Using SigLib With Fast IIR Filter Functions

Numerix Ltd.

December 2004

Numerix Ltd.
7 Dauphine Close, Coalville, Leics, LE67 4QQ, UK
Phone : +44 (0)208 020 0046, Fax : +44 (0)208 020 0047
Internet : http://www.numerix-dsp.com
Email : support@numerix-dsp.com

Introduction

SigLib is an ANSI C source library of DSP functions that is designed to include a large number of DSP functions and to be portable across as many different architectures and operating systems as possible. While this architecture makes the library idea for prototyping and development, many applications require optimum performance.

This applications note looks at the issues related to using SigLib with DSP functions that have been optimised for a particular architecture. In particular we have chosen the TMS320C67x architecture because it is typical of many modern DSPs. The function chosen for this applications note is the cascaded Infinite Impulse Response (IIR) filter. This was chosen because it is an algorithm that commonly requires optimised performance.

For this applications note we follow the usual development path of proving the algorithm using the SigLib functionality and then replace the SigLib IIR filter functions with hand optimised assembly coded versions – in this case, the biquad.asm program from Texas Instruments.

One of the main issues related to using DSP functions from multiple sources is the different APIs involved. In this case, the function parameters are in a different order and the coefficient arrays themselves require a different structure. In a real time DSP system it is critical that any variations to the APIs are accounted for during initialisation so that there are no run-time penalties. In this case we have written a simple C function to convert the coefficients from one format to the other.

Implementation Details

This applications note includes two new functions : `SDS_FastIir ()` and `SDA_FastIir ()`. These utilise the TI “biquad.asm” optimised assembly coded IIR filter function.

Diagram 1 shows the standard IIR biquad filter structure. If the C code for this is compiled then there are some dependencies that preclude the compiler from generating optimised code.

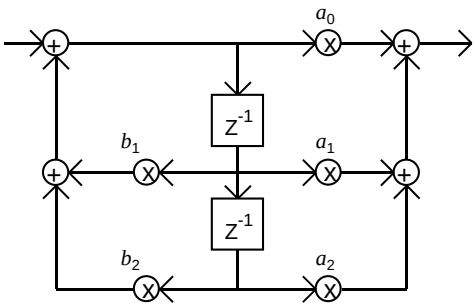

Diagram 1 : The Standard IIR Biquad Filter Structure

The optimised IIR filter structure (shown in diagram 2) utilises a modification of the standard IIR biquad structure, where the feedforward coefficients are scaled so that a0 = 1.0. This results in a structure that can fully utilise the parallelism of the TMS320C67x architecture but a consequence of this is that the output is scaled compared to the standard IIR filter by the value used to scale a0. Being 1.0, the a0 coefficient can be removed from the coefficient array and the new array has the following coefficients for each biquad : A1, A2, B1, B2. To use these functions you must also ensure that the coefficient and state arrays are aligned on a suitable 64 bit memory alignment.

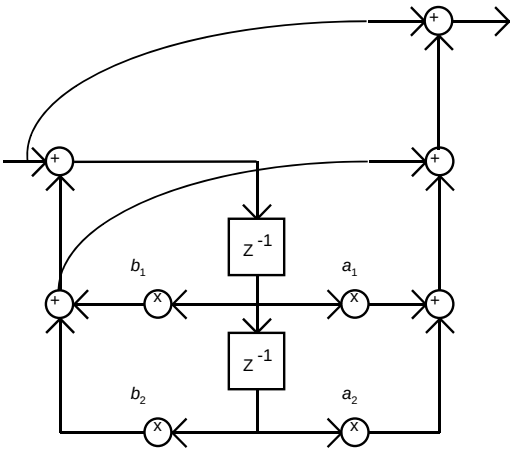

Diagram 2 : The Optimised IIR Biquad Filter Structure

The example code included with this applications not include a function (`generate_ti_iir_coeffs`) that will convert the SigLib format coefficients into a format compatible with the TI optimised function.

The new function (`SDA_FastIir`) can work "in-place" i.e. where the source and destination pointers can point to the same location.

The Source Code

The source code for this applications not is included in the file “FastIIR.c”. This can be used on the TMS320C6x simulator or on real hardware. To compile, link and run the program, you will neen to have the TMS320C6x tool set installed. Then use the following batch file on the command line :

```
> clrfas FastIIR
```

Benchmark Results

The optimised IIR filter function requires (`4*(numBiquad) + 29`) cycles. The benchmarks were obtained with TMS320C6x C/C++ Compiler Version 4.36. The following table provides results of benchmark tests that compare the results of the standard SigLib IIR filter (`SDA_Iir`) function against the optimised function.

Number Of Cascaded Biquads	Sample Length	SigLib Benchmark	Optimised Filter Benchmark	Performance Increase
3	10	810	561	1.44
3	20	1600	1101	1.45
6	10	1410	681	2.07
6	20	2800	1341	2.09

The table shows that the optimised IIR biquad filter function will give a 2x performance increase for 12^{th} order IIR filters.