DESC9115 2013 WRITTEN REVIEW 2

DIGITAL AUDIO SYSTEMS
Instructor: William L. Martens
Tutor: Luis A. Miranda J.

Due Date: Friday 6th June, 2013

David Cuthbertson
SID 430070468

David Cuthbertson
SID 430070468

Abstract:

This report delves more deeply into the implementation of FIR filters for audio frequency band splitting in a digital multiband compressor. Specifically the implementation of FFT convolution and Block Convolution as a means of increasing computational efficiency and minimizing input-output delay is explored. It is directly related to the how the DSP technique could be applied to the multiband compressor product developed in Lab Report 2, furthering the development of the product towards 'real time' implementation in the future.

Introduction:

In the realm of DSP filters for digital audio, a there can be a high reliance on 'ideal' filters to give very powerful desired output responses. Such filters can be found in the multiband compressor product development for Lab Report 2, which implements 2000point FIR filters in its "linear phase" band splitting stage. Direct convolution of filters such as these can result in inappropriately high computational time for a given audio task. DSP developers have tackled this issue in the past decades, trading off computational efficiency of filters against their performance, often finding a balance for the given task.

The example given in this review is a digital multiband compressor intended for further development as 'real time' processor. In this situation it is desirable to have the linear phase and steep roll-off characteristics of FIR filters during the band splitting process. A resulting FIR filter can cause unacceptably large computational load if implemented using direct convolution (Smith 1997, p.310). Without massive processing power, the filter cannot be realized in 'real-time' with the current direct convolution method discussed in Lab Report 2. This review seeks to find a possible solution to this problem.

The ideal filter response:

A good band splitting section of a multiband compressor has a few important characteristics. A steep roll off and a high amount of stop band attenuation to give good separation between frequency bands, is desired. It needs to have a 'flat' pass band amplitude response, and it ideally would exhibit a linear phase response.

In Lab Report 2, it was shown that it is easily possible to design linear phase low pass, high pass and band pass FIR filters. By using filters of the same length, it is also possible to have a crossover filter design with a linear combined output phase. (Azizi, Hetzel and Schöpp 1995, p. 2). Further to this, FIR filters with a 'flat' combined amplitude response are achievable. An excerpt from Lab report 2 is shown in Fig 1.

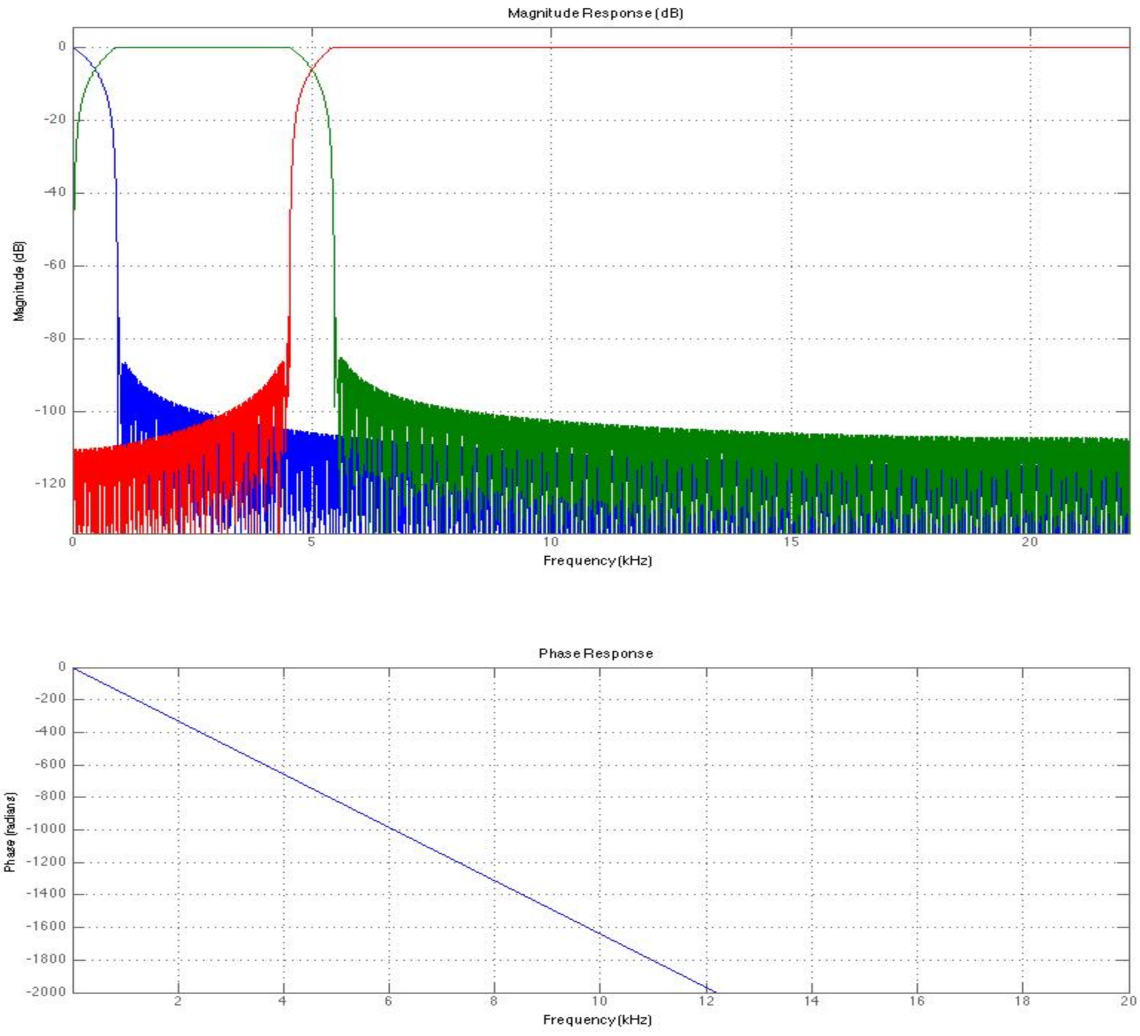


Fig 1 a) Amplitude and b) Phase response of ideal FIR filter for band splitter.

As can be seen, the FIR filter offers an ideal response in terms of amplitude and phase. However the computational speed of this filter using direct convolution is rather slow. The length of the filters in the current product design is 2000 points.

FFT Convolution using the Overlap-and-Add method:

A possible solution to our high processing demand can be found by investigating filtering in the frequency domain or FFT convolution, which is based on the fundamental DSP operation “overlap-and-add”.

The “overlap-and-add” method of segmenting or windowing a signal has been used for many years in many different applications. It involves splitting a signal into small parts and processing them separately before add them back together to create the output signal. For filtering applications, “overlap-and-add” can equate to a large increase in computational efficiency when convolving a large signal with a large filter kernel (Smith 1997, p. 318). For use in ‘real time’ processing, the input signal could theoretically be very large, and so the overlap-and-add method is a good starting point to base our processing. The segmentation of the input signal in overlap-and-add processing can be described by the following formula.

$$x_k[n] = x(n + kL)$$

Where $\mathbf{x_k}$ is the segment, **L** is the length of the segment and **n** is the input sample.

The segmented signal is then "zero padded". Assuming the length of the filter = M-1, we add M-1 zeros to the right of the segment. After the segments are processed, they are added in succession to overlap the last M-1 samples of each segment, producing a complete linear convolution. The M-1 zeros at the end of each segment are added to the first M-1 samples of the next segment and effectively disappear. (Jones 2004 p.4)

FFT Convolution or 'Fast Convolution':

The process of FFT convolution is based around the principle that convolution in the time domain corresponds to multiplication in the frequency domain (Smith, p. 312).

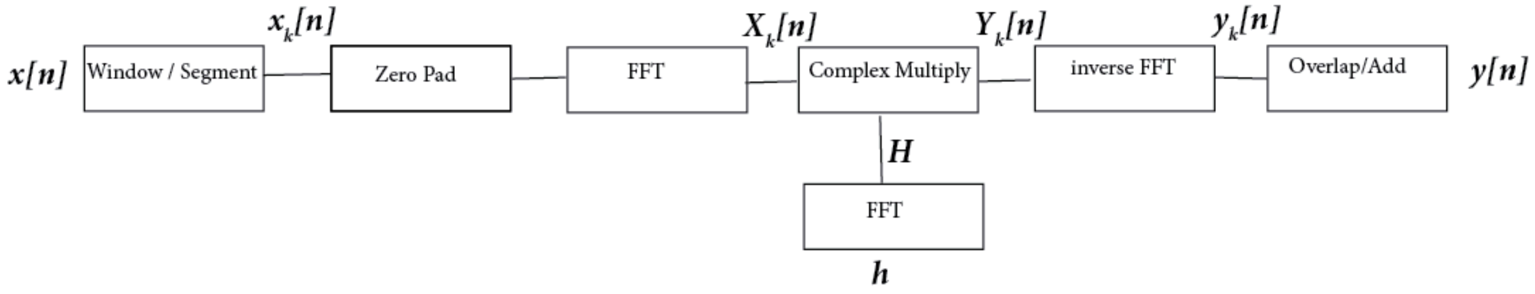
$$y[n] = x(n) * h(n)$$

is equal to

$$Y[n] = X(n) \cdot H(n)$$

Where **y** is output **x** is input and **h** is the filter in the time domain. **Y** is the output, **X** is the input and **H** is the filter in the frequency domain.

By taking advantage of this principle and combining it with the "overlap-and-add" method, we can implement complex FIR filter kernels with relatively small segments of input data to gain an output using much less processing. Below is a block diagram of the signal flow throughout our filter.



This process can also be described by the equations:

$$y_k[n] = IFFT(FFT(x_k[n]) \cdot FFT(h[n]))$$

and

$$y[n] = \sum_k y_k\,(n - kL)$$

Where $\boldsymbol{x_k}[\boldsymbol{n}]$ is the audio segment and $\boldsymbol{h}[\boldsymbol{n}]$ is the filter kernel.

This "Fast Convolution", while greatly increasing the computational efficiency of the filtering, does little improve the actual input-output delay of the system. This is due to the fact that it must first wait for the segment of L input samples to accumulate before they can be converted into the frequency domain. Thus creating a time delay.

Comparison of computational efficiency:

How much more efficient is FFT convolution than direct convolution? Smith (1997, p. 318) offers the view that standard convolution is faster to compute for filters up to about 32-64 points in length, after which it is faster to implement FFT convolution when using long signals. Jones (2004, p. 1) provides us with a simple set of equations to calculate computational cost:

Direct convolution:	- N^2 *complex multiplies*
	- *N(N-1) adds*
FFT convolution:	- *3FFT's + N multiplies*
	- $N + \frac{3N}{2} log_2 N$ *complex multiplies*
	- $3(N log_2 N)$ *complex adds*

(Jones 2004 p.1)

By evaluating this information and applying it to our specific situation, it can be established that FFT convolution would offer large increases in computational efficiency for our product development. Using a 2000 point filter, the "Fast Convolution" method could take orders of magnitude less computational power than the direct convolution method. The extra programming complexity would be well worth the effort.

A zero-latency / minimum latency breakthrough:

Gardner (1995 p.127), states that it is possible to harness the efficiency of block processing (frequency domain) to implement FIR filters without any input-output delay. The process is similar to FFT convolution in that it does the filtering in the frequency domain, but it avoids the time delay caused by waiting for the input samples to accumulate. This is done by 'blocking' the filter response **h[n]** into segments of its own, then using a combination of direct convolution and block convolution (frequency domain) to compute the output **y[n]**.

To implement this process, the filter response is first segmented into blocks. The minimum block size is N points long, usually 64 points, representing the smallest block size at which block processing is more efficient than direct processing (Gardner 129 p. 318).

The first block h_0 is 2N in length, the next blocks h_1 and h_2 are N in length. h_3 and h_4 are 2N, h_5 and h_6 are 4N and this sequence of length continues until our whole filter response is segmented. For our example of a 2000 point filter we would have h_0 through to h_8 (changing slightly our filter length, L to 2048 points). The diagram below shows how our 2048 point filter would be segmented.

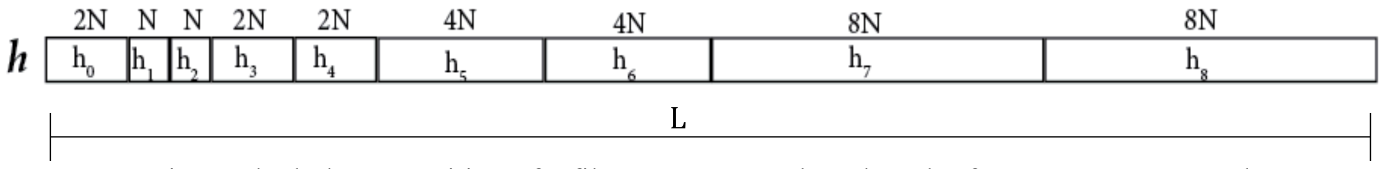


Fig 2. Block decomposition of a filter response. Where length of segment **N = 64** and length of filter **L = 2048**

A minimum-delay is achieved by calculating the first block of outputs using direct convolution in the time domain, $x * h_0$. This provides an immediate output whilst simultaneously providing enough time to accumulate the samples and to process the remaining filter blocks $h_1, h_2 \dots h_8$ in the frequency domain (Gardner 1995 p.129).

The following diagram taken from Gardner's paper, shows how the block processing segments are scheduled in time.

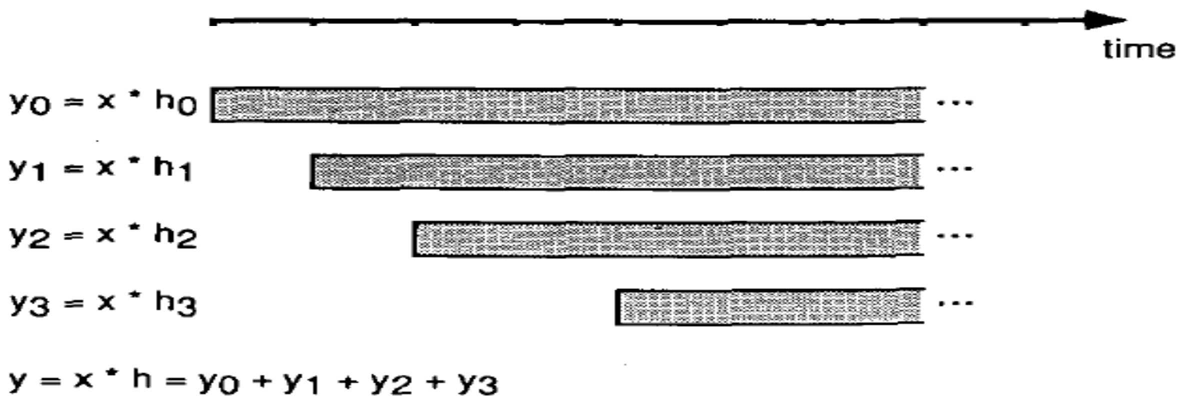


Fig 3. Block processing schedule (Modified from Gardner 1995 p.128)

Each block gets processed in a time sequence, the IFFT of the result is taken and their outputs get added to create the output signal **y[n]**. This shares similarity to the "delay-and-add" process discussed previously.

To comprehend block processing in a more intuitive way, it could be thought of as "an FIR of FFT's" where by the input is convolved with a sequence of small FFT's of the filter – combined with a direct convolution of the very beginning of the filter. In this way, a very long and complex filter can be convolved with a large input signal with minimal computational load, whilst maintaining a very small input-output delay.

Optimization:

Optimization of this processing technique using intuitive design can further improve its speed and efficiency. The most obvious 'tweak' is to pre calculate the spectra of the filter blocks, this is shown by Gardner (1995, p.132).

Gardner offers another intuitive optimization. Within the convolution algorithm discussed, the spectra of the input signal may be reused for every even number of h (besides h_0). The FFT of the input segment need only be calculated for the corresponding odd number of h and then just used again. For example, the input spectra is calculated for h_1 and then reused for h_2, similarly spectra for h_3, reused for h_4. This effectively halves the number of FFT's used for the input signal.

Conclusion:

Through implementation of block convolution and intuitive optimization of the DSP, it has been shown that the 'ideal' FIR filters used for the multiband compressor in Lab Report 2 can be implemented in a much faster and more efficient way. The frequency domain filtering process and optimizations explored in this review offer a strong resolution to the problem outlined in the introduction. This begins to pave the way for further design and possible implementation of the multiband compressor product in real time audio situations.

It must be acknowledged that other aspects of the product development in Lab Report 2 including the compression algorithm require further development before a real time implementation could be achieved. This however, is beyond the scope of this written review.

Work Cited:

S. A. Azizi, H Hetzel, H Schöpp, 1995 'Design and Implementation of Linear Phase Crossover Filters using the FFT', J Audio Eng. Preprint 3991, pp. 2

W. Gardner, 1995, 'Efficient Convolution without Input-Output Delay', J. Audio Eng. Soc., Vol. 43, No. 3, pp. 127-129, 132

D. Jones, 2004, 'Fast Convolution', The Connexions Project, http://cnx.org/content/m12022/latest/

S Smith, 1997, Ch. 18 'FFT Convolution, 'The Scientist and Engineers Guide to Digital Signal Processing', California Technical Publications, California, pp. 312, 318