ANALOG DEVICES

TNJ-053
アナログ電子回路技術ノート

LTspice でやってみるデジタル・フィルタ解析（後編）

ΣΔADC での sinc フィルタってホントに sinc?

著者: 石井 聡

はじめに

前回の技術ノートではLTspiceを使って平均化フィルタについて考えてみました。平均化フィルタは sinc フィルタになることがご理解いただけたかと思いますし、LTspice での怪しい（?）テクニックもご高覧いただけたものかと思います。

今回はここまでの理解と怪しいLTspiceテクニックをツールとして、実際のΣΔADC で用いられている sinc フィルタについて解析していきたいと思います。

ΣΔADC で用いられているフィルタ形状は、回路構成としては「平均化フィルタ」には見えないものなのですが、これが sinc フィルタになる理由や、サンプリング定理から考えてちょっと不思議に思われる話題などを掘り下げてみたいと思います。

ΣΔ変調器の ADC を取り上げる

アナログ・デバイセズでは、「実際のΣΔADC」として AD 変換した結果を出力するADC以外に、iCoupler®技術（デジタル・アイソレータ技術）を用いて直流をアイソレーションしたうえで、AD変換結果をΣΔ変調器のビット・ストリーム出力として、そのまま吐き出す「ΣΔ変調器」という派生製品があります。一例とすると、

…

AD7402：シグマ・デルタ変調器、16 ビット、絶縁型

http://www.analog.com/jp/ad7402

【概要】

AD7402 は、アナログ入力信号を高速の 1 ビット・データストリームに変換する高性能、2 次のΣΔ変調器で、アナログ・デバイセズの iCoupler®技術を用いたデジタル絶縁機能を内蔵しています。AD7402 は、4.5 V～5.5 V（VDD1）電源で動作し、±250 mV（フルスケール±320 mV）の差動入力信号に対応します。差動入力はガルバニック絶縁が要求される高電圧アプリケーションでのシャント電圧モニタリングに適しています。

アナログ入力は高精度アナログ変調器によって連続的にサンプリングされ、データレート 10 MHz の 0 と 1 のコード密度としてデジタル出力ストリームに変換されます。元の情報は、適切なデジタル・フィルタによって再構築され、39 kSPS で 87 dB の S/N 比（SNR）を達成することができます。

（後略）

…

が挙げられます。データシートに記載のあるブロック図を図 1 に示します。初段部分にΣ-ΔADC とは書かれていますが、ここは単なるΣΔ変調器であり、AD7402 の出力はビット・ストリームが出ています。なおトランスで分離されているイメージがデジタル・アイソレータであることを示しています。

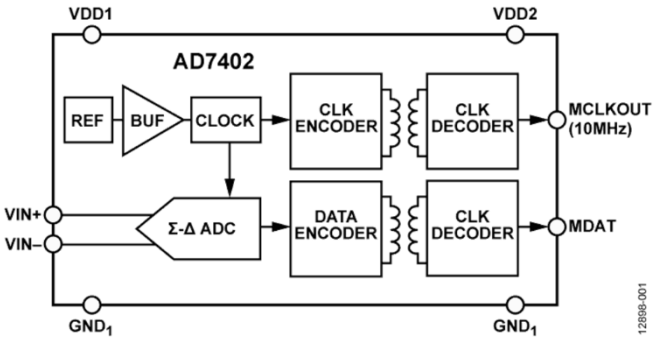

図 1. AD7402 のブロック図

この AD7402 の出力は、その後段に配置する FPGA などを用いたデジタル・フィルタで、本来のデジタル値（16 ビットとか 24 ビット）でフィルタリング操作がおこなわれ、そのフィルタ出力として実際の AD 変換結果が得られます。

「単なるΣΔ変調器のビット・ストリーム…。ん？ビット？ストリーム？」という点は、別に難しいことを考える必要はなく、ΣΔ変調器から出てくる情報が 1 ビット情報であり、これを「ビット・ストリーム」と呼ぶだけの話しです。アナログ的に考えれば、このビット・ストリームをアナログ LPF でフィルタしたときに得られる直流電圧が AD 変換結果です。つまり「ビット・ストリームの電圧平均値が AD 変換結果だ」という理解で十分です。

これをデジタル的に行うものがデジタル・フィルタという話しなだけなのです。

AD7402 のデータシートに記述のある sinc3 フィルタ

さて AD7402 のデータシートには、上記に説明したデジタル・フィルタの回路例（RTL; Register Transfer Level 言語）が掲載されています。この回路（RTL）は「sinc3」つまり sinc フィルタが 3 段従属接続されたフィルタになってるのですが、「ホントに sinc フィルタなのかいな？」と思わせる構造のものなのです。

それこそ私もアナログ・デバイセズに入社したころは、もともとRFエンジニアだったこともあり、「ホントかいな？」と思ったものでした（笑）。またこの回路には、以降に示すような疑問もあり、どうもうまく自己解決できなかったのでした。

それでは、その回路構成について考えてみましょう。ちなみに私は VHDL 言語人なので、Verilog は今ひとつよく判らずなのでした…（汗）。

Rev. 0

アナログ・デバイセズ株式会社

本　　社／〒105-6891 東京都港区海岸 1-16-1 ニューピア竹芝サウスタワービル
電話 03（5402）8200
大阪営業所／〒532-0003 大阪府大阪市淀川区宮原 3-5-36 新大阪トラストタワー
電話 06（6350）6868

```
/*Accumulator (Integrator) Perform the accumula-
tion (IIR) at the speed of the modulator. */

always @ (negedge mclk1, posedge reset)
begin
/* reset は省略 */
       begin
              /* perform accumulation process */
              acc1 <= acc1 + ip_data1;
              acc2 <= acc2 + acc1;
              acc3 <= acc3 + acc2;
       end
end
```

図 2. アキュムレータ・ブロック（RTL）

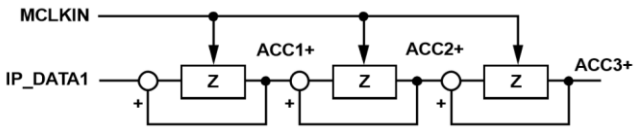

図 3. アキュムレータ・ブロック（Z変換）

```
always @ (mdata1)
       if(mdata1==0)
              ip_data1 <= 37'd0; /* change 0 to
a -1 for twos complement */
       else
              ip_data1 <= 37'd1;
```

図 4. ビット・ストリームからビット拡張する（RTL）

初段のアキュムレータ・ブロック

図 2 は sinc3 デジタル・フィルタ初段の部分の RTL です。Z 変換のブロックで表記したもの（データシート抜粋）も図 3 に示します。mclk1（図中では太字表記）がシステム・クロック（ΣΔ変調器の動作クロック）で、各段ごとにフィードバックされ、それぞれ各 37 ビット（この幅に RTL 内で指定されています）のレジスタに累積されていきます。「acc1 <= acc1 + ip_data1」という記述からも分かります。

図中のコメントある「Accumulator」は累積という意味です。累積は「積」という漢字が用いられていますが、掛け算するわけではなく、足し算が積み重なるという意味です。累積という操作は「積分」操作でもあり、図 2 の RTL でもコメントに「Integrator」と記載が見えます。

またそれぞれの経路が各 37 ビットとなっていますが、その源となる図 2 の ip_data1（図中では太字表記）のさらにその源は、図 4 のようにΣΔ変調器からの入力信号である 1 ビットのビット・ストリーム（mdata1）を 37 ビット（ip_data1）にビット拡張したものです。

後段のディファレンシエータ・ブロック

図 2 の「acc3」、図 3 の ACC3+出力は、後段のディファレンシエータ・ブロックの入力になります。ディファレンシエータ・ブロックの RTL を図 5 に示します。図中の「Differentiator」とは微分とか差異という意味です。この回路の入力にはアキュムレータ・ブロックの出力（acc3）がつながっていることも確認できます。これを Z 変換のブロックで表記したものを図 6 に示します。

この回路（図 5）をよくみてみると、図 2 では mclk1 がシステム・クロックであったものが、この図 5 では word_clk（図中では太字表記）がシステム・クロックになっています（図 6 では WORD_CLK）。

このディファレンシエータ・ブロックは、mclk1 が分周されたクロック速度、もっとデジタル信号処理的にいうと「デシメート（間引き）された」クロックで動作しているのです…。つまり word_clk のレートで 1 遅延（z^{-1}）とすると、デシメートされた分（これを「デシメーション・レート」DRとします）、DR分で遅延（z^{-DR}）されていることになります。

後段のディファレンシエータ・ブロックのクロック生成

この word_clk がどう生成されているかを、あらためて RTL で見てみると、図 7、図 8 の構成になっています。

図 7 はカウンタを構成するブロックで、word_count という 16 ビットのカウンタが mclk1 のレート（ΣΔ変調器の動作クロック）でカウント・アップし、dec_rate（デシメーション・レートDR）の設定値と同じカウント数で word_count がカウントを繰り返すことになります。

つづいて図 8 の word_clk を見てみましょう。この RTL から、word_count の 1 周に同期して word_clk が生成されています。

話しを戻すと、これが図 5、図 6 のディファレンシエータ・ブロックのクロックになります。

```
/*Differentiator (including decimation stage)
Perform the differentiation stage (FIR) at a
lower speed. */
always @ (negedge word_clk, posedge reset)
begin
/* reset は省略 */
       begin
              diff1 <= acc3 - acc3_d2;
              diff2 <= diff1 - diff1_d;
              diff3 <= diff2 - diff2_d;
              acc3_d2 <= acc3;
              diff1_d <= diff1;
              diff2_d <= diff2;
       end
end
```

図 5. ディファレンシエータ・ブロック（RTL）

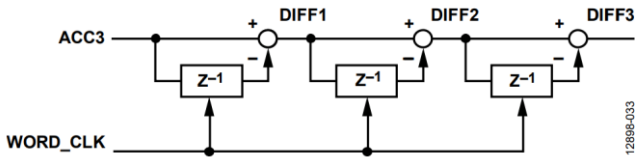

図 6. ディファレンシエータ・ブロック（Z変換）

```
/* decimation stage (MCLKOUT/WORD_CLK) */
always @ (negedge mclk1, posedge reset)
begin
/* reset は省略 */
       begin
              if (word_count == dec_rate - 1)
                     word_count <= 16'd0;
              else
              word_count <= word_count + 16'b1;
       end
end
```

図 7. デシメート・カウンタを構成するブロック（RTL）

```
always @ (negedge mclk1, posedge reset)
begin
/* reset は省略 */
   begin
       if (word_count == dec_rate/2 - 1)
              word_clk <= 1'b1;
       else if (word_count == dec_rate - 1)
              word_clk <= 1'b0;
   end
end
```

図 8. ディファレンシエータのクロックを生成するブロック（RTL）

デシメートという用語を辞書で引いてみると

この RTL を見て個人的には、「word_clk を作ることなく、ディファレンシエータ・ブロックも mclk1 をシステム・クロックとして、一方で word_count でストローブ・パルスを作って、それで完全同期回路として動かしたいなあ」とか余計な想いを巡らせてしまいました…。

まあ、それは良しとして、「しかしこの『デシメート』という言葉は一体どういう意味だったのか？」と今更ながらふと思い、辞書を引いてみました…。辞書は TNJ-036 でも紹介した、アナログ・デバイセズ入社時に購入した、Oxford Advanced Learner's Dictionary 7th Edition（オックスフォード現代英英辞典［第7版］[1]）です。

そこに現れた言葉は…、

> 1 [usually passive] to kill large numbers of animals, plants or people in a particular area.
>
> 2 [informal] to severely damage something or make something weaker.

これ（上の 1 の部分）を訳してみると「ある特定の地域に存在する動物、植物や人間を多数殺す」なんて恐ろしい意味が掲載されています…。デシメートというと、日本語的には「間引く」ということで、多数生えた植物の一部を間引くとか、多数生（な）った果実の実を摘果するとかいうふうにとらえますが、結構恐ろしい意味なのですね…（汗）。

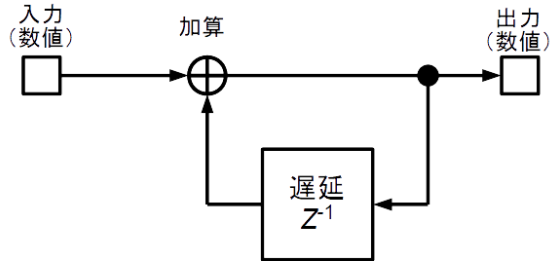

図 9. 一般的な無限インパルス応答フィルタの表現でのアキュムレータ・ブロック$H_A(z)$

この回路の構成

気を取り直して…。もとの話題に戻りましょう（汗）。

おのおの同じ回路が 3 段直列に接続されている

図 3、図 6 から分かることは、おのおの同じ回路が 3 段直列に接続されているということです。全体をひとつの Z 変換の式で書いてみると、

$$H(z) = H_A(z)H_A(z)H_A(z) \cdot H_D(z)\, H_D(z)H_D(z) \qquad (1)$$

と表すことができます。ここで$H_A(z)$はアキュムレータ・ブロックのひとつの回路、$H_D(z)$はディファレンシエータ・ブロックのひとつの回路です。

アキュムレータ・ブロックのひとつの回路だけを考える

ここでアキュムレータ・ブロックのひとつの回路$H_A(z)$だけを考えてみます。アキュムレータ・ブロックに相当するデジタル・フィルタの構造は、図 9 に記載する無限インパルス応答（Infinite Impulse Response; IIR）フィルタになります。入力・出力それぞれのn番目のサンプルを$x(n), y(n)$とすると

$$y(n) = x(n) + y(n-1) \qquad (2)$$

これを Z 変換で表すと、

$$Y(z) = X(z) + z^{-1}Y(z) \qquad (3)$$

$y(n-1)$が 1 サンプル過去になりますから、$z^{-1}Y(z)$となります。ここから伝達関数$H_A(z)$を求めると、

$$Y(z) - z^{-1}Y(z) = (1 - z^{-1})Y(z) = X(z) \qquad (4)$$

$$H_A(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - z^{-1}} \qquad (5)$$

となります。

データシートの図のアキュムレータ・ブロックは標準の IIR フィルタとは異なっている

ところで気が付かれた方もいらっしゃるかと思いますが、この図 9 や式(2)～式(5)と、図 2／図 3 の表記は同じではありません。この図 2／図 3 の表記は、式(2)的には

$$y(n-1) = x(n) + y(n-1) \qquad (6)$$

というような感じで表されるものです。式の右辺は同じですが、左辺で得られるサンプルが 1 サンプル古いものです。

とはいえこの遅延は、結局$H_A(z)$ のひとつの回路内で吸収される事象なので、デジタル・フィルタの実動作としては問題ありません。

ディファレンシエータ・ブロックのひとつの回路だけを考える

つづいてディファレンシエータ・ブロックのひとつの回路$X_D(z)$だけを考えると

$$y(n) = x(n) - x(n - DR) \tag{7}$$

となります。DRはデシメーション・レート（間引き率）です。これをZ変換で伝達関数として表すと

$$H_D(z) = 1 - z^{-DR} \tag{8}$$

となります。

$H_A(z)$と$H_D(z)$をひとつずつ従属接続したものはsinc フィルタになっている

これらの検討により、アキュムレータ・ブロックのひとつの回路$H_A(z)$とディファレンシエータ・ブロックのひとつの回路$H_D(z)$を従属接続したものは

$$H_{AD}(z) = H_A(z) \cdot H_D(z) = \frac{1 - z^{-DR}}{1 - z^{-1}} \tag{9}$$

として伝達関数で表されることが分かります。

この式が単純な多項式になるように、分子を分母で割ってみます。計算の見切りが良くなるように、$DR = 4$として定数にしてみます。さらに次数を降順にしておくと、

$$H_{AD}(z) = \frac{-z^{-4} + 1}{-z^{-1} + 1} \tag{10}$$

さて、それではやってみましょう。

1)

$$\begin{array}{r} z^{-3} \\ -z^{-1}+1 \;\;) -z^{-4}+1 \\ -z^{-4}+z^{-3} \\ \hline -z^{-3}+1 \end{array}$$

2)

$$\begin{array}{r} z^{-3}+z^{-2} \\ -z^{-1}+1 \;\;) -z^{-4}+1 \\ -z^{-4}+z^{-3} \\ \hline -z^{-3}+1 \\ -z^{-3}+z^{-2} \\ \hline -z^{-2}+1 \end{array}$$

3)

$$\begin{array}{r} z^{-3}+z^{-2}+z^{-1} \\ -z^{-1}+1 \;\;) -z^{-4}+1 \\ -z^{-4}+z^{-3} \\ \hline -z^{-3}+1 \\ -z^{-3}+z^{-2} \\ \hline -z^{-2}+1 \\ -z^{-2}+z^{-1} \\ \hline -z^{-1}+1 \end{array}$$

4)

$$\begin{array}{r} z^{-3}+z^{-2}+z^{-1}+1 \\ -z^{-1}+1 \;\;) -z^{-4}+1 \\ -z^{-4}+z^{-3} \\ \hline -z^{-3}+1 \\ -z^{-3}+z^{-2} \\ \hline -z^{-2}+1 \\ -z^{-2}+z^{-1} \\ \hline -z^{-1}+1 \\ -z^{-1}+1 \\ \hline 0 \end{array}$$

ちゃんと割り切れるのですね…。つまり

$$\frac{1 - z^{-4}}{1 - z^{-1}} = 1 + z^{-1} + z^{-2} + z^{-3} \tag{11}$$

となり、な、なんと…、ひとつ前の TNJ-052 の図 5 の平均化フィルタ（sinc フィルタ）そのものになっていることが分かります。これを図で描くと図 10 のようになります。ということで、アキュムレータ・ブロックのひとつの回路とディファレンシエータ・ブロックのひとつの回路を従属接続したものは、sinc フィルタになっていることが分かりました！

LTspice でシミュレーションしてみる

図 10 の上の回路を LTspice で作ってシミュレーションしてみました。シミュレーション回路を図 11 に示します。上がアキュムレータ・ブロックで、下がディファレンシエータ・ブロックです。アキュムレータ・ブロックは連続時間系でのシミュレーションなので、数値が飽和しないように 0.999 倍としてあります。

この回路で、$H_A(z), H_D(z), H_{AD}(z)$をシミュレーションしてみます。

アキュムレータ・ブロック$H_A(z)$のシミュレーション結果を図 12 に示します。数値が大きくなるので縦軸は対数表示にしてあります。図 13 はディファレンシエータ・ブロック$H_D(z)$のシミュレーション結果です。

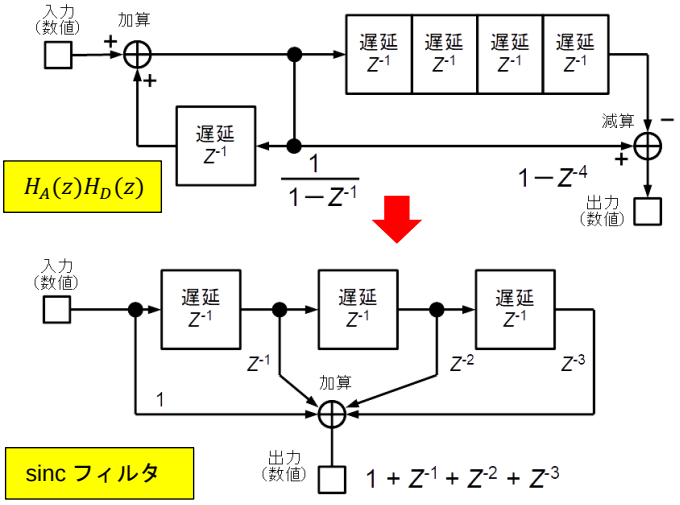

図 10. $H_A(z)H_D(z)$は平均化（sinc）フィルタになる
［上：$H_A(z)H_D(z)$をひとつずつ従属接続したもの。
下：平均化（sinc）フィルタ］

続いて図 14 は、全体［$H_{AD}(z) = H_A(z) \cdot H_D(z)$］のシミュレーション結果です（というより、グラフ機能で$H_A(z) \cdot H_D(z)$を計算させたものです）。sinc 関数になっていることが分かります。なお 0Hz, 1Hz のところで数値がゼロになっていますが、$\mathrm{sinc}(x) = \sin(x)/x$であり、$x = 0$とした場合、本来は$\mathrm{sinc}(0) = 1$になります。

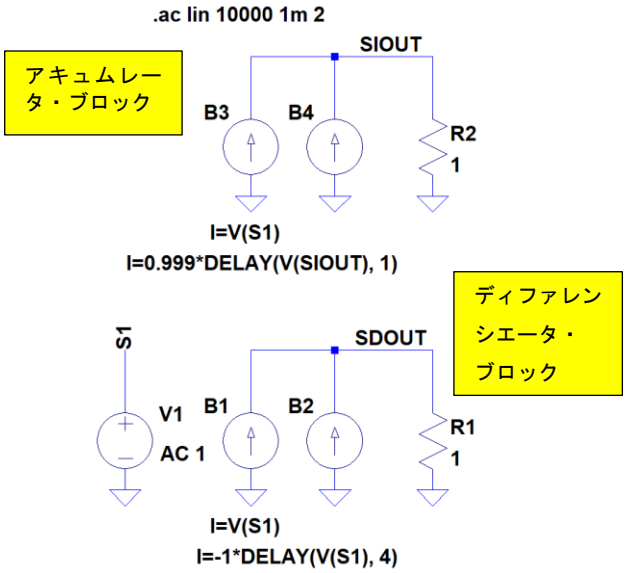

図 11. $H_A(z), H_D(z), H_{AD}(z)$を LTspice でシミュレーションしてみる回路（上：アキュムレータ・ブロック$H_A(z)$、下：ディファレンシエータ・ブロック$H_D(z)$。$H_{AD}(z)$は$H_A(z)$と$H_D(z)$を乗算することで計算する

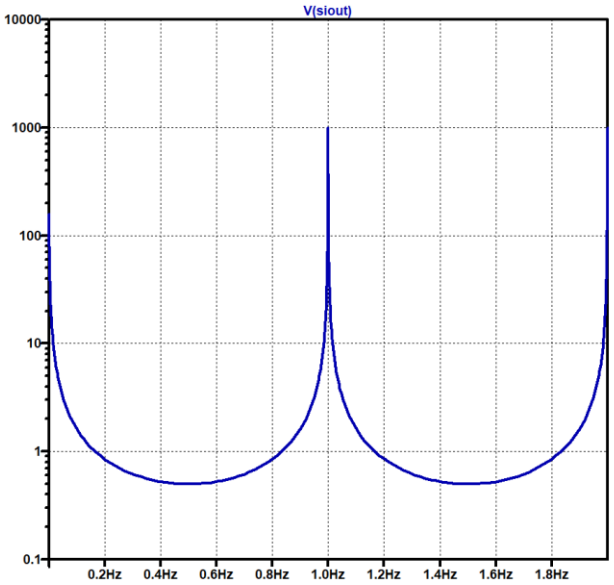

図 12. アキュムレータ・ブロック$H_A(z)$のシミュレーション結果（数値が大きいので縦軸は対数にしてある）

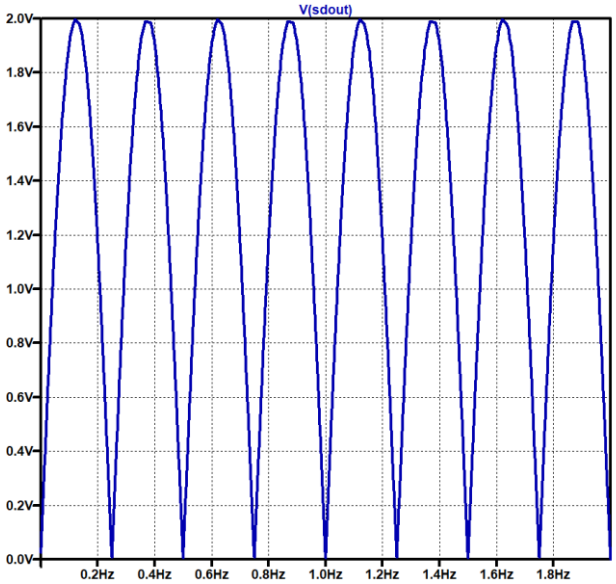

図 13. ディファレンシエータ・ブロック$H_D(z)$のシミュレーション結果

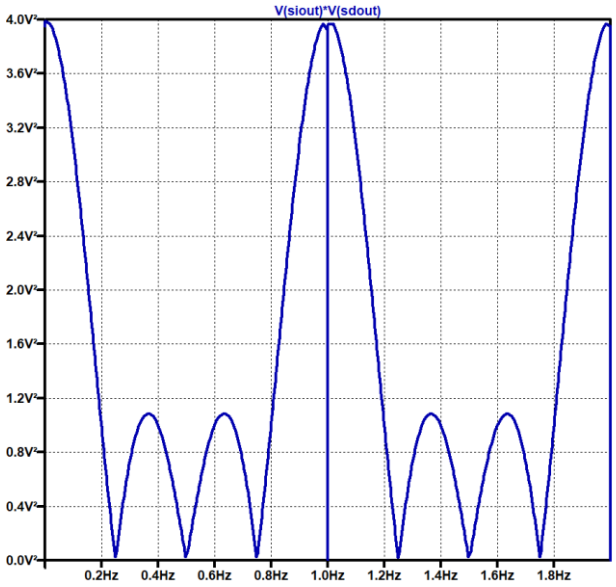

図 14. 全体$H_{AD}(z) = H_A(z)H_D(z)$のシミュレーション結果

sinc フィルタを 3 段接続することで sinc3 フィルタになる

式(2) ～ 式(9)で示したように、sinc フィルタはアキュムレータ・ブロックのひとつの回路$H_A(z)$と、ディファレンシエータ・ブロックのひとつの回路$H_D(z)$を従属接続したもので、

$$H_{AD}(z) = H_A(z) \cdot H_D(z) \quad (9\text{ 再掲})$$

一方で式(1)で表された AD7402 の sinc3 フィルタは

$$H(z) = H_A(z)H_A(z)H_A(z) \cdot H_D(z)\, H_D(z)H_D(z) \quad (1\text{ 再掲})$$

でしたが、この並べる順番を変えれば、

$$H(z) = [H_A(z) \cdot H_D(z)] \cdot [H_A(z) \cdot H_D(z)] \cdot [H_A(z) \cdot H_D(z)] \quad \cdots\cdots\cdots \quad (12)$$

として、sincフィルタが3段従属接続された回路になっていることが分かります。

Cascaded Integrator Comb （CIC）フィルタとも呼ばれる

このフィルタ形状は、Cascaded Integrator Comb (CIC) Filter とも呼ばれるものです。デジタル信号処理を習い始めたころ勉強した書籍[2]にCIC filterの説明があり、「なるほど」とは思っていたのですが、sinc フィルタと CIC フィルタが同じ特性（同じもの）だと分かった、気が付いたのは、アナログ・デバイセズに入社してからだいぶ経ったころだったのでした…。

この部分でデシメートしてよいものか

その、アナログ・デバイセズに入社したころ、この AD740x シリーズのデータシートを初めて見ることとなりました。VHDLは知っていたため、ここで紹介した Verilog のコードもある程度は読めたので、またそこにブロック図の記載もあったので、その動作を追うことができました。

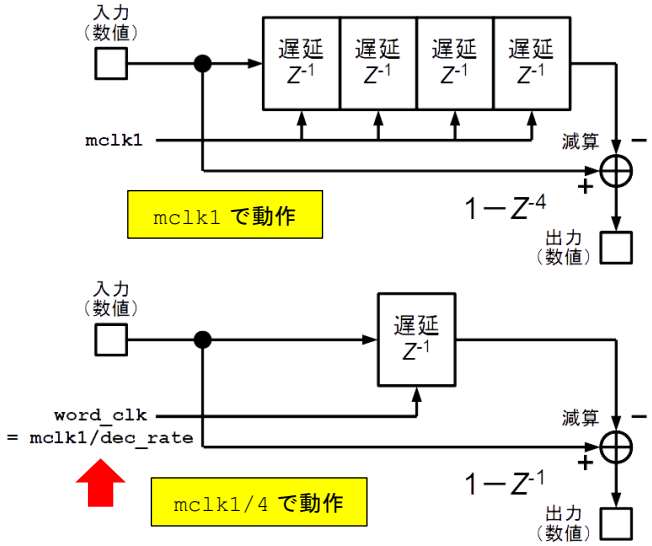

図 15. デシメーションされたクロックにより遅延系の構成が 4 遅延から 1 遅延になる

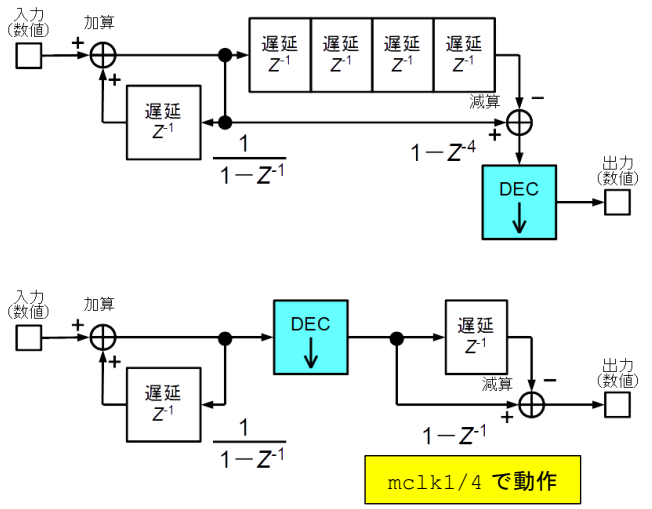

図 16. デシメーション操作自体はどちらで操作をおこなっても結局は同じ

最初にお話ししたとおり、もともとRFエンジニアだったこともあり、そのときは、「これでホントに sinc フィルタなのかいな？」と思ったものでした（このことは上記の検討で解決できたわけですが）。そしてさらに、以降に示すような別の疑問も生じてきたのでした。

ディファレンシエータ・ブロックはクロック分周で簡素化されている

先に示したように、ディファレンシエータ・ブロックでは、図7や図8のように、もともとのシステム・クロック mclk1（これはイコールΣΔ変調器の動作クロック、ビット・ストリームのレートでもあります）を dec_rate（デシメーション・レートDR）だけ分周（間引き）した、このブロック用のクロック word_clk が使われます。

このクロック word_clk（ここまでの例では$DR=4$で 1/4 に分周したもの）により、図15のように、4 遅延（z^{-4}）の遅延系の構成を 1 遅延（z^{-1}）に縮小できます。シフト・レジスタの段数を減らして簡素化できているのです。

なお最終的な AD 変換結果出力速度についても、デシメーション・レートDRでシステム・クロック速度からデシメーションされて出てきます。そのため図16のように、デシメーション操作をディファレンシエータ・ブロックで行っても、この出力のところで行っても、操作としては同じです。

ディファレンシエータ・ブロックで折り返しが生じるのではないか

上記の操作は、サンプリング周波数が$1/DR$だけ遅くなることになります（この例では 1/4）。ここで疑問が生じたのでした。「**ナイキスト帯域外の信号成分が折り返して、ナイキスト帯域内に現れるのではないか**」という疑問です。

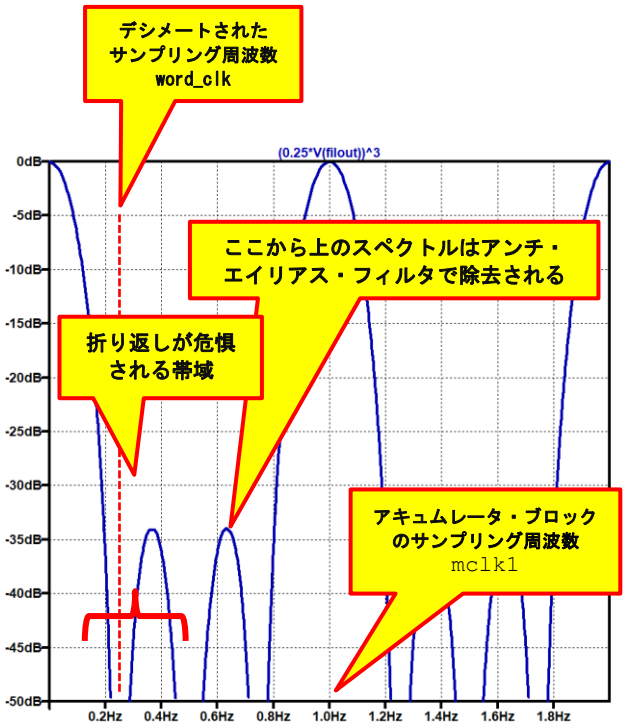

図 17. sinc3 のフィルタ特性とデシメートされたサンプリング周波数によりエイリアシングが生じるのでは？

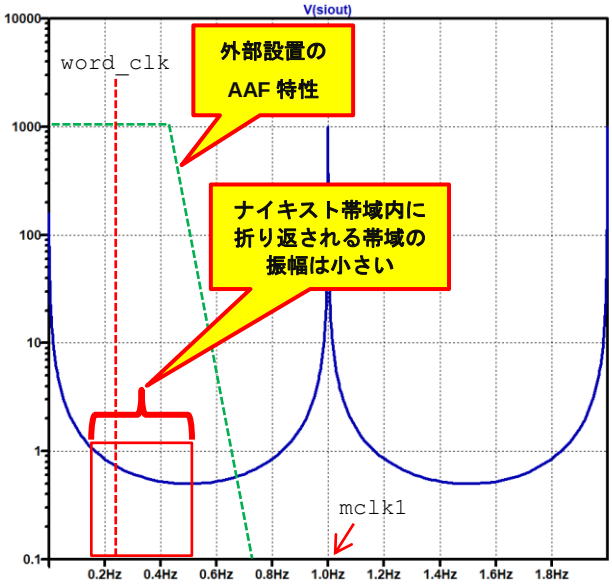

図 18. アキュムレータ・ブロック$H_A(z)$の伝達関数
（図 12 を再構成したもの）

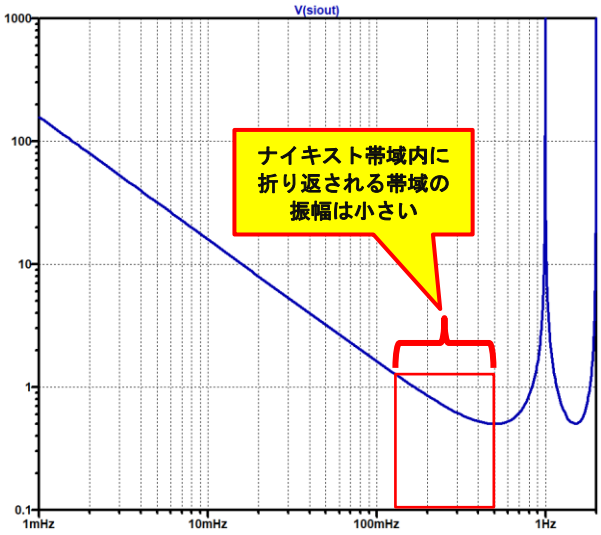

図 19. 図 18 の周波数軸を対数としたもの

TNJ-052 でも示したように、また一般的によく知られているように、アナログ信号のスペクトルはサンプリング定理により折り返されてデジタル信号として現れます。

sinc3 フィルタの伝達関数は図 17 のようになります。デシメーションにより、同図中に赤破線で示された「デシメートされたサンプリング周波数 word_clk」で再サンプルされます。そうすると word_clk/2～2×word_clk あたりのスペクトルが、word_clk のレートのナイキスト帯域に折り返されるのではないかというのが私の疑問でした。

アキュムレータ・ブロックを見ると問題なさそうに感じるけれども？

これについて考えてみましょう。図 18 に図 12 のアキュムレータ・ブロック$H_A(z)$の伝達関数のプロットに加筆したものを再掲します。後段のディファレンシエータ・ブロックのことは考えず、ここではこのブロックのみを考えます。

このブロックは mclk1 でサンプリングされ、積分器（累積 = 積分なので）としての周波数特性が図のように形成されます。また図中の緑破線のように、外部に設置するアンチ・エイリアシング・フィルタ（Anti-Aliasing Filter; AAF）によりナイキスト帯域外の余計な成分は落とされます。これらの合成がアキュムレータ・ブロック単体で得られる周波数特性です。図 19 に図 18 の周波数軸を対数としたものも示してみます。

ディファレンシエータ・ブロックに入るところでデシメーションされると

この信号は、つづいて後段のディファレンシエータ・ブロックで word_clk のレートでデシメーションされます（同図中の赤破線）。このレートにおけるナイキスト帯域外の周波数（word_clk/2～2×word_clk の 2nd, 3rd, 4th ナイキスト・ゾーンあたり）が折り返されますが、word_clk のナイキスト帯域内の伝達率の大きさに比べて、折り返される帯域外の振幅レベル（伝達率）は十分に小さく「は」なっています（「は」というところが意味深な…。以降につながるストーリでありまして）。ここだけ見ると、折り返しはとくに問題なさそうに見えますが…。

ディファレンシエータ・ブロックの伝達関数も考慮すると

つづいてディファレンシエータ・ブロックを見てみましょう。この部分はデシメーションされたレートで動作しています。

このブロックの伝達関数を図 11 の回路を使ってシミュレーションしてみます。その結果が図 20 です。DC では信号伝達率がゼロ、そして 0.25Hz ステップでゼロとなる点（これをヌル点といいます）が繰り返し生じています。「櫛（くし）状」になっていますね。このために Cascaded Integrated Comb（櫛）と呼ばれるわけです。

この伝達関数でアキュムレータ・ブロック$H_A(z)$の出力がフィルタリングされ、デシメーションにより word_clk（mclk1/4）で折り返されます。

ここで DC のあたりを見てください。DC のところは伝達率がゼロです。DC では、アキュムレータ・ブロック$H_A(z)$とこのディファレンシエータ・ブロック$H_D(z)$との合算（掛け算）で、出力は 1 になります。実はせっかく$H_A(z)$での積分動作により低域の伝達量が増大しているのもかかわらず、それが$H_D(z)$での微分動作で、その増大を打ち消してしまっているのですね…。

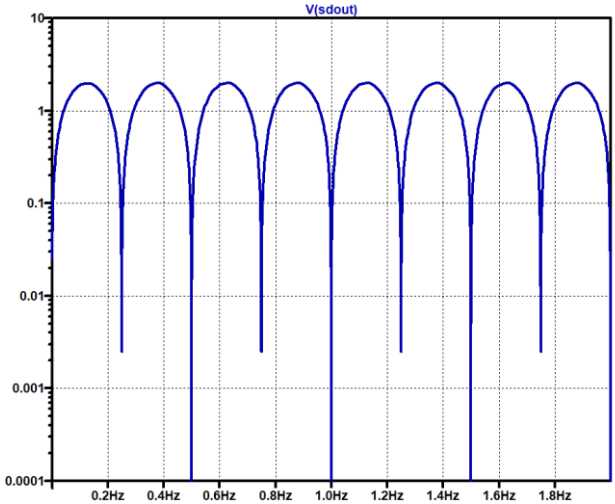

図 20. ディファレンシエータ・ブロック$H_D(z)$の伝達関数

その結果として、sinc フィルタ形状の周波数特性が結果的には形成されてしまいます。

アキュムレータ・ブロックで低域の伝達率がとても大きくなることから、「高域の折り返しの影響が低減するだろう」とおもいきや、結局は sinc フィルタ形状となり、それがそのままデシメーションで折り返されるという「もとの木阿弥」状態になっていることが分かります。なんと、というか、なるほど、という感じです。

答えをいうと「sinc フィルタの周波数特性とデシメーションにより折り返しがみえてくる」

ということが、私が感じていた疑問に対する答えのようです…。

アキュムレータ・ブロックの伝達率がディファレンシエータ・ブロックの伝達率でキャンセルされる。それにより「sinc 形状そのものの周波数特性が折り返される」。sinc フィルタには意外な折り返しの問題があったのですね…。

それでも実際のΣΔADC 回路では、DC に近い低域のところしか取り扱わないことが多いと考えられます。またある程度周波数が上昇してくると、sinc フィルタの特性により、検出される信号の振幅が低下するという問題点もあります。そのためこの折り返しはあまり問題にならないと考えられるものです。

それでもこのような折り返しが生じるんだということは理解しておくとよいでしょう。

ΣΔ変調器で生じる折り返しは外部にアンチ・エイリアシング・フィルタを設置しておく

示したように、もともとのシステム・クロック mclk1（ΣΔ変調器の動作周波数、ビット・ストリームのレート）で生じるエイリアシングは、ΣΔ変調器の前段に配置されるアンチ・エイリアシング・フィルタで除去する必要があります。

逆にいうとこの mclk1 でもエイリアシングが起こるため、mclk1 のレートに対するアンチ・エイリアシング・フィルタは外部に必要だということです。これはΣΔADC でも SAR ADC の場合と同じく注意が必要なことです。

外部にアンチ・エイリアシング・フィルタを設置すれば、システム・クロック mclk1 のレートで発生するエイリアシングは問題になることはありません。

アキュムレータとディファレンシエータの並べる順番は？

一般的にはアキュムレータ・ブロックを前段に、（デシメーションした）ディファレンシエータ・ブロックを後段に配置することが多くの解説書や web ページで説明されており、また「どちらを先にしても回路動作としては変わらない」という説明も多くみかけます。

しかし、ディファレンシエータ・ブロックを前段に配置して、そこでデシメーションすれば、ここで示したような特性が実現できないことが分かります（デシメーションしなければ問題ありません）。つまりデシメーションを行うのであれば、順番としては、アキュムレータ・ブロックを前段に、ディファレンシエータ・ブロックを後段に配置することが必須となります。

一方で説明してきたように、ディファレンシエータ・ブロックを後段に配置し、ここでデシメーションすれば、シフト・レジスタの段数を減らすことができ（出力ワードのビット幅だけ必要なのでロジック数が増えるため）、良好という点もあります。といっても現在の SoC FPGA ではこの増加は無視できるものでしょうが…。

ところで、どの書籍か記憶がないのですが、ディファレンシエータ・ブロックを先に置くと問題があるという記述を読んだことが（読んだような記憶が）あります。今思えば、引き算処理による残差の伝搬（アキュムレータ・ブロックでの累積）のことかなと思いますが、定かではありません。

まとめ

前回と今回で LTspice を使って平均化フィルタと sinc フィルタ／CIC フィルタについて考えてみました。CIC フィルタはその Z 変換の式を割り算してみると、平均化フィルタと等価になっていること、まさしく sinc フィルタになっていることがご理解いただけたかと思います。

ディファレンシエータ・ブロックでデシメートすることについては、sinc フィルタ／CIC フィルタの回路構成として、折り返しが生じてしまうことが分かりました。探究していけばいくほど、あらたな気づきを得ることができるわけですね。

ちょっとオマケ（オーバ・フローについて）

ちなみに図 9 のような構成でアキュムレータ・ブロック（積分回路）を作れば、DC 成分が加わったときに、アキュムレータの値がオーバ・フローしてしまいます。「ここは注意すべき」という記述を見ますが、CIC フィルタ構成として図 5 のように後段のディファレンシエータ・ブロックで差分計算が行われることを考えると、unsigned int（符号なし整数）で `acc3` が構成されているため、差分計算としてはオーバ・フローの前後でも正しい差分量を計算できることにも気がつきます。簡単な回路ですが、ホント、良くできてますね。

謝辞

今回の CIC フィルタの周波数特性の構成と折り返しの考察については、社内のエンジニアである E 氏とディスカッションさせていただき、この技術ノートを完成することができました。この場を借りて同氏にお礼を申し上げます。

参考文献

[1] A S Hornby; Oxford Advanced Learner’s Dictionary of Current English, Seventh Edition, Oxford University Press, 旺文社

[2] 中村 尚五; デジタルフィルタ （ビギナーズ）, 東京電機大学出版局

[3] Steven Xie; 高精度 ADC 用のフィルタ設計における課題と検討事項, Analog Dialogue, Vol. 50, No. 2, Apr. 2016, Analog Devices, http://www.analog.com/media/jp/analog-dialogue/volume-50/number-2/articles/practical-filter-design-precision-ADCs_jp.pdf

[4] アプリケーションノート AN-283, シグマ・デルタ ADC/DAC の原理, Analog Devices

[5] Matthew P. Donadio; CIC Filter Introduction, http://dspguru.com/files/cic.pdf