MAXQ積和演算ユニット(MAC)を使った信号処理

モジュール構造のMAXQアーキテクチャでは、シングルサイクル実行のMAC(積和演算)ユニットを組み込み、信号処理で必要となる演算を容易に行うことができます。

今まで、マイクロコントローラとディジタル信号プロセッサ(DSP)は、さまざまなマイクロコンピュータの中でも両極に位置するものと考えられてきました。マイクロコントローラは、同時性のないイベントに対して短い待ち時間で応答しなければならない制御アプリケーションに適しており、一方、DSPは、大量の数学的演算が必要とされるアプリケーションに威力を発揮します。マイクロコントローラによって大量の数学的演算を行うことも*可能*ですが、マイクロコントローラALUは、基本的に1演算ずつ、順番に処理するように作られているため、あまりすぐれた利用方法とは言えません。同様に、DSPで制御アプリケーションを作ることも不可能ではないものの、アーキテクチャの問題から、コード的にも時間的にも非効率的な動作にならざるを得ません。

制御中心だが信号処理も若干必要といったアプリケーションでは、どのDSPや従来型マイクロコントローラを使用すべきなのかという選択が難しくなります。そのようなケースでは、つい、マイクロコントローラにDSPコードを実行させてしまいたくなります。しかしこの方法では、実行時間の大半をDSP機能に費やしてしまい、制御に問題が生じてしまう結果になりがちです。

このような問題も、MAXQアーキテクチャのような最新のプロセッサアーキテクチャなら解決可能です。MAXQアーキテクチャはモジュール構造となっており、MAC(積和演算ユニット)を追加し、アーキテクチャに統合することは簡単です。ハードウェアMACを追加すれば、制御プロセッサ上で走るアプリケーションに悪影響を与えることなく、16x16の積和演算を1サイクルで実行することができます。このアーティクルでは、MAXQマイクロコントローラのMACモジュールを使い、このような問題を解決する方法を紹介します。

MAXQとMACモジュールの組みあわせ方

DSPが必要とされるケースの1つに、アナログ信号のフィルタリングがあります。信号調節を完了したアナログ信号がADCに入力され、サンプリングしたデータのストリームをディジタルドメインでフィルタリングするというアプリケーションです。フィルタは、一般に次式で表されます。

$$y[n] = \sum b_i x[n-i] + \sum a_i y[n-i]$$

ただし、b_iとa_iは、それぞれ、システムのフィードフォワード応答とフィードバック応答を特性づけます。

a_iとb_iの値によって、ディジタルフィルタは大きく2つに分類することができます。1つはFIR(有限インパルス応答)フィルタで、もう1つはIIR(無限インパルス応答)フィルタです。システムにフィードバック要素が存在しないフィルタ(すべての$a_i = 0$)を、FIR型といいます。

$$y[n] = \sum b_i x[n-i]$$

これに対し、a_iとb_iの両者ともゼロでないフィルタを、IIR型といいます。

前述のFIRフィルタの式を見ればわかるように、数値演算の中心は、入力サンプルに定数をかけ、その結果をn個のサンプルについて合算するという処理です。この処理をC言語で書くと、以下のようになります。

```
y[n]=0;
for(i=0; i<n; i++)
        y[n] += x[i] * b[i];
```

乗算器ユニットを持つマイクロプロセッサでは、同じことを以下の疑似アセンブラコードによって実行することができます。

```
move ptr0, #x       ;Primary data pointer -> samples
move ptr1, #b       ;Secondary DP -> coefficients
move ctr, #n        ;Loop counter gets number of samples
move result, #0     ;Clear result register
```

```
ACC_LOOP:
        move acc, @ptr0    ;Get a sample
        mul   @ptr1        ;Multiply by coefficient
        add   result       ;Add to previous result
        move  result, acc  ;...and save the result back
        inc   ptr0         ;Point to next sample
        inc   ptr1         ;Point to next coefficient
        dec   ctr          ;Decrement loop counter
        jump  nz, ACC_LOOP ;Jump if there are more samples
        end
```

このように、乗算器があっても、積和ループは12の命令と(シングルサイクル実行のユニットと乗算器だったと仮定して) 4 + 8nサイクルが必要となります。

これに対し、MAXQ乗算器は真の積和演算ユニットです。前述の操作をMAXQアーキテクチャで行うと、コード空間は12ワードから9ワードに、実行時間は4 + 5nサイクルに短縮されます。

```
        move DP[0], #x        ; DP[0] -> x[0]
        move DP[1], #b        ; DP[1] -> b[0]
        move LC[0], #loop_cnt ; LC[0] -> number of samples
        move  MCNT, #INIT_MAC ; Initialize MAC unit
MAC_LOOP:
        move DP[0], DP[0]     ; Activate DP[0]
        move MA, @DP[0]++     ; Get sample into MAC
        move DP[1], DP[1]     ; Activate DP[1]
        move MB, @DP[1]++     ; Get coeff into MAC and multiply
        djnz  LC[0], MAC_LOOP
```

MAXQ積和演算ユニットでは、2番目のオペランドがユニットにロードされると、必要な演算が自動的に行われます。結果は、MCレジスタに記録されます。このMCレジスタは40ビット長であり、オーバーフローなしで多くの32ビットデータの加算を行うことができます。従来は原子操作を行うごとにオーバーフローの確認が必要だったため、この部分も改善された点だといえます。信号処理フローでMACを効率的に利用する方法を示すため、シンプルなアプリケーションとして、DTMFトランシーバの例を示します。

DTMFの概要

DTMFとは、電話網において、端末(電話などのデバイス)から交換機まで、アドレス情報を伝える信号手法です。4種類のディスクリートトーンを2セット使用します。ディスクリートトーンは、「低グループ」(1kHz以下)と「高グループ」(1kHz以上)で調和的な関係にあります。電話のキーパッドの数字は、低グループのトーン1つと高グループのトーン1つによって表されます。トーンと数字の関係を、**図1**に示します。

DTMFトーンエンコーダ

DTMFトランシーバのエンコーダ部分は、比較的簡単です。ディジタル式のサイン波オシレータを2個用意し、一方を低グループの周波数のいずれか、もう一方を高グループの周波数のいずれかに合わせます。

ディジタル的にサイン波を合成する方法は、いろいろと考えられます。ディジタル合成ではなくサイン波を生成する方法もあります。ポートピンから生成される方形波を強くフィルタリングしてしまうのです。この方法はさまざまなアプリケーションで使われていますが、Bellcore仕様では、この方法で実現されるものよりもスペクトル純度の高いサイン波が要求されています。

電話網におけるDTMF信号技術によって、端末(電話などのデバイス)から交換機にアドレス情報が伝えられます。

...MAXQ積和演算ユニットでは、2番目のオペランドがユニットにロードされると、必要な演算が自動的に行われます。

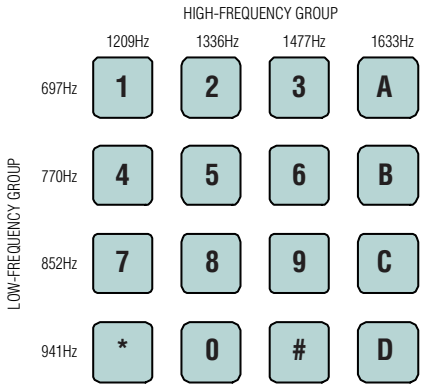

図1. 高周波数グループと低周波数グループから1つずつの周波数を取り出して組み合わせると、DTMF信号が生成されます。

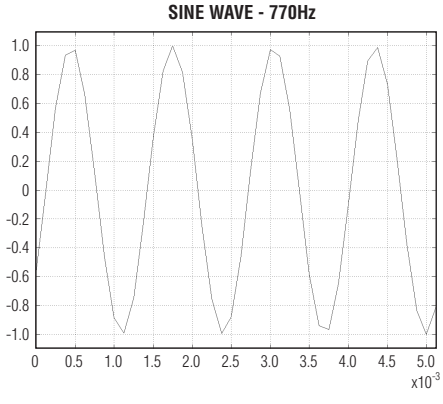

図2. 再帰共振器によるサイン波の生成。

サイン波を生成する別の方法として、テーブルルックアップ法があります。この方法では、サイン波1/4分のデータをROMテーブルに記録しておき、このテーブルから予め計算した間隔でサンプリングすることによって、必要な波形を生成します。しかし、仕様で定められたスペクトル純度となるよう、高分解能のデータをサイン波1/4分も用意しようとすると、膨大な量の記憶領域が必要になります。実は、もっといい方法があるのです。

再帰型のディジタル共振器[1]を使うと、サイン波が生成することができます(**図2**)。この共振器は、次の差分式で表される2ポールフィルタとして実現します。

$$X_n = k * X_{n-1} - X_{n-2}$$

ただし、*k*は次式で表される定数です。

$$k = 2\cos(2\pi * \text{toneFrequency} / \text{samplingRate})$$

DTMFダイアラで使用されるトーンの数は少ないため、必要となるkの値(8個)を予め計算し、ROMに記録しておきます。たとえば、列1のトーン(770Hz)を8kHzのサンプリングレートで生成するために必要な定数は、次式のように求められます。

$$k = 2\cos(2\pi * 770 / 8000) = 2\cos(0.60) = 1.65$$

計算しておくべき値がもう1つあります。オシレータの起動時に必要となる初期値です。X_{n-1}とX_{n-2}の両方がゼロであれば、その後のX_nはすべてゼロとなります。オシレータを起動する際、X_{n-1}はゼロとし、X_{n-2}を次式により設定します。

$$X_{n-2} = -A * \sin(2\pi * \text{toneFrequency} / \text{samplingRate})$$

今回の例では、大きさ1のサイン波が望ましいとして、実際の値は次のようになります。

$$X_{n-2} = -1 * \sin(2\pi * 770 / 8000) = -\sin(0.60) = -0.57$$

プログラミングは簡単です。まず、中間変数2つ(X1, X2)を初期化します。X1はゼロとし、X2には初期励起値(上記で算出した値)をロードし、発振の準備を行います。以下の演算を行うと、サイン波の次のサンプル値が得られます。

$$X0 = k * X1 - X2$$
$$X2 = X1$$
$$X1 = X0$$

乗算1回、減算1回を行うたびに、サイン波の次のサンプル値が得られます。MAXQマイクロコントローラのシングルサイクルハードウェアMACを使用する場合、次のコードでサイン波を生成することができます。

MACユニットを持つMAXQマイクロコントローラは、従来のマイクロコントローラとディジタル信号プロセッサの間に存在するギャップを埋めることができます。

```
move DP[0], #X1            ; DP[0] -> X1
move MCNT, #INIT_MAC       ; Initialize MAC unit
move MA, #k                ; MA = k
move MB, @DP[0]++          ; MB = X1, MC=k*X1, point to X2
move MA, #-1               ; MA = -1
move MB, @DP[0]--          ; MB = X2, MC=k*X1-X2, point to X1
nop                        ; wait for result
move @--DP[0], MC          ; Store result at X0
```

DTMFトーン検出

検出しなければならない周波数が少ないため、修正Goertzelアルゴリズム[2]を使用します。このアルゴリズムは一般的なDFT方式よりも効率がよく、インバンド信号を高い信頼性で検出することができます。このアルゴリズムは、**図3**に示すシンプルな2次フィルタの形で実現されます。

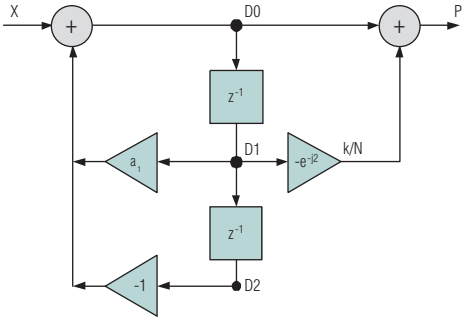

図3. Goertzelアルゴリズムは、2次フィルタとして構成されます。

ある周波数をGoertzelアルゴリズムによって検出するために、定数値を予め計算しておく必要があります。DTMFデコーダの場合、コンパイル時に算出しておくことが可能です。必要なトーン周波数がすべて正確に定義されているからです。定数は、次式により算出します。

$$k = \text{toneFrequency} / \text{samplingRate}$$
$$a_1 = 2\cos(2\pi k)$$

まず、中間変数3つ(D0、D1、D2)をゼロに初期化します。サンプルXを受け取ったら、次の演算を行います。

$$D0 = X + a_1 * D1 - D2$$
$$D2 = D1$$
$$D1 = D0$$

十分な数のサンプルを受信したら(サンプリングレートが8kHzのとき、通常は205個)、最後に算出されたD1とD2を用いて次の計算を行います。

$$P = D1^2 + D2^2 - a_1 * D1 * D2$$

算出されたPは、入力信号に含まれるテスト周波数の2乗を表します。DTMFが持つ4列すべてをデコードするためには、各サンプルに対し8つのフィルタを適用します。各フィルタは固有のk値と固有の中間変数セットを持ちます。変数はすべて16ビットであるため、アルゴリズム全体としては、48バイトの中間記憶容量が必要となります。

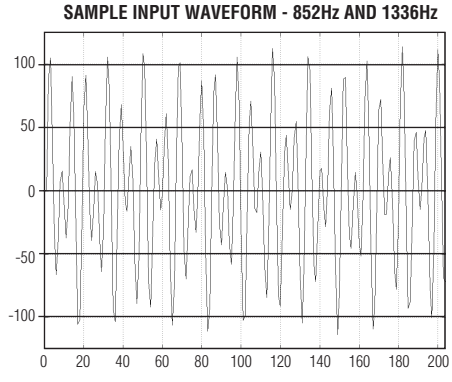

図4. DTMFデコーダのサンプル入力波形です。

各トーン周波数のP値の算出が完了したら、高グループと低グループに1つずつ、他のトーンよりも値が大幅に大きいトーンがあるはずです。違いは、他の2倍以上で、多くの場合10倍以上にもなります。**図4**にデコーダへのサンプル入力信号の例を、**図5**にはGoertzelアルゴリズムの結果の例を示します。出力スペクトラムがこのようにならない場合は、DTMFトーンが存在しないか、ノイズが大きすぎて信号がブロックされているかのいずれかを意味します。

MAXIMのウェブサイトに、Goertzelアルゴリズムのデモンストレーションを行うスプレッドシートと、MACを持つMAXQプロセッサのサンプルコードが用意してあります。japan.maxim-ic.com/MAXQ_DTMFをご覧ください。

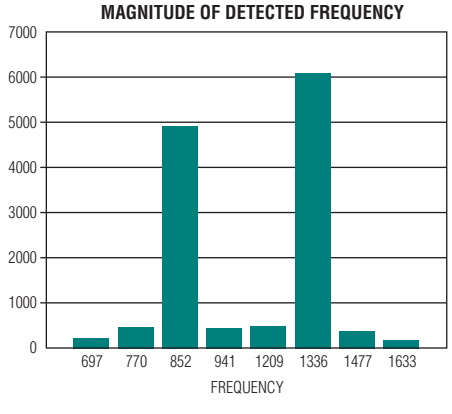

図5. 今回のDTMFデコーダによって、さまざまな周波数の検出が行えます。

まとめ

MACユニットを持つMAXQマイクロコントローラは、従来のマイクロコントローラとディジタル信号プロセッサの間に存在するギャップを埋めることができます。MAXQマイクロコントローラにハードウェアMACを追加すると、従来16ビットのマイクロコントローラでは不可能だったレベルの信号処理が可能になります。シングルサイクルMACによって現実世界のアプリケーションで必要とされることの多い機能を実現すれば、リアルタイムな信号処理ができます。

[1] Todd Hodes, John Hauser, Adrian Freed, John Wawrzynek, and David Wessel. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-99, March 15~19, 1999), pp. 993~996.
[2] Alan Oppenheim and Ronald Schafer, Discrete-Time Signal Processing. Prentice Hall.