic_vector'("0011100111010100"); --39D4
WAIT FOR 10 ns; -- Time=2610 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=2620 ns
WAIT FOR 90 ns; -- Time=2710 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=2800 ns
s <= transport '1';
A_IN <= transport std_logic_vector'("0000000110010010"); --192
G_IN <= transport std_logic_vector'("011111111111"); --7FF
WAIT FOR 10 ns; -- Time=2810 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=2820 ns
WAIT FOR 90 ns; -- Time=2910 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=3000 ns
A_IN <= transport std_logic_vector'("0000001100100101"); --325
G_IN <= transport std_logic_vector'("000000000000"); --0
WAIT FOR 10 ns; -- Time=3010 ns
ACCCLK <= transport '1';
```

```
WAIT FOR 10 ns; -- Time=3020 ns
WAIT FOR 90 ns; -- Time=3110 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=3200 ns
A_IN <= transport std_logic_vector'("0000000110010010"); --192
WAIT FOR 10 ns; -- Time=3210 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=3220 ns
WAIT FOR 90 ns; -- Time=3310 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=3400 ns
WAIT FOR 10 ns; -- Time=3410 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=3420 ns
WAIT FOR 90 ns; -- Time=3510 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=3600 ns
s2 <= transport '1';
OEN <= transport '1';
WAIT FOR 10 ns; -- Time=3610 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=3620 ns
WAIT FOR 90 ns; -- Time=3710 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=3800 ns
s2 <= transport '0';
RST <= transport '1';
OEN <= transport '0';
WAIT FOR 10 ns; -- Time=3810 ns
ACCCLK <= transport '1';
WAIT FOR 10 ns; -- Time=3820 ns
WAIT FOR 90 ns; -- Time=3910 ns
ACCCLK <= transport '0';
WAIT FOR 90 ns; -- Time=4000 ns
s <= transport '0';
RST <= transport '0';
```

```
                A_IN <= transport std_logic_vector'("0011111111100000"); --3FE0
                WAIT FOR 10 ns; -- Time=4010 ns
                ACCCLK <= transport '1';
                WAIT FOR 10 ns; -- Time=4020 ns

                IF (TX_ERROR = 0) THEN
                        write(TX_OUT,string'("No errors or warnings"));
                        writeline(results, TX_OUT);
                        ASSERT (FALSE) REPORT
                                "Simulation successful (not a failure).  No problems
detected. "
                                SEVERITY FAILURE;
                ELSE
                        write(TX_OUT, TX_ERROR);
                        write(TX_OUT, string'(
                                " errors found in simulation"));
                        writeline(results, TX_OUT);
                        ASSERT (FALSE) REPORT
                                "Errors found during simulation"
                                SEVERITY FAILURE;
                END IF;
        END PROCESS;
END testbench_arch;

CONFIGURATION iir_top_cfg OF testbench IS
        FOR testbench_arch
        END FOR;
END iir_top_cfg;
```