組み込み技術者のためのオープン・ソースによるDSPアルゴリズム開発手法

第4回　演算関数の扱い

この連載では，ディジタル信号処理プロセッサ（DSP）の上で動作するソフトウェアのアルゴリズムを開発する方法を解説している．今回はDSPに実装するために必要な固定小数点演算について解説する．また，この演算をC言語の関数で実装する方法を紹介する．（編集部）

冴木 元

● DSPの命令をC言語で実装する

今回は，DSPの固定小数点演算をどのようにして実装するかを解説します．読者の皆さんは，DSPのソフトウェアでは乗算を，

```
c=a*b;
```

と記述できないことをご存知でしょうか．その理由は，DSPは固定小数点（演算）による実装が主流だからです．DSPは「サチュレート」などの固定小数点特有の演算処理を命令として含んでいます．しかし，C言語の命令はこれらの演算に1対1に対応しません．従って，C言語の*は固定小数点演算の乗算には用いることができません．

固定小数点DSPによる実装を前提としたリファレンス・コードには，必ずDSPの演算をC言語の関数で実装した関数群が用意されています．これらを本稿では「演算関数」と呼びます[注1]．演算関数は，DSPの命令をC言語の関数で記述したものです．有名なのは，ETSI（European Telecommunications Standards Institute）という団体で用意され，さまざまなCODECの実装に用いられているbasic_opというものです．これは，3GPPのWebサイトなどから入手が可能です．

本稿で例として用いているSpeexの場合，basic_opは演算関数とされていません．その代わりに，fixed_generic.hというヘッダ・ファイルにマクロの形で演算が定義されています．後述するように，Speex独自の演算関数は，一般的に用いられているbasic_opのような考え方とはかなり異なる部分があります．

本稿では，まずbasic_opの代表的な演算関数の実装方法について解説します．次にSpeexの実装はどうなっているのかを簡単に解説し，basic_opと比較してC64x+における実装はどのようにしたらいいのかを考えます．

● 固定小数点実装のための演算関数

演算関数とは，パソコン（汎用コンピュータ）上で信号処理アルゴリズムのシミュレーションを行うときに，DSP特有の命令を実装するために用いられるC言語の関数です．このようなものが必要となる理由は，大部分のDSPが固定小数点演算を実行する機構を備えているためです．DSPが浮動小数点演算を扱えるのであれば，リファレンス・コードが浮動小数点演算で実装されていても問題ありません．しかし，DSPは固定小数点演算による実装が一般的です．従って，リファレンス・コードは浮動小数点版と固定小数点版の2種類があることになります．Speexには浮動小数点版と固定小数点版が共に収められています（**図1**）．

サチュレートや正規化といった固定小数点特有の演算処理は，通常パソコンなどで用いられる汎用CPUに組み込まれているとは限りません．従って，固定小数点版のリ

注1：演算関数とは，arithmetic operatorの日本語訳である．実際は，開発者によって違う呼び方をされているらしい．

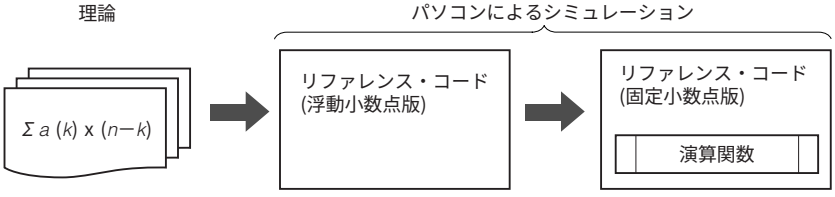

図1
固定小数点によるリファレンス
リファレンス・コードは2種類ある．多くのDSPは固定小数点演算実装のため，浮動小数点版のリファレンス・コードしかないのなら，改造して固定小数点版を作らなければならない．固定小数点版の実装のためには演算関数が必要となる．Speexには浮動小数点版と固定小数点版の両方が用意されている．

KEYWORD ── DSP，Speex，固定小数点演算，論理シフト，算術シフト，正規化，サチュレート，積和算，積差算，シフト

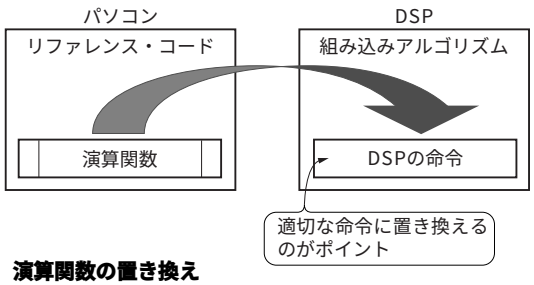

図2　演算関数の置き換え

C言語の関数である演算関数をそのままDSP上でコンパイルしてしまっては，処理量が多くなりすぎる．アルゴリズムを実装するに当たっては，演算関数をどのようなDSPの命令に置き換えたらいいかを考える必要がある．

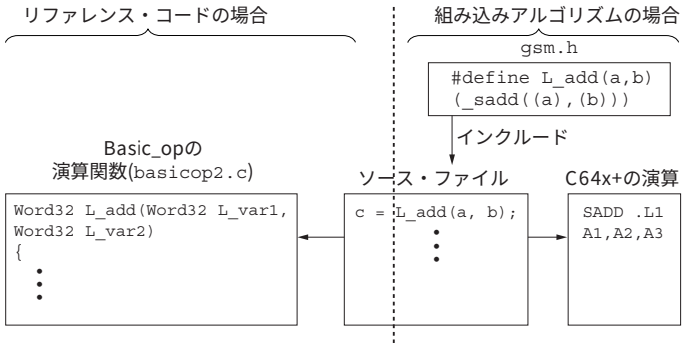

図3　CCS環境におけるbasic_op演算関数の置き換え

CCS環境では，basic_opの演算関数をC64x+の命令に置き換えることは簡単である．basic_opの演算関数とC64x+用のintrinsics関数の対応がgsm.hに用意されている．組み込みアルゴリズム用のソース・コードでは，このヘッダ・ファイルを演算関数の定義ファイル(basicop2.c)の代わりにインクルードすれば，置き換え作業は終わる．

ファレンス・コードでは，これらの演算処理をC言語の関数として実装するようにしています．

● 演算関数の置き換え

DSPでは，固定小数点特有の演算処理がチップの命令として実装されています．適切な命令を使用していれば，余計なサイクル数はかかりません．しかし，C言語で実装されたリファレンス・コードについては，このような命令を汎用CPUが持っていることを期待できません．そこで，固定小数点演算を関数として用意し，固定小数点特有の計算処理に対応しています．リファレンス・コードからDSPの組み込みアルゴリズムを作る際には，この演算関数を適切なDSPの命令に取り替える作業が求められます(**図2**)．演算関数はC言語の関数なので，そのままコンパイルして組み込みモジュールの中で使うと，処理量が増大してしまいます．

basic_opを用いてCCS(Code Composer Studio)で実装する場合，この置き換え作業そのものは簡単です．＜CCS_INSTALL_DIR＞\C6000\cgtools\include\gsm.hというヘッダ・ファイルを見ると，basic_opの演算関数とintrinsics関数，またはそれを用いたマクロとの対応が記されています[注2]．開発者はこのヘッダ・ファイルをインクルードすれば，コンパイラが自動的に演算関数をC64x+用の適切な命令に置き換えてくれます(**図3**)．

● 計算結果を最小値，最大値で止める

DSPの固定小数点演算において基本となる演算方法について解説します．

サチュレート(またはクリップ)とは，オーバフローが発生するときに計算結果を最大値または最小値で止める処理を指します．おそらく，DSPの命令体系を知る上で最も基本となる演算処理でしょう．

16ビット演算の場合，最大値は0x7fff(10進数で32767)，最小値は0x8000(10進数で－32768)です．同じ符号同士の足し算の場合，あるいは違う符号同士の引き算の場合に，この値を超えてしまうことがありえます．例えば，0x4000＋0x4000という演算を考えてみます．電卓で計算すると0x8000になります．しかし，16ビット精度の演算の場合は，表しうる最大の正の値は0x7fffであるため，これをそのまま計算結果にすると符号反転が生じてしまうことになります．なぜなら，0x8000はMSBの符号ビットが1，すなわちマイナスの値となるからです(2進数で表されたデータ列の最上位ビットをMSB，最下位ビットをLSBと呼ぶ)．これでは，正の値の計算結果を求める

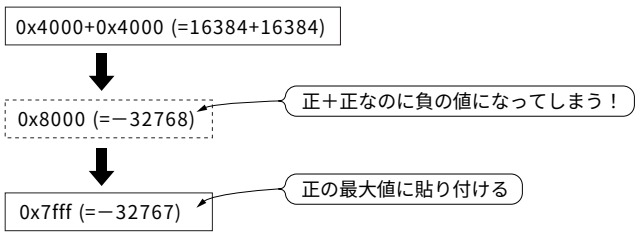

図4　サチュレートとは

例えば0x4000＋0x4000という加算を考えた場合，16ビットの演算器でそのまま実施してしまうと答えが0x8000，つまり負の値になる．このような符号反転を防ぐために正または負の最大値に演算結果を貼り付ける処理がサチュレート．

注2：intrinsics関数とは，コンパイラの組み込み関数のこと．CPUの演算を効率良く使ったアセンブリ・コードをコンパイラに生成させるために用いられる．