AVR223：AVRでのﾃﾞｼﾞﾀﾙ濾波器

要点

- ﾃﾞｼﾞﾀﾙ濾波器の実装
- 係数とﾃﾞｰﾀの尺度調整
- 4次FIR濾波器の高速実装
- 2次IIR濾波器の高速実装
- 最適化に関する方法

8-bit AVR® マイクロ コントローラ

応用記述

1. 序説

外部ｱﾅﾛｸﾞ源/感知器からのﾃﾞｰﾀ処理を伴う応用は通常、ﾃﾞｰﾀの或る種のﾃﾞｼﾞﾀﾙ濾波を必要とします。極端に高い濾波性能に関しては通常ﾃﾞｼﾞﾀﾙ信号ﾌﾟﾛｾｯｻ(DSP)が選択されますが、多くの場合でこれらはその使用に対して高価すぎます。これらの状況では8ﾋﾞｯﾄや16ﾋﾞｯﾄのﾏｲｸﾛ ｺﾝﾄﾛｰﾗ(MCU)が面白くなってきます。これらは安価、効率的で、DSPが往々にして持たない必要とするI/O機能と通信単位部の全てを持ちます。

AtmelのAVRﾏｲｸﾛ ｺﾝﾄﾛｰﾗはそれらの強力な基本設計、強力な命令、組み込み10ﾋﾞｯﾄA/D変換器(ADC)のため、信号処理応用に対して優秀です。megaAVR®系列は更にﾊｰﾄﾞｳｪｱ乗算器を持ち、これは信号処理応用に於いて重要です。

この資料はAVRﾊｰﾄﾞｳｪｱ乗算器の使用、累積器機能用の汎用ﾚｼﾞｽﾀの使用、固定小数点構造での算法実装時の係数尺度調整法、実装を最適化/変更するのに可能な方法の提示に集中します。2つの実装例が含まれています。

例えﾃﾞｼﾞﾀﾙ濾波器の原理がこの応用記述の焦点でないとは言え、幾許かの基本は網羅されます。ﾃﾞｼﾞﾀﾙ濾波器の原理のもっと詳細な文献の関連一覧がこの資料の最後に含まれます。

2. 一般的なﾃﾞｼﾞﾀﾙ濾波器

全てのﾃﾞｼﾞﾀﾙ、線型時不変(LTI:Linear Time-invariant)濾波器は**式1.**の形式の差分方程式によって記述することができます。出力信号は$y[n]$、入力信号は$x[n]$によって定義されます。

式1. ﾃﾞｼﾞﾀﾙ濾波器に関する一般的な差分方程式

$$\sum_{i=0}^{M} a_i \times y[n-i] = \sum_{j=0}^{N} b_j \times x[n-j]$$

濾波器はその次数と係数(a_iとb_j)によって独自に定義されます。濾波器の次数はMとNの最大として定義され、計算で使われる最長遅延を記します。通常、係数はa_0が1に等しくなるように尺度調整されることに注意してください。そこで濾波器の出力は**式2.**で示されるように計算され得ます。

式2. 濾波器出力に対する差分方程式

$$y[n] = \sum_{j=0}^{N} b_j \times x[n-j] + \sum_{i=1}^{M} (-a_i) \times y[n-i]$$

$x[n]$がｲﾝﾊﾟﾙｽ($n=0$に対して1、$n\neq0$に対して0)の場合、その出力は濾波器のｲﾝﾊﾟﾙｽ応答($h[n]$)と呼ばれます。

濾波器はMの値から2つの形式の1つとして分類することができます。

- $M=0$に対する、有限ｲﾝﾊﾟﾙｽ応答(FIR)
- $M\neq0$に対する、無限ｲﾝﾊﾟﾙｽ応答(IIR)

これら2つの濾波器形式間の違いは帰還です。IIR濾波器に関して、その出力試料は再帰的に(換言すると、入力試料に加えて以前の出力から)計算されます。そして用語の有限/無限は(現実の実装に於ける量子化効果を無視した)濾波器のｲﾝﾊﾟﾙｽ応答の長さを記述します。$N=0$のIIR濾波器が濾波器の特別な場合で、"全極(all-pole)"と呼ばれることに留意してください。これら2つの濾波器の種類の更なる情報についてはこの資料の最後で提示される文献一覧を参照してください。

(訳補) この応用記述では係数を2のべき乗倍し、そして積和出力を逆に1/(2のべき乗)することにより、演算を整数型で処理する方法を記述しています。

Rev. 2527B-07/08, 2527AJ3-03/21

デジタル濾波器は度々複合周波数領域のZ領域で記述されます。式1.のZ変換が式3.で示されます。

式3. 一般的なデジタル濾波器のZ変換

$$Z\left\{\sum_{i=0}^{M} a_i \times y[n-i] = \sum_{j=0}^{N} b_j \times x[n-j]\right\} = Y(z) \times \sum_{i=0}^{M} a_i \times z^{-i} = X(z) \times \sum_{j=0}^{N} b_j \times z^{-j}$$

濾波器に対する伝達関数$H(z)$は通常、簡潔な表現を与えて容易な周波数分析を可能とするのにそれが役立つものとして提供されます。伝達関数は式4.で示されるように、Z領域に於ける濾波器の出力と入力間の比として定義されます。伝達関数が濾波器のインパルス応答($h[n]$)のZ変換であることに注意してください。

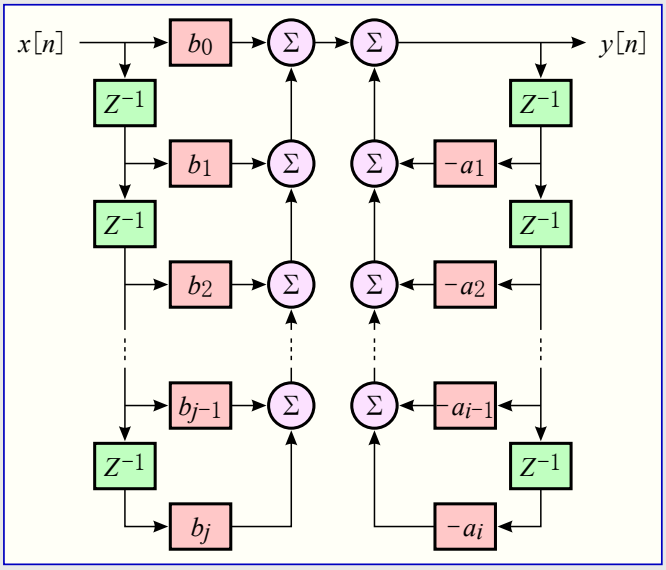
式4. 一般的なデジタル濾波器の伝達関数

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{j=0}^{N} b_j \times z^{-j}}{\sum_{i=0}^{M} a_i \times z^{-i}} = \frac{\sum_{j=0}^{N} b_j \times z^{-j}}{1 + \sum_{i=1}^{M} a_i \times z^{-i}}$$

ここで見られるように分子は濾波器関数の順送部を、そして分母が帰還を記述します。Z領域のより多くの情報についてはこの資料の最後で提示される文献一覧を参照してください。

与えられた伝達関数から濾波器を実装する目的に関しては、Z領域に於けるzが遅延要素を表し、指数が試料単位での遅延長を定義することを知るので充分です。図2-1.は直接形式1表現に於ける標準的なデジタル濾波器でのこれを図解します。

図2-1. 標準的なデジタル濾波器の直接形式1表現



3. 濾波器実装の考察

与えられたMCU構造で濾波器を実装する時に多くの問題が考慮されなければなりません。例えば以下です。

- 入力と出力の分解能(ビット数)は最大許容濾波器利得と処理量に影響を及ぼします。
- 濾波係数の分解能は周波数応答と処理量に影響を及ぼします。
- 濾波器次数は処理量に影響を及ぼします。
- 小数点濾波係数は整数型乗算器で使われる時に何らかの考慮が必要です。

これらとその他の実装の問題がこの項で検討されます。

加えて、濾波のコードを理解するのにAVRハードウェア乗算器と仮想累積器についての知識が重要なので、それらの簡単な説明もここで行われます。

3.1. AVRハードウェア乗算器

AVRが8ビット構造なのでハードウェア乗算器は8ビット×8ビット=16ビットの乗算器です。この応用記述に於いて乗算器はMUL,MULS,MULSUの3つの異なる命令を使って実施されます。これらの命令は各々、符号なし、符号付、それと符号付き×符号なしの乗算です。

濾波器の算法は積の総和です。最初の積は2つのNビット値間の"単純な"乗算(MUL)で、2Nビットの結果を提供します。この乗算は各々の被乗数から一度に1バイト間(の乗算)で作られます。結果は2Nビットの"累積器"に格納されます。次の積が計算され、そして"累積器"に加算され、所謂、積和演算(MAC)です。

この応用記述の濾波器例は次の2つの異なる乗算操作を使って作ります。

- 16×16=24ビット符号付乗算
- 16×16=24ビット符号付積和

これらは24ビットの結果を持つ符号付きの16ビット対16ビットです(実装例の入力試料と係数が全16ビット範囲を使わないので結果は32ビットよりも小さくなります)。

AVRは固定小数点乗算用の命令も持っていますが、この応用記述でそれらが使われないことに注意してください。それらと他の乗算命令についてのより多くの情報に関しては「AVR201:AVRハードウェア乗算器の使い方」応用記述を参照してください。

3.2. AVR仮想累積器

AVRは専用の累積器を持たず、代わりにAVRは汎用ﾚｼﾞｽﾀのどれにも"仮想累積器"の形成を可能にします。この資料を通してこの仮想累積器は(単に)"累積器"として参照され、とは言え、他の構造で使われる通常の累積器よりもずっと柔軟です。

例として、2つの12ﾋﾞｯﾄ値を乗算(MUL)するためにAVRで24ﾋﾞｯﾄ累積器が必要とされる場合、3つの8ﾋﾞｯﾄ汎用ﾚｼﾞｽﾀが24ﾋﾞｯﾄ累積器に結合されます。積和(MAC)が40ﾋﾞｯﾄ累積器を必要とするなら、この累積器を形成するのに5つの汎用ﾚｼﾞｽﾀが結合されます。この累積器の柔軟性を使うことは、累積器容量が32ﾋﾞｯﾄ以下に固定されている場合に必要とされる、結果または中間結果が積和(MAC)操作中に前後に移動されるべきことが全くないことを保証します。

AVR累積器の柔軟性は固定小数点(FP)算法での溢れを避けるための重要な道具で、これが次に検討されます。

3.3. 固定小数点値の溢れ

溢れは濾波器算法に於いて2箇所、算法の中間結果と濾波器の出力で起きるかもしれません。

3.3.1. 中間結果での溢れ回避

濾波器算法の中間結果で溢れが起きるかもしれないことの理由は以下です。

- 分解能N1とN2を持つ2つの値の乗算は(N1+N2)ﾋﾞｯﾄの結果を生成し得ます。
- 2つの値の加算はより大きな分解能の被演算子(演算対象)よりも1ﾋﾞｯﾄ多い分解能を持つ和(結果)を生成し得ます。

式5.で記述される4次FIR濾波器を考察してください。

出力は5つの乗算の総和です。ﾃﾞｰﾀと係数の両方が符号付き16ﾋﾞｯﾄとの仮定で、算法は殆どに於いて**式6.**で計算されるように34ﾋﾞｯﾄの累積器が必要です。

Nは必要とするﾋﾞｯﾄ数、Kは入力試料と係数の(符号ﾋﾞｯﾄを除いた)ﾋﾞｯﾄ分解能、Mは加算回数です。(式後尾で)付加されている1ﾋﾞｯﾄは符号ﾋﾞｯﾄです。この場合に於ける累積器はこれらの操作のために起きるかもしれない最大絶対値を保持するために5つの汎用ﾚｼﾞｽﾀ(40ﾋﾞｯﾄ)が必要です。

IIR濾波器では濾波算法に於いて出力試料が使われることに留意してください。出力が入力よりも高い分解能を持つ場合に累積器は出力の分解能に従って尺度調整されることが必要です。

式5. 4次FIR濾波器の差分方程式

$$y[n] = \sum_{j=0}^{4} b_j \times x[n]$$

式6. 4次FIR濾波器で必要な累積器分解能

$$\begin{aligned} N &\geqq 2\times K + \log_2(M) + 1 \\ &= 2\times 15 + \log_2(5) + 1 \\ &\fallingdotseq 33.32 \\ N &= 34 \end{aligned}$$

3.3.2. 出力での溢れ回避

出力段での溢れを避けるため、出力段で利用可能な分解能で結果を表現することが可能なように、濾波器の利得は制限されなければなりません。利得に於ける限度は勿論、入力信号の(出力と比較した)分解能と範囲に依存します。

溢れを避けるための最も保守的な判断基準は、入力の最大絶対値で乗算された濾波器のｲﾝﾊﾟﾙｽ応答の絶対総和が出力の最大絶対値を超え得ないことです。**式7.**がこの基準を示します。

ｲﾝﾊﾟﾙｽ応答がこの基準を満たさない場合、絶対総和を充分に低減する係数での乗算が単純に必要です。

式7. 濾波器出力での溢れを避けるための保守的な基準

$$|X_{MAX}| \times \sum_{n=0}^{\infty} |h[n]| \leqq |Y_{MAX}|$$

$$\sum_{n=0}^{\infty} |h[n]| \leqq \frac{|Y_{MAX}|}{|X_{MAX}|}$$

符号付き整数に関する正の範囲の最大値が負の範囲の絶対最大値よりも1小さなことに留意してください。入力がMﾋﾞｯﾄ、出力がNﾋﾞｯﾄと仮定して、**式8.**は最悪の筋書きに対する基準を示します。

式8. 符号付き整数に対する"最悪時"の保守的な基準

$$\sum |h[n]| \leqq \frac{2^{N-1}-1}{2^{M-1}}$$

この基準の遂行は溢れが全く起こらないことを保証しますが、欠点は濾波器利得のかなりの減少です。この基準のため、入力信号の特性は過度に悲観的になるかもしれません。

別の一般的な基準、それは(正弦波のような)狭帯域信号に対して良好で、入力の絶対最大値で乗算される濾波器の絶対最大利得が出力の絶対最大値を超え得ないと言います。**式9.**がこの基準を示します。

これがこの応用記述の濾波器実装で使われる基準です。入力と出力に於ける同じ分解能で、濾波器は利得1(0dB)を越えないでしょう。

利得に於ける限度が入力信号の特性に依存し、このために最適な限度を探すために或る程度実験が必要かもしれないことに注意してください。

式9. 狭帯域信号での溢れを避けるため基準

$$\max_{H(\omega)} |X_{MAX}| \times H(\omega) \leqq |Y_{MAX}|$$

$$\max_{H(\omega)} H(\omega) \leqq \frac{|Y_{MAX}|}{|X_{MAX}|}$$

3.4. 係数の尺度調整

別の重要な問題は固定小数点(FP)構造に於ける濾波係数の表現です。FP表現は値が整数でなければならないことを必然的に意味する訳ではありません。言及されたように固定小数点乗算も利用可能です。けれども、この応用記述では整数乗算だけが使われ、従って焦点は整数表現にあります。

本質的に最も正確な数の表現は可能な限り多くのﾋﾞｯﾄを使うでしょう。整数乗算で小数型濾波係数を使う目的に関し、この問題はそれらの表現でどんな溢れも起きない最大公倍数により、全ての係数を尺度調整することに要約されます。この尺度調整は$a0$係数にも適用し、故に(特にIIR濾波器に対して)正しい値を得るために出力の縮尺が必要です。ﾊｰﾄﾞｳｪｱでの除算は実装されておらず、故に2のべき乗での乗除算がﾋﾞｯﾄ移動で容易に行い得るので尺度調整率は2^kの形式であるべきです。係数尺度調整の原理と後続する結果の縮尺は**式10.**で示されます。

式10. 濾波係数の尺度調整と結果の縮尺

$$2^k \times \sum_{i=0}^{M} a_i \times y[n-i] = 2^k \times \sum_{j=0}^{N} b_j \times x[n-j]$$

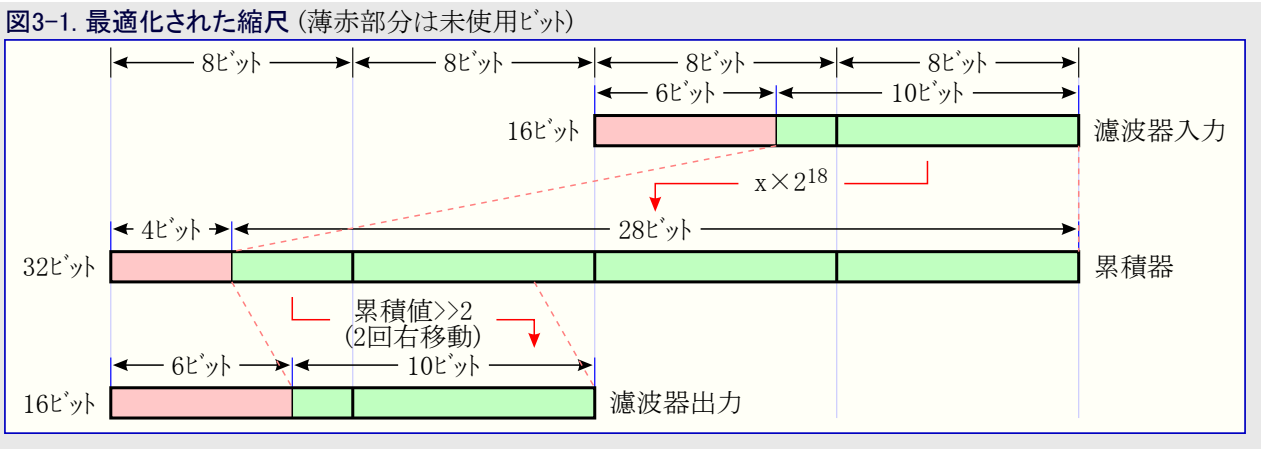
$$y[n] = \left\{ \sum_{j=0}^{N} (2^k \times b_j) \times x[n-j] + \sum_{i=1}^{M} (-2^k \times a_i) \times y[n-i] \right\} >> k$$

縮尺時に符号ﾋﾞｯﾄが保護されなければならないことに注意してください。これはASR(算術的右移動)命令で簡単に行われます。

例として、濾波係数b_j={0.9001,-0.6500,0.3000}を考察してください。符号付き16ﾋﾞｯﾄ整数表現が使われる場合、尺度調整された係数は[-2^{15}～$2^{15}-1$]=[-32768～32767]の範囲の値を持たなければなりません。本質的に最大の絶対値を持つ係数は最大尺度率を制限する1つです。この場合、何の溢れもなしで可能な最大尺度率は2^{15}です。(先の)値の尺度調整された係数の丸め結果は{29494, -21299,9830}で、(縮尺された)絶対丸め誤差は概ね{1.5×-10^5,6.1×-10^6,1.2×-10^5}です。

指数kが8の倍数以上なら、縮尺の最適化が可能です。これがその場合なら、ﾌﾟﾛｸﾞﾗﾑは結果のﾊﾞｲﾄを単純に"無視"できます。32ﾋﾞｯﾄの結果が2^{18}で縮尺されるべき例が**図3-1.**で図解されます。これは上位16ﾋﾞｯﾄ(MSB)を2度ﾋﾞｯﾄ移動することによって達成されます。

図3-1. 最適化された縮尺 (薄赤部分は未使用ﾋﾞｯﾄ)



3.4.1. 縮尺の影響

中間結果での溢れを避けるのに何故ﾋﾞｯﾄを追加するのかで迷うかもしれませんが、それは基本的に結果を指定された分可能に合わせるために"それらを捨てます"。その説明は計算中の精度のためにﾋﾞｯﾄ数が必要とされ、濾波器が利得1を持ち、そして出力が整数として解釈されるべきであることです。

利得1を持つ濾波器に関して、係数は小数、換言すると1未満で、従って乗算は小数値を実際に表現するﾋﾞｯﾄを追加します。けれども、総和はより高い意味を表現するﾋﾞｯﾄを追加します。しかし、濾波器の利得1のため、これらのﾋﾞｯﾄは結果に於いて決して使われません。濾波器の出力は入力のそれよりも大きな絶対値を得ることがなく、従って入力と同じ整数範囲での表現を出力に許します。

縮尺は単に結果の小数部を取り去り、ただ求めた結果の整数部を残します。明らかにこれは精度が減ったことを意味します。これはIIR濾波器が帰還を持つため、IIR濾波器で重要です。この精度減の影響が問題なら、それは2つの方法で低減することができます。

- 出力が利用可能な範囲全体を使うように濾波器利得を最大にしてください。
- 出力と濾波器利得の両方の分解能を増やしてください。

2つの内の後者だけは、より大きな累積器を必要とし得るので、ｺｰﾄﾞ量と処理量に影響を及ぼし得ます。

3.4.2. 処理量を増やすための分解能減少

濾波算法の速度向上のため、入力試料や係数の分解能減少が望ましいのかもしれず、このために累積器の容量が減らされ得ます。より小さな累積器は算法が試料と濾波係数の乗算毎により少ない操作を必要とすることを意味します。けれどもこれを行う前に以下の2つの問題が考慮に取り入れられる必要があります。

- 入力試料分解能の削減は系内に雑音が持ち込まれることを意味し、これは一般的に好まれません。
- 濾波係数分解能の削減は達成に対して望む濾波器応答が困難になるかもしれないことを意味します。

処理量を増す他の方法については8頁の「**濾波器実装の最適化**」をご覧ください。

4. 濾波器実装

この応用記述の濾波器例はIAR EWAVRｺﾝﾊﾟｲﾗ5.03A版を使って開発、ｺﾝﾊﾟｲﾙされています。

実装で使った濾波器係数はこの目的用に作成されたｿﾌﾄｳｪｱを用いて計算されています。これを行うことができるｿﾌﾄｳｪｱは多過ぎるほどあります。ｿﾌﾄｳｪｱ ﾌﾟﾛｸﾞﾗﾑはMatlab™のような高価な数学ﾌﾟﾛｸﾞﾗﾑからｳｪﾌﾞ上のJavaｱﾌﾟﾚｯﾄに及びます。この応用記述の最後に含まれる文献一覧に於いて濾波器係数計算の話題を扱うｳｪﾌﾞｻｲﾄの一覧が提供されます。代替は手計算による"辛い"方法で係数を計算することです。濾波器係数を計算する(のとそれらの濾波器の安定性を調査する)ための方法は[1.]と[2.]で記述されています。

4次高域通過(HP)FIR濾波器と2次帯域通過(BP)IIR濾波器の2つの異なる濾波器が実装され、符号付き10ﾋﾞｯﾄ試料が使われます。FIR濾波器は符号付き13ﾋﾞｯﾄ係数を使い、一方IIR濾波器は符号付き12ﾋﾞｯﾄ係数を使います。これは必要とされる最大累積器容量が24ﾋﾞｯﾄであることを保証します。

濾波器は効率性の理由のため、ｱｾﾝﾌﾞﾘ言語で実装されます。この実装はCから濾波器関数を呼び出せるような方法で作られています。濾波器関数を呼ぶのに先立って、濾波器節点(ﾉｰﾄﾞ)(遅延素子のﾒﾓﾘ)が初期化されていることを必要とします。さもなければ濾波器の始動条件が未知にされます。全ての濾波器に対して、濾波器関数の初期化と呼び出しをするCｺｰﾄﾞ例が提供されます。

濾波器で必要とする全てのﾊﾟﾗﾒｰﾀが走行時に渡され、従って濾波器関数は追加のｺｰﾄﾞ空間の必要なしに複数の濾波器を実装するために再使用することができます。これは濾波器の縦列接続も許します。縦列で1つの濾波器の出力を次の濾波器の入力へ供給することにより、より高次の濾波器を形成するのに度々複数の2次濾波器が使われます。けれども、縦列での各濾波器からの出力が次の濾波器に供給される前に縮尺されるので、最終出力が期待するようにならないかもしれません。これは濾波器間で精度が失われるためです。本質的にこれの影響は縦列長増加と共にもっと明白になります。

濾波器に於ける高い処理量が非常に重要なので、この実装は濾波器の高速実行に全てを集中します。ｺｰﾄﾞ量低減、更なる処理量増加、使用ﾒﾓﾘ低減のための代替実装の示唆については8頁の「濾波器実装の最適化」をご覧ください。

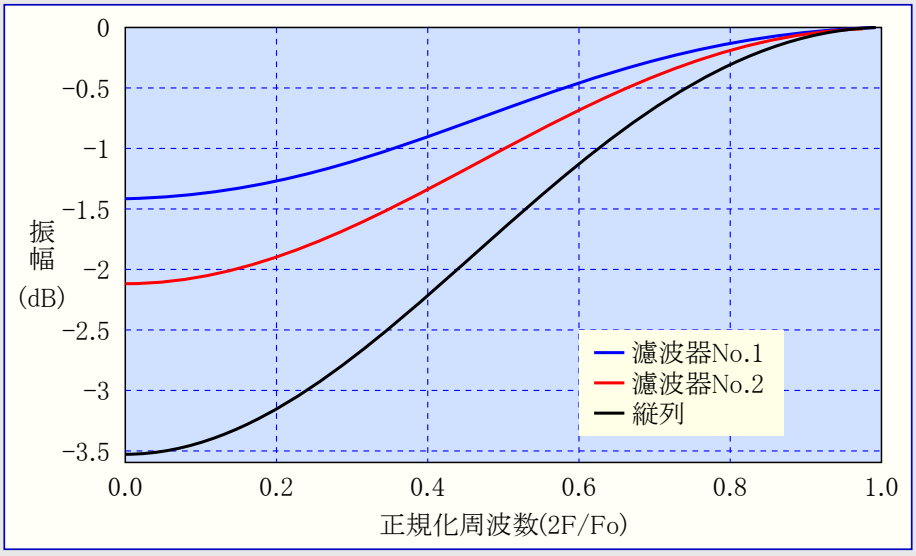
4.1. 4次FIR濾波器

縦列技法の実演目的のため、この濾波器は2つの2次高域通過(HP)濾波器を使うことによって実装されます。両濾波器はﾊﾐﾝｸﾞ窓を使い、[1.]で記述される窓関数法で作られました。濾波器ﾊﾟﾗﾒｰﾀは**表4-1.**で示されます。**図4-1.**は両濾波器の個別と縦列での振幅応答(周波数特性)を示します。

表4-1. 2次FIR濾波器ﾊﾟﾗﾒｰﾀ

濾波器	次数	ｶｯﾄｵﾌ	係数 (b_0,b_1,b_2)	尺度調整	尺度調整後係数 (b_0,b_1,b_2)
No.1	2	0.4	−0.0373, 0.9253, −0.0373	2^{12}	−153, 3790, −153
No.2	2	0.6	−0.0540, 0.8920, −0.0540	2^{12}	−222, 3653, −222

図4-1. FIR濾波器の振幅応答(周波数特性)



濾波器ルーチンはそれらを効率的にするためにアセンブリ言語で実装されます。けれども、濾波器関数を呼ぶのに先立って濾波器パラメータと濾波器節点(ノード)は初期化が必要です。この初期化はCで行われます。濾波係数と濾波器節点を含む構造体が各々の濾波器に対して定義されます。構造体は次のように定義されます。

```
struct FIR_filter{
  int filterNodes [FILTER_ORDER];                  // メモリ上の濾波器節点(ノード)
  int filterCoefficients [FILTER_ORDER+1];         // メモリ上の濾波器係数
} filter04 = {0, 0, B10, B11, B12},                // 濾波器No.1初期化
  filter06 = {0, 0, B20, B21, B22};                // 濾波器No.2初期化
```

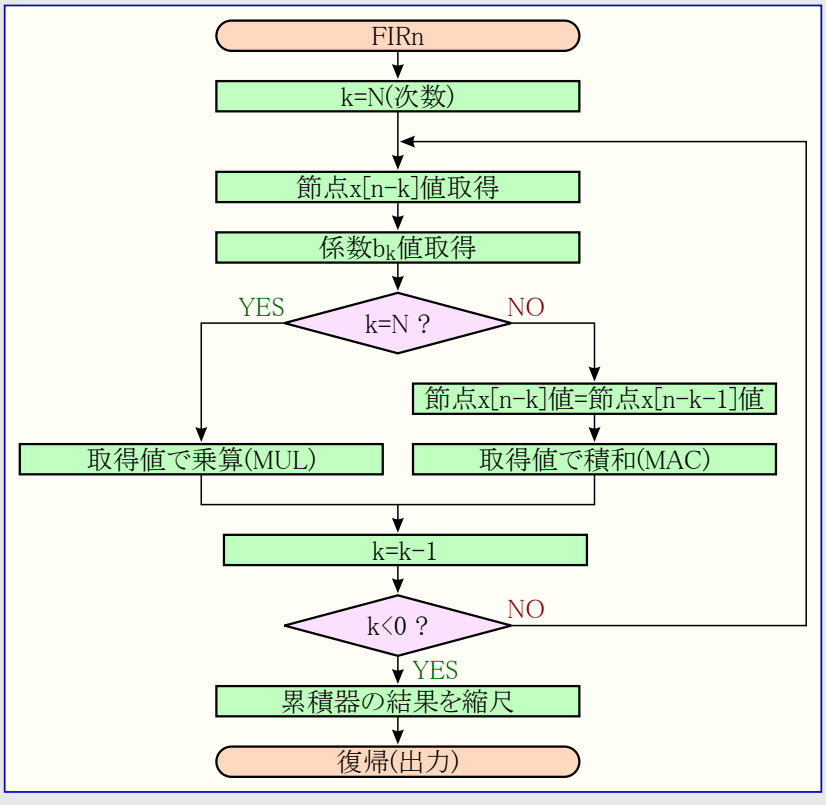
filterNodes配列は以前の入力試料を保持するFIFO緩衝部として使われます。filterCofficients配列は濾波器の順送係数に使われます。一旦濾波器が初期化されると、濾波器関数を呼び出すことができます。濾波器関数は次のように定義されます。

```
int FIR2(struct FIR_filter *myFilter, int newSample);
```

Zレジスタがデータの間接アドレス指定に使えるため、換言すると、ポインタ操作のため、最初に濾波器構造体へのポインタがZレジスタに複写されます。そして算法の中核は走行する準備が整います。試料(節点)が取得され、相手となる係数で乗算(MUL)されます。積は24ビット幅の累積器に加算されます。全ての試料と係数が乗算と累積(MAC)されると、結果が縮尺されて返されます。これは**図4-2.**の流れ図で見ることができ、これはFIR濾波算法に於ける流れの一般的な説明です。

図4-2.の流れ図は繰り返しを使って実装された場合としての算法を示し、実際には算法が一直線状のコードとして実装されていることに注意してください。この繰り返しは単に見易さを改善するために使われています。

図4-2. 一般的なFIR濾波器算法



言及したように、2つの濾波器が縦列で使われます。Cファイルでは、最初に濾波器関数への引数として入力試料で濾波器No.1を通し、次に同じ関数への引数として最初の濾波器の出力で濾波器No.2を通すことにより、これは単純に行われます。

4.1.1. 濾波算法の性能

表4-2.は単独2次FIR濾波器の性能を示します。濾波に関する数は濾波算法(MULとMAC操作、節点(ノード)更新)のコードに対してで、付随処理に関する数は(縮尺を含む)濾波器関数の残りに対してです。関数呼び出しと対応する復帰は含まれません。

表4-2. 2次FIR濾波器の性能

命令数(濾波+付随処理)	**実行周期数**(濾波+付随処理)	**濾波器効率**(周期数/次)
50+10	76+10	38

濾波器の次数変更は累積器容量の変更を必要とするかもしれず、そしてこれは次当たりの濾波の命令/周期数も変えることに注意してください。

4.2. 2次IIR濾波器

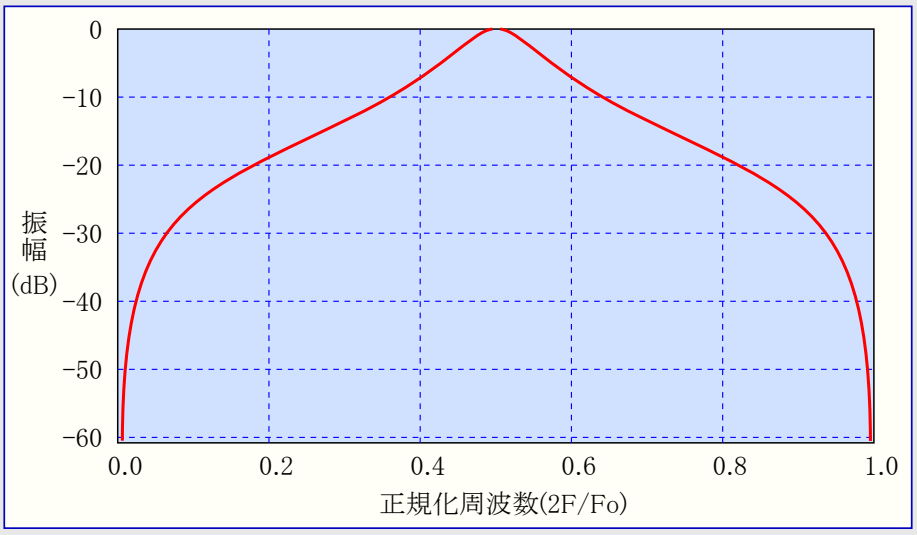
この実装についてはﾊﾞﾀｰﾜｰｽ帯域通過(BP)濾波器が設計されました。この濾波器のﾊﾟﾗﾒｰﾀは表4-3.で、振幅応答は図4-3.で示されます。

表4-3. 2次IIR濾波器ﾊﾟﾗﾒｰﾀ

次数	ｶｯﾄｵﾌ	係数 ($b_0,b_1,b_2:a_0,a_1,a_2$)	尺度調整	尺度調整後係数 ($b_0,b_1,b_2:a_0,a_1,a_2$)
2	0.45 : 0.55	0.1367, 0, −0.1367 : 1.0000, 0.0000, 0.7265	2^{11}	280, 0, −280 : 2048, 0, 1488

a_0係数が濾波器出力を縮尺しなければならない要素なので、それはCｺｰﾄﾞに於いて定義されませんが、縮尺のためにｱｾﾝﾌﾞﾘ言語ｺｰﾄﾞで暗黙のうちに定義されています。

図4-3. 2次IIR濾波器の振幅応答(周波数特性)



濾波器ﾊﾟﾗﾒｰﾀと濾波器節点(ﾉｰﾄﾞ)を含む構造体が濾波器用に定義されています。濾波器関数が呼ばれる前に濾波器は初期化されるべきです。構造体は次のように定義されます。

```
struct IIR_filter{
  int filterNodesX[FILTER_ORDER];                  // 濾波器節点(ﾉｰﾄﾞ),x(n-k)格納
  int filterNodesY[FILTER_ORDER];                  // 濾波器節点(ﾉｰﾄﾞ),y(n-k)格納
  int filterCoefficientsB[FILTER_ORDER+1];         // 濾波器順送係数
  int filterCoefficientsA[FILTER_ORDER];           // 濾波器帰還係数
  } filter04_06={0, 0, 0, 0, B0, B1, B2, A1, A2};  // 濾波器初期化
```

filterNodesXとfilterNodesYの配列は各々、以前の入力試料と以前の出力値を保持するFIFO緩衝部として使われます。filterCofficientsBとfilterCofficientsAの配列は各々、濾波器の順送係数と帰還係数に使われます。一旦濾波器が初期化されると、濾波器関数を呼び出すことができます。濾波器関数は次のように定義されます。

```
int IIR2(struct IIR_filter *myFilter, int newSample);
```

Zﾚｼﾞｽﾀがﾃﾞｰﾀの間接ｱﾄﾞﾚｽ指定に使えるため、換言すると、ﾎﾟｲﾝﾀ操作のため、最初に濾波器構造体へのﾎﾟｲﾝﾀがZﾚｼﾞｽﾀに複写されます。そして算法の中核は走行する準備が整います。試料(ﾃﾞｰﾀ)が取得され、相手となる係数で乗算されます。積が24ﾋﾞｯﾄ幅の累積器に加算されます。全てのﾃﾞｰﾀと係数が乗算と累積(MAC)されると、濾波器節点(ﾉｰﾄﾞ)FIFO緩衝部が更新されます。最後に結果が縮尺されて返されます。FIFO緩衝部の要素y[n-1]を更新する前に累積器の結果が縮尺されることに注意してください。ﾃﾞｰﾀの流れ図は図4-4.で示されます。

図4-4. 一般的なIIR濾波器算法

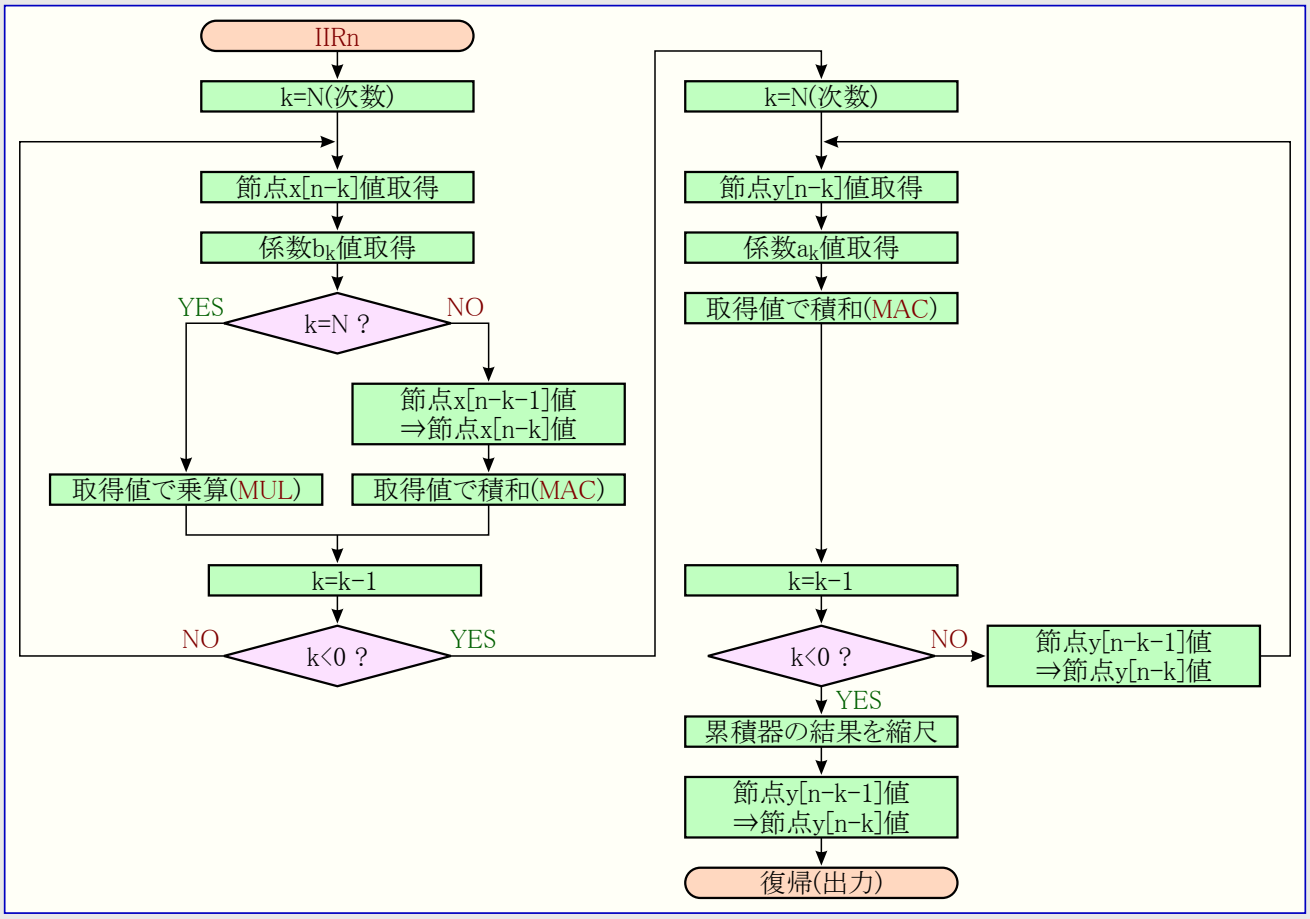


図4-4.の流れ図は繰り返しを使って実装された場合としての算法を示し、これは流れ図の読み易さを増すために行われています。算法は一直線状のｺｰﾄﾞを用いて実装されています。

4.2.1. 濾波算法の性能

表4-4.は2次IIR濾波器の性能を示します。濾波に関する数は濾波算法(MULとMAC操作、節点(ﾉｰﾄﾞ)更新)のｺｰﾄﾞに対してで、付随処理に関する数は(縮尺を含む)濾波器関数の残りに対してです。関数呼び出しと対応する復帰は含まれません。

濾波器の次数変更は累積器容量の変更を必要とするかもしれず、そしてこれは次当たりの濾波の命令/周期数も変えることに注意してください。

表4-4. 2次IIR濾波器の性能

命令数(濾波+付随処理)	実行周期数(濾波+付随処理)	濾波器効率(周期数/次)
86+8	132+8	66

5. 濾波器実装の最適化

この応用記述で実装される濾波器は効率的に作られましたが、違う濾波器への改造が未だ容易です。このためにそれらは部分的な最適化です。以下ではｺｰﾄﾞ量や処理量に関して濾波器を最適化する方法が示唆されます。

5.1. ｺｰﾄﾞ量と処理量の両方の改善

ｺｰﾄﾞ量と処理量の両方を改善する1つの方法は意味のある計算だけが実行されるのを保証することです。この応用記述で実装される濾波器用のｱｾﾝﾌﾞﾘ言語ｺｰﾄﾞはどんな濾波係数の組でも動くように作られています。けれども、例の2次IIR濾波器は2つの0係数を持ちます。0係数での乗算と後続する累積は、それが出力に寄与しないので、勿論、省けます。このようにして、ｺｰﾄﾞ量が減らされ、そして処理量が増やされます。

5.2. ｺｰﾄﾞ量削減

ｺｰﾄﾞ量の削減はﾏｸﾛとしての代わりに関数呼び出しとしてMAC操作を実装することによって達成できます。けれども、これは各関数呼び出しと対応する復帰が追加の周期を消費するため、処理量に影響します。

ｺｰﾄﾞ量を削減する別の方法は、4次FIR濾波器例で実演されたように、より低い次数の濾波器の縦列としての高次濾波器実装です。とは言え、先に言及したように、縦列に於ける各濾波器間での縮尺のため、これは濾波器の精度を減らします。また、0係数でのMAC操作を省くことによる最適化も、このような実装では不可能かもしれません。

5.3. 使用ﾒﾓﾘ削減

両実装例に於いて濾波係数は単純にSRAMへ置かれています。使用ﾒﾓﾘを削減する簡単な方法はﾌﾗｯｼｭ ﾒﾓﾘに係数を置き、そして必要な時に取得することです。濾波器節点(ﾉｰﾄﾞ:以前の入出力試料)があるのと殆ど同じくらい多くの係数があるので、これは濾波器ﾊﾟﾗﾒｰﾀ用のSRAM使用量を潜在的にほぼ半分にするでしょう。

6. 参考文献

[1.] "Discrete-Time signal processing", A.V.Oppenheimer & R.W.Schafer. Prentice Hall InternationalInc. 1989. ISBN 0-13-216771-9

[2.] "Introduction to Signal Processing", S.J.Orfanidis, Prentice Hall International Inc., 1996. ISBN 0-13-240334-X

[3.] FIR filter design, http://www.iowegian.com/scopefir.htm

[4.] FIR filter design, http://www.dsptutor.freeuk.com/FIRFilterDesign/FIRFiltDes102.html

[5.] FIR filter design, http://www.dsptutor.freeuk.com/KaiserFilterDesign/KaiserFilterDesign.html

[6.] IIR filter design, http://www.apogeeddx.com/BQD_Appnote.PDF

[7.] IIR filter design, http://moshier.ne.mediaone.net/ellfdoc.html

[8.] FIR and IIR filter design, http://www-users.cs.york.ac.uk/~fisher/mkfilter/

[9.] FIR and IIR filter design, http://www.nauticom.net/www/jdtaft/papers.htm

本社

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
TEL 1(408) 441-0311
FAX 1(408) 487-2600

国外営業拠点

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
TEL (852) 2245-6100
FAX (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
TEL (33) 1-30-60-70-00
FAX (33) 1-30-60-71-11

Atmel Japan
104-0033 東京都中央区
新川1-24-8
東熱新川ﾋﾞﾙ 9F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製品窓口

ｳｪﾌﾞｻｲﾄ
www.atmel.com

技術支援
avr@atmel.com

販売窓口
www.atmel.com/contacts

文献請求
www.atmel.com/literature

本応用記述はAtmelのAVR223応用記述(doc2527.pdf Rev.2527B-07/08)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には( )内に英語表記や略称などを残す形で表記しています。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。