SCUBA-2

Low Pass Filtering

Revision History:

Rev. 1.0 MA July 28, 2006 Initial Release

Rev. 1.1 MA Sept. 15, 2006 added Document number

Rev. 1.2 MA Apr. 30, 2007 added new scaling factor k3, explained sampling rate.

Rev. 1.3 MA Jul. 4, 2008 explained fs of 12195 and corrected k3.

Created on: 6 February 2006

Last saved on: 04 July 2008

Table of Contents

1. Introduction

This document describes the type of audio filter that is desired for MCE operation and reviews some of the challenges involved in the implementation of the filter.

1.1 Filter Requirements

Filter response type: Low Pass Filter

Sampling rate, f_s =12195Hz [(1)]

Cut-off frequency: 100Hz

Filter Order: 4

1.2 Filter Specifications

Design Method: 4-pole IIR – Butterworth

Realization type: Direct form II – Series biquads

Filter coefficients are generated using FDAtool in Matlab.

Filter Coefficients: SOS: (Quantized filter to second-order sections)

Section 1:

Numerator: 1 2 1

Denominator: 1 -1.9587428340882587 0.96134553442399129

Gain = $1/k_1$ = 0.00065067508393319923 (not implemented)

Section 2:

Numerator: 1 2 1

Denominator: 1 -1.9066292518523014 0.90916270571237567

Gain = $1/k_2$= 0.00063336346501859835 (not implemented)

Coefficients width: 15b (implemented as signed binary fractional SBF 1.14)

Filter Input:

input width: 18b

scaling factor for the input to the filter chain: 2^{-11}

(i.e. first-stage feedback calculation results are scaled down before being fed to the filter)

$1/k_3$ = scaling factor between 2 biquads

Internal arithmetic:

Delay width: 29b (Wn width)

Filter Output:

Output width: 32b

Output gain: 1216= (simulation)

(calculated as $(k_1*k_2)/k_3$=1184 – non quantized arithmetic)

The gain difference of 1184 and 1216 can be attributed to the coefficient quantization effects.

Note (1): corresponding to 50MHz/100*41 where row_len = 100, num_rows=41, clk=50MHz

2. Digital Filter Design - Background

In this section, some background information is provided to clarify why the particular type of filter is chosen.

2.1 IIR Filters

One type of digital filter is the Infinite Impulse Response (IIR) filter, which is not as well supported and is generally used in the lower sample rates (i.e., < 200kHz). The IIR uses feedback in order to compute outputs, and it is known as a recursive filter.

Advantages of the IIR Filter:

1. Better magnitude response
2. Fewer coefficients
3. Less storage is required for storing variables
4. A lower delay
5. It is closer to analog models

A number of different classical IIR filters are available.

Butterworth
The Butterworth filter provides the best approximation to the ideal lowpass filter response at analog frequencies. Passband and stopband response is maximally flat.

Bessel
Analog Bessel lowpass filters have maximally flat group delay at zero frequency and retain nearly constant group delay across the entire passband. Filtered signals therefore maintain their waveshapes in the passband frequency range. Frequency mapped and digital Bessel filters, however, do not have this maximally flat property. Bessel filters generally require a higher filter order than other filters for satisfactory stopband attenuation.

2.1.1 IIR Filter Expressions

IIR Filters are recursive: the output is fed back to make a contribution. The expression for the IIR is shown below; note that a delayed version of the y(n) output plays a part in the output:

$$y(n) = \sum_{i=0}^{n} b(i)x(n-i) \; - \sum_{i=0}^{m} a(i)y(m-i)$$

a(i) and b(i) are the coefficients of the IIR filter. Another way to express a IIR Filter is as a transfer function with numerator coefficients “bi” and denominator coefficients “ai”:

$$H(z) = \frac{\sum\limits_{i=0}^{n} b_i z^{-i}}{1 + \sum\limits_{i=0}^{m} a_i z^{-i}}$$

2.2 IIR Filter Structures

Direct Form II

The Direct Form I architecture description noted that the forward and reverse FIR filter stages can be swapped, which creates a centre consisting of two columns of delay elements. From this, one column can be formed; hence, this type of structure is known as "canonical", meaning it requires the minimum amount of storage.

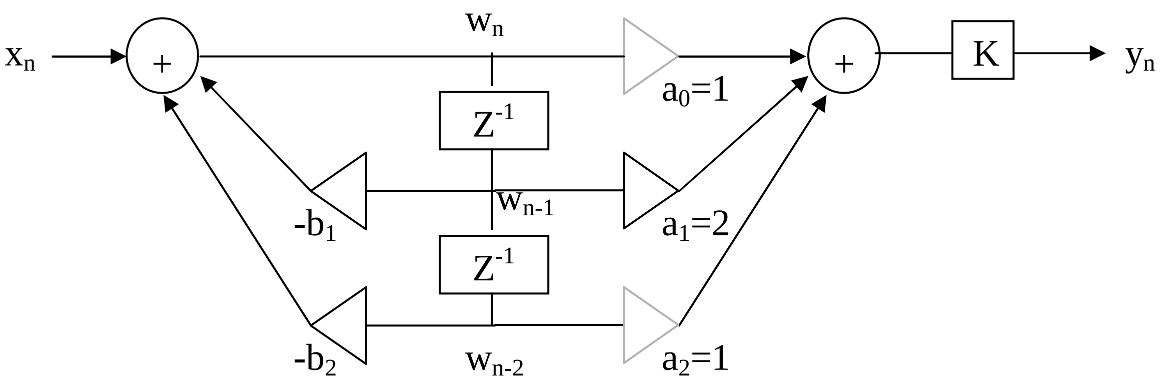


Figure 1: Direct Form II representation of a biquad

Biquad
The Biquad filter structure is that of a Direct Form-II, but it includes a second-order numerator and denominator coefficient (i.e., it is simply two poles and two zeros). This structure is used in FPGA/DSP implementations, because it is not terribly sensitive to quantization effects.

Butterworth Biquad:

The butterworth biquad expression is:

$$H(z) = \frac{1 + 2*z^{-1} + z^{-2}}{1 + b_1*z^{-1} + b_2*z^{-2}}$$

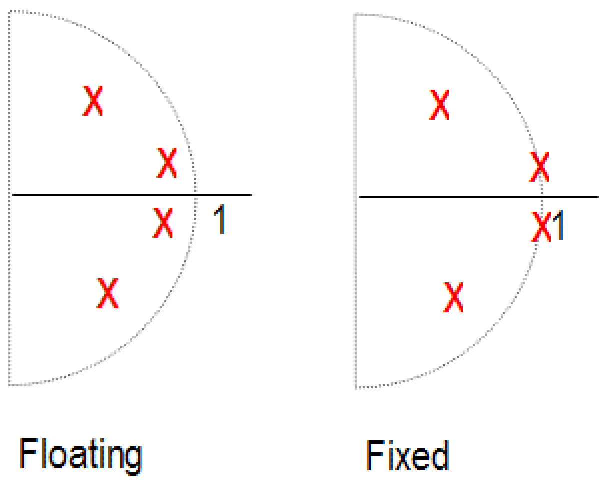
The coefficients in the nominator have the charm that simplify the calculations such that no multiplier is needed and the entire nominator can be calculated using shift and accumulators. Multiplications are expensive operations in FPGA/DSP implementations.

2.3 Fixed Point Implementations

Several issues must be examined in detail to ensure satisfactory fixed-point operation of the IIR filter:

1) Coefficient/Internal Quantization
2) Wraparound/Saturation
3) Scaling

Coefficient/Internal Quantization In order to examine the effect of quantization, it is useful to look at the pole/zero plot. This shows how the zeros (depths in the frequency response plot) and poles (peaks in the frequency response plot) are positioned. In fact, an issue with IIR stability relates to the denominator coefficients and their positions, as poles, on the pole/zero plot:



The poles for the floating-point version of the plot are shown on the left; they are within the unit circle (i.e., the values of the coefficients are less than 1). Once the coefficients are quantized, these poles move, which affects the frequency response. If they move onto the unit circle (i.e., the poles equal “1”), you potentially have an oscillator; If the poles become greater than 1, the filter becomes unstable.

Wraparound/Saturation
A fixed-point implementation has a certain bit width, and hence has a range. Calculations may cause the filter to exceed its maximum/minimum ranges. For example, let’s consider a 2’s complement value of ‘ 01111000’(+120) + ‘00001001’(+9) = ‘10000001’ =(-127). The large positive number becomes a large negative number; this is known as “wraparound”, and it can cause large errors.

Scaling
There are two methods of dealing with overflows:
1. If scaling is used, values can never overflow. DSP processors tend to use different kinds of scaling in order to fit within their fixed structure.
2. Use saturation logic. In our example, the results would be ‘01111111’(+127).

2.4 The Artifacts of IIR filters

The main artifacts of IIR filtering are the quantization noise and the limit cycle oscillations.

The truncation or rounding of the IIR accumulator at the output of filter creates quantization noise. This noise is fed into the filter recursive path. The noise is multiplied by the IIR recursive path transfer function. The impact of this noise source is very significant in the low cutoff frequency filters of the second order, since the recursive path gain is proportional to the second power of Fc/Fs ratio. The filter stages with high Q can also suffer from this effect because the gain is proportional to Q.

The best way to reduce the quantization noise is improve the arithmetic accuracy. For a multi-stage filter, the noise contributions of the stages can add.

The other way to reduce the quantization noise is the noise shaping. Noise shaping is feeding the accumulator truncation error back into the filter. That allows for better SNR at low frequencies for the cost of an increased noise at the high frequencies. The noise shaping with higher order error feedback can significantly improve the SNR, however the added complexity and limited performance makes it less attractive, then the increased precision arithmetic.

The limit cycles are the low amplitude oscillations which may occur in IIR filters. The reason for those oscillations is a finite precision of the arithmetic operations. The limit cycle existence depends on the particular filter coefficients and the input data. The filters with high Q, low Fc/Fs ratio and the rounding of the accumulator at the output rather then with truncation have a higher probability of a limit cycle behavior.

Usually the amplitude of limit cycle oscillations does not exceed several LSBs. The methods to avoid the limit cycles are the following:

- Improve the precision of filter arithmetic

- Implement a center clipping nonlinear function
- Dithering (adding a random noise with the amplitude of several LSBs)
- Gating: blocking the filter if the energy of the signal is below a certain limit

3. Implementation

3.1 Block Overview

The filter is implemented as part of the fsfb_calc block in order to share FPGA resources, particularly multipliers. Review fsfb_calc.doc first.

3.2 Block Functionality

3.3 Block Data Flow

3.4 Block Location and Block Interface within System

3.4.1 First Stage Feedback Filter Queue

This 64 x 32b RAM block stores the filtered output calculation results. The width of this queue is the same as the wishbone data. It is important to note that the filter results are not double buffered, since delay is acceptable in reading filter results. When a wishbone read request comes in, the read starts at the beginning of the next frame, in order to be aligned with the frame boundaries.

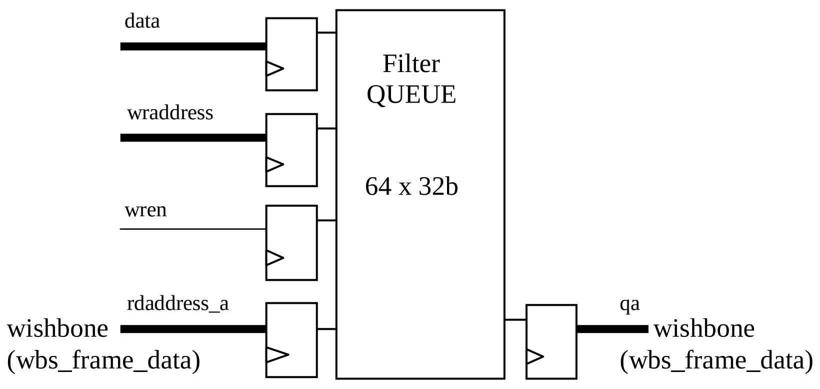


Figure 1: First Stage Feedback Filter Queue

3.4.2 First-Stage Feedback Filter Registers

The fsfb_filter_regs block instantiates 2 RAM blocks to store the previous 2 samples of wn, where wn is the interim filter calculation results. For details of the filter calculations, refer to the fsfb_calculations.doc where the implementation of the second-order Butterworth low-pass filter that is implemented.

The calculations are:

$w_{temp} = b_1 * w_{n-1} + b_2 * w_{n-2}$
$w_n \quad = x_n – w_{temp}/2^m$
$y_n \quad = w_n + 2 * w_{n-1} + w_{n-2}$

where x is the input to the filter and y is the output of the filter, b1 and b2 are the filter coefficients, m is the number of bits for the filter coefficients.
Note that the filter is reset through initialize_window_i signal. Each RAM block has 64 words and the word length is determined in the pack file by FLTR_DLY_WIDTH.

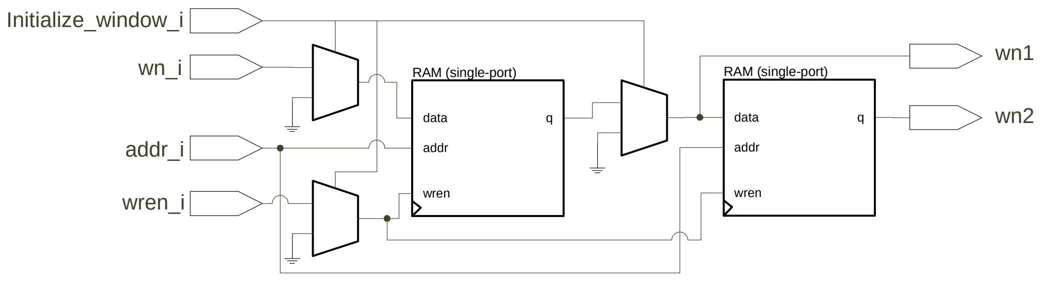


Figure 2: Filter Registers Storage

4. References:

http://www.xilinx.com/xlnx/xweb/xil_tx_display.jsp?sGlobalNavPick=&sSecondaryNavPick=&category=&iLanguageID=1&multPartNum=1&sTechX_ID=mf_iir

http://www.web-ee.com/primers/files/Comm_filters.pdf

http://www-users.cs.york.ac.uk/~fisher/mkfilter/mzt.html

http://www.abvolt.com/research/publications2.htm#top

"Digital Filters: Basics and Design"
Dietrich Schlichtharle
Springer Verlag
ISBN 3-540-66841-1

fsfb_calc.doc

fsfb_calculations.doc