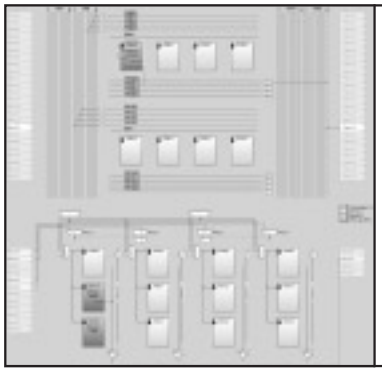
連　載



PSoC マイコン活用講座

積和演算器を使ったディジタル信号処理アプリケーション

第5回　電子オルゴールの実験と製作

桑野 雅彦
Masahiko Kuwano

PSoCのCPUコアM8Cには，乗算命令がありません．しかし，PSoCのシステム・リソースには，1クロックで積和演算を実行することができるMultiply Accumulate(MAC：積和演算器)が搭載されています．

今回は，ディジタル信号処理と積和演算の関係，そして，この積和演算器の使いかた，電子オルゴール(**写真5-1**)への応用例を紹介します．

積和演算はディジタル信号処理の基本

● ディジタル信号処理に欠かせない積和演算

ディジタル信号処理では，

$(a_0 \times b_0) + (a_1 \times b_1) + (a_2 \times b_2) + (a_3 \times b_3) + \cdots$

という積和演算がよく登場します．除算は逆数の乗算，減算は負号を付けて加算すると考えれば，

$(a_0 \div b_0) - (a_1 \times b_1) + (a_2 \div b_2) - (a_3 \times b_3) \cdots$

という演算も積和演算と言えるでしょう．

楽器に配置したマイクの音を，スタジオ・ミキサで合成するという操作は，まさしく，

出力＝
(入力1)×(ゲイン1)＋(入力2)×(ゲイン2)＋…

という積和演算を行っています．同様にエコーやリバーブなどは，

新出力＝(旧出力に時間遅れを付加したもの)×(減衰度)＋(入力信号)

というぐあいに，出力を時間遅れさせたものを減衰(乗算)させて，入力信号と加算という演算(積和演算)を繰り返し行っていると見ることができます．

PSoCのSC(Switched Capacitor)ブロックによる積分回路の動作を見ると，入力をある倍率にして次々に

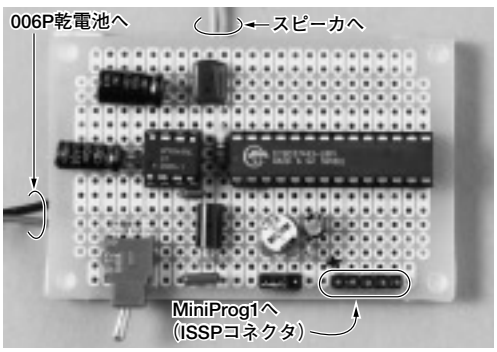


写真5-1　製作したPSoCオルゴールの外観

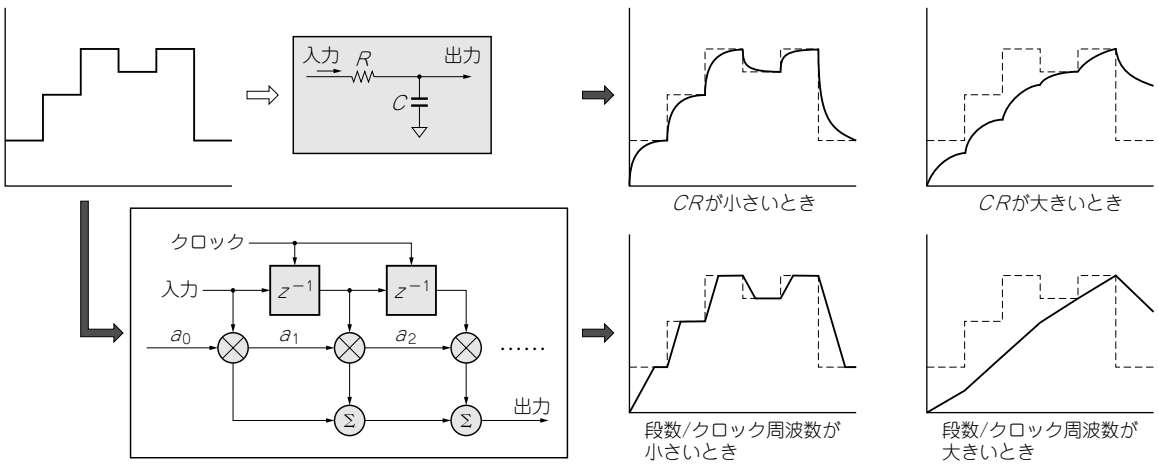


図5-1　アナログ・フィルタとディジタル・フィルタの比較

コンデンサ(Fcap)に充電(加算)していくという動作にほかなりませんし，微分回路の動作を見ても，入力を一定倍率にしたものと1回前に取っておいたデータの差分が出力されると見れば，やはり積和(少し正確に言えば積差だが)演算であり，いわゆるディジタル・フィルタも積和演算の形で表現されています．

● FIRフィルタとIIRフィルタの動作

もう少し具体的な例を見てみましょう．**図5-1**はFIR(Finite Impulse Response：有限長インパルス応答)フィルタと，抵抗とコンデンサで作られたアナログ・フィルタを比較したものです．

z^{-1}は1クロックごとにデータを隣に受け渡すシフトレジスタのようなもので，⊗は乗算を，a_0，a_1，a_2がそれぞれ入力値と掛けられる値を，Σは入力値の加算を示します．

この図では，入力が1クロックごとに変化しているようにも見えますが，変化点の間は10クロックなり，100クロックなりの時間がかかっているものと考えてください．

このFIRフィルタがどのような動作になるのか，試しに1クロックごとに変化するようなデータを与えた状態を追いかけてみましょう．図のように乗算が3段あるとして，a_0，a_1，a_2を仮に0.5，0.3，0.2とします．

いま入力がずっと0の状態から1に変化してそのまま1を継続したとすると，

初期状態

出力 = 0 × 0.5 + 0 × 0.3 + 0 × 0.2 = 0

1クロック目

出力 = 1 × 0.5 + 0 × 0.3 + 0 × 0.2 = 0.5

2クロック目

出力 = 1 × 0.5 + 1 × 0.3 + 0 × 0.2 = 0.8

3クロック目以降

出力 = 1 × 0.5 + 1 × 0.3 + 1 × 0.2 = 1.0

となります．まさに積和演算そのものであることがわかります．

この計算結果を見ると，入力は急速に変化していますが，出力は徐々に変化していくことがわかります．*CR*によるロー・パス・フィルタの動作とよく似ています．これが例に出した**図5-1**の波形です．

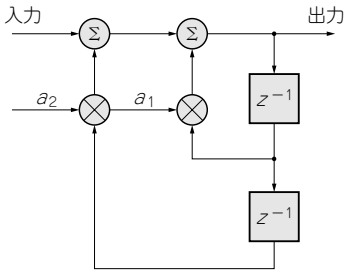


図5-2　IIRフィルタの例

a_0，a_1，a_2が1/3ずつであれば，3回ぶんのデータの平均を取っていることになります．段数を増やしていけば，より多くのデータの平均値になるので，細かい凹凸はなくなっていくと想像できます．

図5-2はIIR(Infinite Impulse Response：無限長インパルス応答)フィルタの一例です．FIRフィルタでは，入力信号が右端まで到達すればそれ以降は捨てられてしまい，出力には影響を与えません．

IIRフィルタの場合には，出力を入力側に戻して加算しているので，エコーと同じように影響自体は次第に小さくなるものの，分解能以下になるまで延々と影響を与え続けることになります．この場合に行われている演算も，結局は2段のディレイ出力それぞれに乗算を行って，入力と加算を行うという積和演算にほかならないということがわかります．

このほか，入力信号を正弦波の和の形で表すフーリエ変換なども含め，信号処理のさまざまな場面で，

$$\Sigma X(n) \cdot a(n)$$

の形で表される式が登場します．信号処理すなわち積和演算と言ってもよいくらいの基本処理なのです．

マイコン，DSPにおける積和演算処理

● 通常のマイコンを使った積和演算とその処理時間

とかく何かをやろうと思うと，すぐに積和演算が必要となってくる信号処理ですが，乗算命令をもたない8ビット・クラスのワンチップ・マイコンでは，整数乗算であっても地道にシフトと加算を組み合わせていくことになるため，負荷はそれなりに大きいものです．

例えば，2進法で`110b`と`101b`という3桁の乗算を考えてみます．やりかた自体は通常の10進法での計算方法と同じです．2進法なので1桁ぶんの乗算自体は`1`×`1`が`1`になる以外は全部ゼロとなるだけで「九九」のような桁上がりは考慮する必要はありません．

手順を簡単にするために，筆算とは逆に上の桁から計算してみます．

① 加算結果をゼロ・クリア(筆算では出てこない)
② 乗数(初期値`101`)の最上位ビットをチェック
③ `1`なら，加算結果に被乗数(`110`)を加算
④ 必要な桁数ぶん終わったら終了
⑤ 乗数を左にシフト
⑥ 合計値を左にシフト

この例でいけば，④の段階の値をトレースすると(合計値は6ビット長にする)

1ループ目：乗数：`101`，合計値　`000110`
2ループ目：乗数：`010`，合計値　`001100`
3ループ目：乗数：`100`，合計値　`011110`

となって終了します．この例でもわかるとおり，合計