Real-Time Implementation of Multi-Channel Audio Crosstalk Cancellation Using Mixed Single Frequency Delay Line Filtering Algorithm

Chunduri SreenivasaRao, [1] Dhulipalla VenkataRao[2]
[1]*ECE Department, KL University, India*
[2]*Principal, Narsaraopet Institute of Technology, JNT University, India*

Abstract : *To reproduce high fidelity audio sound and spatial reverberation characteristics, it is desired to use the long filter coefficients in audio surround systems. To implement these long filters, the existing frequency domain based techniques such as overlap save method, multi-rate running convolution suffer from more computational complexity. Apart from computational complexity, algorithm delay is another important factor that needs to be reduced in the real time implementation of these systems. In this paper, mixed single frequency delay line filtering technique was proposed to optimize these factors in multichannel audio crosstalk cancellation and showed analytically how the computations are reduced for different multichannel cases. The ability of proposed method is that it provides less computational complexity even at the impulse response lengths of more than 100msec duration. Unlike in existing methods, algorithm delay depends only on processing frame size instead of filter length so that it provides short processing delay. The proposed technique was implemented on 32-bit floating point DSP processor and efficient design is provided to achieve processor level optimization and less implementation complexity. The computational comparison of this method with conventional methods shows that the proposed technique is very efficient for long filters*

Keywords: *Crosstalk Cancellation, FFT, Frequency Delay Line, Mixed Filtering, Overlap Save method.*

I. Introduction

The objective of 3D audio system is the ability to reproduce high fidelity audio sound at the desired locations by preserving the reverberation characteristics and spatial sound pattern of original audio signal in the reproduced sound. This technology has many spatial audio applications such as home theatre entertainment, gaming, teleconference and remote control. In 1983, Blauert discovered Head Related Transfer Function (HRTF) Technology, which is the measurement of free field sound pressure transformation from a specific location to the ears of the listener. The Headphones use HRTF functions to convolve the input signal in order to reproduce spatial audio pattern [1]. Even though they have excellent channel separation and equalization, they are little bit cumbersome and inconvenient to use when more number of users are enjoying the audio. An alternative to HRTF is the binaural or multichannel based loudspeaker technology that assumes the centered listener at some distance from the loudspeakers. The transmission path equalization is obtained by filtering the input signals with acoustic inversion matrix (AIF), obtained by inverting the acoustic transfer matrix (ATF) that contains the impulse responses of intended and unintended sound paths. To make sure that the unintended sounds are cancelled, the product of AIF and ATF should be unit matrix. This approach is generally applicable to two input sources and is called binaural audio crosstalk cancellation (CTC). When multi-channels (or) sources are involved, this becomes multi-channel audio CTC, where the outputs become the linear combination of filtering with the respective source signal [2][3][4][5][6].

For N audio sources, the outputs are derived as

$$y_L(n) = h_{L1}(n) * x_1(n) + h_{L2}(n) * x_2(n) + \ldots + h_{LS}(n) * x_S(n) = \sum_{i=1}^{S} h_{Li}(n) * x_i(n) \quad (1)$$

$$y_R(n) = h_{R1}(n) * x_1(n) + h_{R2}(n) * x_2(n) + \ldots + h_{RS}(n) * x_S(n) = \sum_{i=1}^{S} h_{Ri}(n) * x_i(n) \quad (2)$$

To obtain such transmission path equalization, the impulse responses of AIF matrix may last for several hundreds of milliseconds, which leads to the requirement of thousands of FIR filter coefficients [5]. This demands more computational power to implement these long filters on real-time DSP processors. To overcome the complexity issues, it is essential to develop new implementation techniques without compromising for performance. The aim of this paper is to investigate the optimized algorithms in order to reduce the computational complexity as well as algorithmic delay.

The general scenario of real-time filtering implementation is that the frame length, L can be chosen as very less than filter length, M. The standard and original approach for the filtering is the time domain convolution, which has the major drawback of more computations and because of this, single DSP processor may not sufficient to support the multi-channel audio CTC for long filtering. On the other hand, frequency domain approaches based on overlap save and overlap add methods that provide less computational complexity. This has the disadvantage of algorithm delay that arises due to the addition of zeros in the impulse responses to make the FFT length as a power of 2.

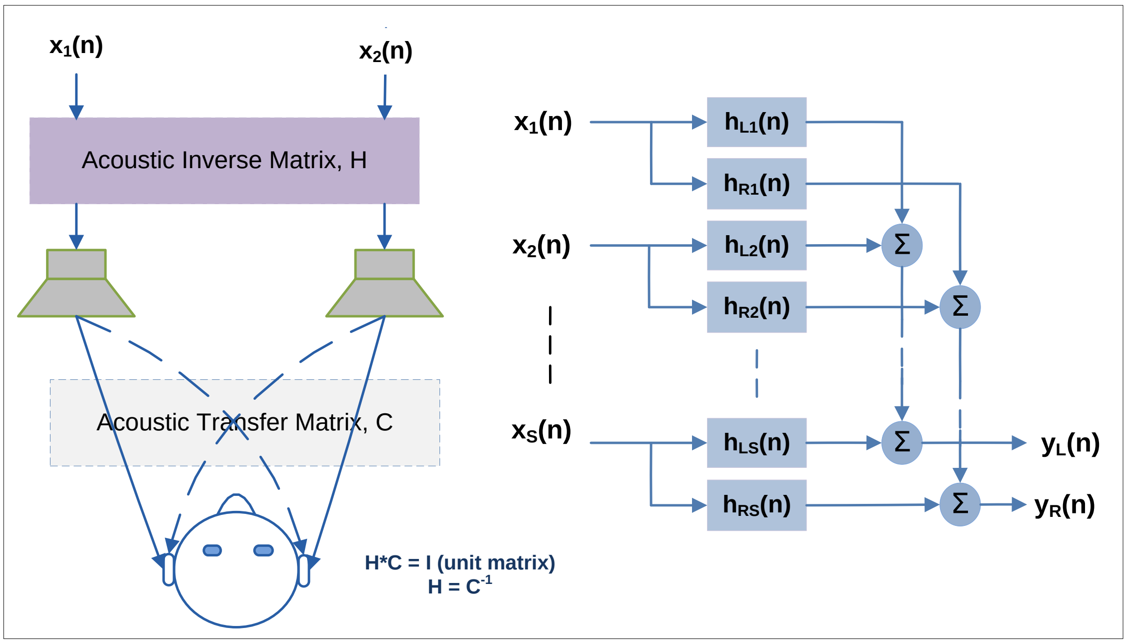

Fig.1.The binaural audio CTC system model (left) describing the perception of filtered audio signals at the listeners' ears (meant for two audio sources). The audio CTC technology with multi-channel sources (right) for real-time implementation point of view

For instance, if the processing frame length is 256 and filter has 1024 coefficients, FFT length becomes 1024 + 256 -1 = 1279. As FFT requires length to be power of 2, the length should be chosen as 2048, which means 1024 zeros are appended to the impulse response and hence the delay of 1024 samples at the output. Hence the sample delay of at least filter length will be introduced at the output [7] [8].

Martin Vetterli proposed multi-rate running convolution algorithm to reduce the delay but no major computational savings can be achieved in this method and also, this technique requires more buffering of data[9][10].

Single frequency Delay Line Filtering (SFDL) efficiently reduces the delay by partitioning the filter length into equal sizes, applying overlap save method to each partition and combining the results of all partitions. This is also called Uniform partitioned convolution as all partitions are of the same size [11][12][13][14]. A brief review of this technique is explained in next Section. If this technique is applied individually to each filter of Fig.1, the implementation complexity is huge and internal DSP memory may not hold all required buffers, particularly for long filters. To avoid such problems, SFDL is combined with mixed filtering [15] in this paper and presented as a new proposed algorithm to reduce computational complexity as well as processing delay. With efficient memory management and the properties of FFT, the proposed technique is very good choice for audio CTC of long filters.

This paper is organized as follows. Section 2 provides the review of mixed filtering and Single Frequency Delay Line Filtering methods. The combination of these two techniques is explained as proposed method in section 3. It also contains theoretical computational complexity, efficient design to avoid more buffer copying routines and better processor level optimization techniques. Section 4 details about the experimental details and results. The computational complexity of proposed method is compared with that of overlap save method. Finally chapter 5 provides the conclusion and future scope to update the proposed method.

II. Review Of Existing Work

In this section, the background details of mixed filtering and single Frequency Delay Line filtering are explained.

II.1 Mixed Filtering

The word 'Mixed filtering' means all the filtering operations of CTC can be performed in single equation. To understand this, let us form a complex sequence with real-time outputs $y_L(n)$ and $y_R(n)$.

$$y(n) = y_L(n) + jy_R(n) = \sum_{i=1}^{S} [h_{Li}(n) + jh_{Ri}(n)] * x_i(n) \tag{3}$$

The frequency domain representation of above equation is obtained as

$$Y(k) = \sum_{i=1}^{S} H_i(k) X_i(k) \tag{4}$$

$$k = 0,1,...N-1 \;\; \& \; N = L + M - 1$$

where the following assumption made in above equation.

$$H_i(k) = F[h_{Li}(n) + j\,h_{Ri}(n)] \quad \& \quad h_i(n) = h_{Li}(n) + j\,h_{Ri}(n)$$

The implementation procedure is simple. The FFTs of two consecutive source signals are obtained by applying single FFT with decomposition [9] [10]. If the system model has S sources, this step requires 0.25*S*O(N) complex multiplications and 0.5*S*(O(N)+2N) complex additions. The 2nd step is implementation of equation (4) i.e. complex frequency multiplication and addition, which requires S*N complex multiplications and (S-1)*N complex additions. Finally 3rd step requires single IFFT computations i.e. 0.5*O(N) complex multiplications and O(N) complex additions. The real and imaginary components of IFFT output yield $y_L(n)$ and $y_R(n)$ respectively. This complexity is applicable for even number of sources. For odd sources, FFT decomposition is not needed as imaginary term will be zero. In Reference [15], mixed filtering was explained for two channel sources. Here, this technique is adapted for multichannel source case.

II.2 SINGLE FREQUENCY DELAY LINE FILTERING

In overlap save method, the FFT size is chosen as N = L+M-1 in order to represent the samples of the spectrum uniquely at discrete set of frequencies. Also, N must be a power of 2. Due to this, overlap save method provides output samples delay of at least filter length, M. If filter length is M = 8192 and the system is operating at 48 kHz sampling frequency, this method provides a delay of 170.67msec (8192/48000), which is not acceptable in real time audio applications. Also, as FFT length increases, this method suffers from computational complexity. These issues are efficiently solved in single frequency delay line filtering. This method relies on partitioning the filter impulse response into equal sizes so that the overlap save method can be applied on each partition and finally the outputs of each partition will be summed to yield the filter output. This method can be understood with the following explanation [11][12][13][14].

Let us assume that x (n), h (n) and y (n) be the input, impulse response and outputs of an LTI system respectively. The lengths of input frame and filter are L and M. The impulse response is partitioned into number of parts m = M/L so that each partition has L samples.

In z-domain, the impulse response is expressed as

$$\begin{aligned} H(z) &= \sum_{n=0}^{N-1} h(n)z^{-n} = \sum_{n=0}^{L-1} h(n)z^{-n} + \sum_{n=L}^{2L-1} h(n)z^{-n} + \ldots. + \sum_{n=M-L}^{M-1} h(n)z^{-n} \\ &= \sum_{n=0}^{L-1} h(n)z^{-n} + z^{-L}\sum_{n=0}^{L-1} h(n+L)z^{-n} + \ldots. + z^{-(M-L)}\sum_{n=0}^{L-1} h(n+\overline{M-L})z^{-n} \\ &= \sum_{n=0}^{L-1}\left[h_0(n) + z^{-L}h_1(n) + \ldots + z^{-(M-L)}h_{m-1}(n)\right]z^{-n} \qquad (5) \end{aligned}$$

Where

$$\left.\begin{aligned} h_0(n) &= h(n), \\ h_1(n) &= h(n+L), \\ &\ldots \\ h_{m-1}(n) &= h(n+M-L), \end{aligned}\right\} \quad n = 0,1,..L-1 \quad \& \quad m = M/L \qquad (6)$$

These are called partitioned impulse responses of equal length, L. The total partitions are M/L.
The output in z-domain is expressed as

$$\begin{aligned} Y(z) = H(z)X(z) &= \sum_{n=0}^{L-1}\left[h(n) + z^{-L}h(n+L) + \ldots + z^{-(M-L)}h(n+\overline{M-L})\right]X(z)z^{-n} \\ &= X(z)\sum_{n=0}^{L-1} h_0(n)z^{-n} + z^{-L}X(z)\sum_{n=0}^{L-1} h_1(n)z^{-n} + \ldots + z^{-(M-L)}X(z)\sum_{n=0}^{L-1} h_{m-1}(n)z^{-n} \\ &= X(z)H_0(z) + \underline{z^{-L}X(z)}H_1(z) + \ldots + \underline{z^{-(M-L)}X(z)}H_{m-1}(z) \qquad (7) \end{aligned}$$

From the above equation, output, Y (z) is expressed as sum of all partitioned outputs. For 1st frame, the FFT is calculated for x (n). When 2nd frame arrives, FFT of x (n) is delayed by L samples and become the input to 2nd partitioned filter. During 3rd frame, FFT of 2nd frame becomes the input to 2nd partitioned filter and FFT of 1st frame becomes the input to 3rd partitioned filter. In this way, only frequency samples of input frames will be delayed and they are summed after complex frequency multiplication with the respective FFTs of partitioned filters before acting as input to IFFT. As FFT values of input samples are delayed and all delays are equal in size, this technique is single frequency delay line filtering. A block diagram of this method is shown in Fig.2. As overlap save method is applied to each partitioned filter, the FFT size of each filter is derived as L + L -1 = 2L as length of each partitioned filter now becomes L. So, it is required to append zeros of length, L, results in output sample delay of L. Hence the delay is reduced from M to L. For our example, the output sample delay becomes 5.33msec (256/48000) for a frame length of 256 samples.

For each frame, one FFT and one IFFT of size 2L are required. The frequency multiplier length is 2L. Such frequency multipliers are m = M/L and hence complex multiplications of *2L*.M/L = 2M are needed. All frequency multipliers have to be added before providing as input to IFFT and hence 2L (m-1) = 2(M - L) complex additions are required.

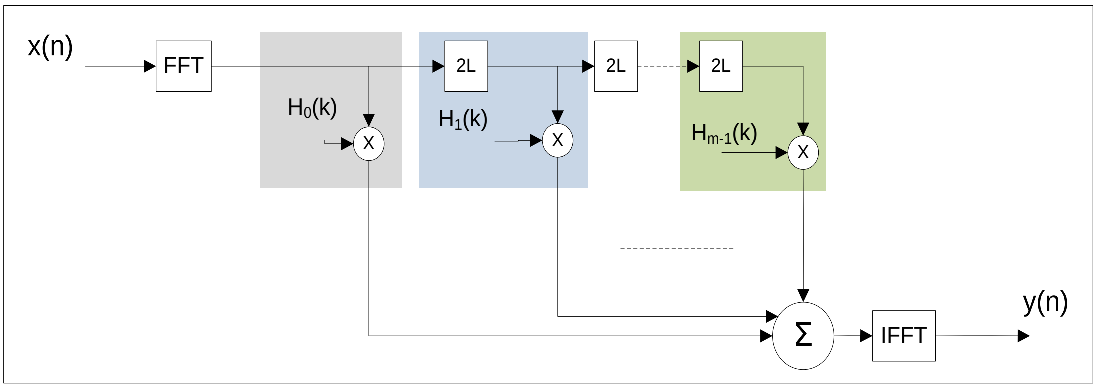

Fig.2. Interpretation of SFDL using block diagram representation. A delay of L samples become 2L samples in FFT domain due to the FFT length N= L+L-1≈2L.

III. Mixed Single Frequency Delay Line Filtering Algorithm

The proposed algorithm is the combination of mixed filtering and SFDL applied to multi-channel audio CTC structure of Fig1. The mathematical model of this is derived as follows. Note that the mathematical model was provided in z-domain and the computational complexity is provided in FFT domain for easy understanding and clarity.

Let us assume that the CTC model has S number of sources with each source is filtered by complex impulse response of length, M. The complex impulse response and its frequency response are represented as

$$h_i(n) = h_{Li}(n) + j\,h_{Ri}(n), \qquad i = 1,2,\ldots S \tag{8}$$

and

$$H_i(z) = \sum_{n=0}^{M-1} h_i(n) z^{-n}, \quad i = 1,2,\ldots S \tag{9}$$

Now each impulse response is partitioned into m=M/L parts of equal length, say, L. The partitioned impulse responses are represented by

$$h_i(n) = \{h_{i,0}(n), h_{i,1}(n), .. h_{i,m}(n)\}, \qquad i = 1,2,\ldots S \tag{10}$$

And their frequency responses are given by

$$H_{i,j}(z) = \sum_{n=0}^{L-1} h_{i,j}(n) z^{-n}, \qquad i = 1,2,\ldots S \quad \& \quad j = 0,1,\ldots m-1 \tag{11}$$

The output complex frequency response is obtained by placing equation (11) in equation (7)

$$\begin{aligned} Y(z) &= \sum_{i=1}^{S} H_i(z) X_i(z) \\ &= \sum_{i=1}^{S} X_i(z) H_{i,0}(z) + \underline{z^{-L} X_i(z)} H_{i,1}(z) + \ldots + \underline{z^{-(M-L)} X_i(z)} H_{i,m-1}(z) \\ &= \sum_{i=1}^{S} \sum_{j=0}^{M/L-1} \left[z^{-jL} X_i(z) \right] H_{i,j}(z) \end{aligned} \tag{12}$$

For the case of stereo channel inputs, S=2. For this case, equation (12) becomes

$$Y(z) = \sum_{j=0}^{M/L-1} \left[z^{-jL} X_1(z)\right] H_{1,j}(z) + \sum_{j=0}^{M/L-1} \left[z^{-jL} X_2(z)\right] H_{2,j}(z) \tag{13}$$

The implementation model can be represented in block diagram as shown in Fig. 3. The SFDL filtering algorithm can be applied to each input and the associated complex impulse response. After addition of complex frequency multiplier outputs of each partitioned filter, a single IFFT is sufficient to provide the filtered outputs yL(n) and yR(n) in real and imaginary parts of output y(n).

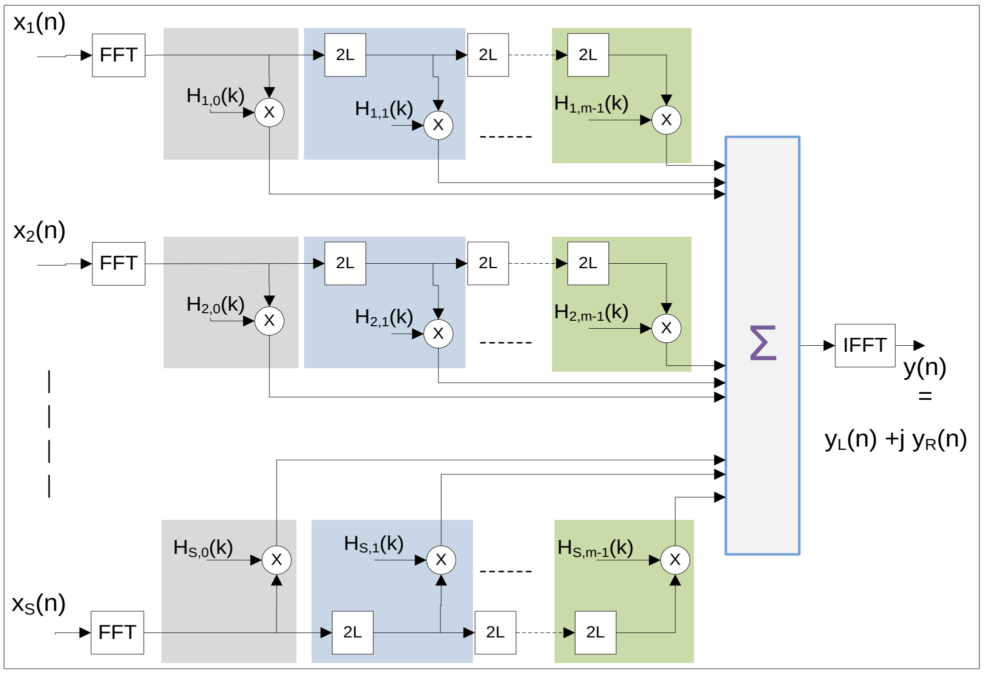

Fig.3. Block diagram of Mixed SFDL filtering algorithm

III.1 THEORETICAL COMPUTATIONAL COMPLEXITY

The following table provides the details of computations required for proposed algorithm. Note that $O(N)=N.\log_2 N$. Initially the computations are shown for a pair of input sources and the associated complex impulse responses. In Remarks column, this is generalized for S number of sources.

Table 1: Computational complexity of MSFDL algorithm

To calculate	Computational complexity		Remarks
	Complex multiplications	Complex additions	
$X_i(k)$ & $X_{i+1}(k)$	0.5 O(2L)	O(2L) + 4L	FFTs of two consecutive sources can be found using single FFT with decomposition [7]. When S is even, complex multiplications = 0.25*S*O(2L) complex additions = 0.5*S*(O(2L) + 4L). When S is odd, decomposition is not required for the last source when calculating its FFT as imaginary term of input becomes zero. complex multiplications = 0.25*(S-1)*O(2L)+0.5*O(2L) complex additions = 0.5*(S-1)*(O(2L) + 4L) +O(2L).
$X_i(k)H_i(k)$	2L * M/L = 2M	2(M-L)	This computation is needed for one input source with the associated filters. Each Partitioned frequency multiplication requires 2L multiplications. Such partitions are M/L and hence 2M complex multiplications are needed. All partitioned multiplier outputs are to be added, which requires 2(M-L) complex additions. For S sources, complex multiplications = 2*S*M complex additions = 2*S*(M-L).
Y(k)	-	(S-1)*2L	This is required due to the addition of individual FFT outputs. Equation (12) can be taken as reference for this calculation.
y(n)	0.5*O(2L)	O(2L)	This computation is required for IFFT calculation.

For even Sources,

Complex multiplications	=	0.25*S*O (2L) + 2*S*M + 0.5*O (2L)
Complex additions	=	0.5*S*(O (2L) + 4L) + 2*S*(M-L) + (S-1) *2L + O (2L)

For odd Sources,

Complex multiplications	=	0.25*(S-1)* O (2L) + O (2L) + 2*S*M
Complex additions	=	0.5*(S-1)*(O (2L) + 4L) + 2*O (2L) + 2*S*(M-L) + (S-1) *2L

IV. EFFICIENT MEMORY MANAGEMENT OF MSFDL FILTERING ALGORITHM

From the algorithm described so far, a lot of memory is required for storing FFT values of filter coefficients corresponding to all channels and for frequency delay buffers. If the filter length is 2048 and all the coefficients are stored in 32-bit floating point format, 32864 bytes of memory is needed for real and imaginary buffers, 16384 bytes for each. Also same size of memory is required for frequency delay buffers. This is needed for support of one input channel. Then we can imagine how much memory is required to support multi-channel cases. Most of the processors are available with less internal RAM memory and more memory with external RAM such as SDRAM, SRAM, etc. Because of less internal RAM, it is not possible to store these buffers in internal DSP memory. The major implementation complexity is involved in complex frequency multiplication as shown in Fig. 3. When implementing this method on real-time DSP processors, memory buffers should be managed efficiently without an increase in computational complexity. From here on, the word 'complex buffer' is used to represent the combination of real and imaginary buffers in the explanation.

Any DSP processor has a dedicated memory bus for DMA process. Using DMA, it is possible to copy the external memory contents into internal RAM based on processing needs. In this paper, the design was explained based on the architecture of SHARC DSP processor. SHARC 214xx series DSP has 5MB of internal memory and 64MB SDRAM. It is almost similar for any floating point processor.

In implementation of complex frequency multiplication for each partition, two complex buffers of size (one buffer of delayed frequency buffer and another one is FFT of filter coefficients), 2L are required in the proposed algorithm. Assume that the FFT values of all complex filters are stored in external RAM. For each partition, real and imaginary buffers are arranged in sequential order i.e. 2L size real buffer followed with 2L imaginary buffer of 1st partition and the same order of buffers for 2nd partition and so on. In the same way, the frequency delay buffers are also arranged in independent memory section of external RAM.

For processing needs, assume that two dedicated memory sections are reserved for real and imaginary buffers as shown in Fig. 4. Initially, the complex buffers of (m-1)th partition and that of 1st input frame are copied into one set of internal dedicated memory sections using DMA process. After this copying is completed, DMA process is enabled for copying the complex buffers of (m-2)th partition and 1st delayed complex input frame buffers into 2nd set of dedicated memory sections. As both memory sections are independent in external RAM, two separate channels are allocated for DMA copying process, say channel 0 for complex buffers of filters and channel 1 for delayed complex input frame buffers. During DMA process, the DSP starts its complex frequency multiplication on its core. So, both operations are performed in parallel. The execution time here is equal to the maximum of DMA process and core process. Once core process is completed, DMA also has to be verified for its completion. After DMA is completed, again it is configured to copy the (m-3)th partitioned complex frequency buffers and 2nd delayed complex input frame buffers into first set of dedicated memory sections. When DSP starts its core process, DMA copying is going on separate memory bus. After copying is over, DMA is verified for completion. In this way, the dedicated memory sections are filled alternatively with external complex buffers. This will be continued till all partitions are completed and repeated for all channel contents.

The high level summary of the implementation steps are provided below.

During initialization,

Calculate the FFTs of all partitioned filters with FFT size of 2L each. Repeat the same process for all filter sets based on the input source. Store these FFT values in external memory in the sequence shown in Fig. 4.

For each frame,

1) Receive the 1st input frame of size L samples of channel 1 and 2. Calculate their FFTs using complex FFT with decomposition. Store these FFT values in the external memory buffers using DMA process
2) Fill output buffer with zero contents to store the real and imaginary outputs of complex frequency multiplication (CFM).
3) Copy the complex buffers of filters and that of input frames into internal dedicated memory sections using DMA process. Verify for DMA completion.
4) Now again enable DMA process to copy next partition's complex buffers and 1st delayed buffer contents into 2nd dedicated internal memory sections. Start the core process to compute CFM of 1st set dedicated memory buffer contents.
5) After core process is completed, verify for DMA completion.
6) Continue steps 3 and 4 till all partitions are completed with the alternative internal memory sections to be used for storing complex buffer contents. For each iteration, make sure to add the complex output of CFM with that of output.
7) Repeat steps 1 to 5 for all input sources
8) After CFM of all sources is completed, calculate the IFFT for the output and send the real and imaginary buffer contents of size L samples as filtered contents.

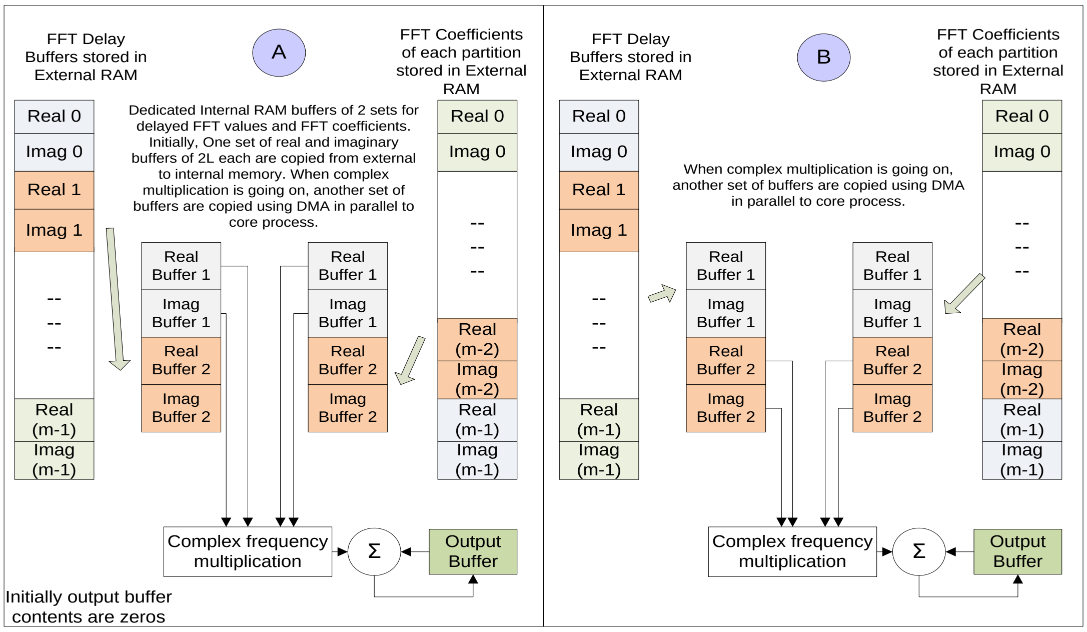
FFT Delay Buffers stored in External RAM
A
FFT Coefficients of each partition stored in External RAM
Dedicated Internal RAM buffers of 2 sets for delayed FFT values and FFT coefficients. Initially, One set of real and imaginary buffers of 2L each are copied from external to internal memory. When complex multiplication is going on, another set of buffers are copied using DMA in parallel to core process.
Real 0
Imag 0
Real 1
Imag 1
Real (m-1)
Imag (m-1)
Real (m-2)
Imag (m-2)
Real Buffer 1
Imag Buffer 1
Real Buffer 2
Imag Buffer 2
Complex frequency multiplication
Σ
Output Buffer
Initially output buffer contents are zeros
B
When complex multiplication is going on, another set of buffers are copied using DMA in parallel to core process.

Fig.4. Block diagram showing the efficient Memory management for Mixed SFDL filtering on SHARC DSP processors using DMA processing. The diagram shows the processing for just initial two frames. The output buffer is initially filled with zeros before starting complex frequency multiplication. This is applicable for one input source with the associated complex buffers. The same is repeated for S number of sources in general.

V. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed algorithm "Mixed Single Frequency Delay Line filtering" was implemented on ADSP SHARC 21469 Ez-Kit Lite board [16] as per the procedure stated in section IV. The frame length was taken as L = 256. The frequency multiplication was implemented efficiently with 4 parallel instructions inside the loop and making use of SIMD (Single Instruction Multiple Data). For FFT & IFFT, the built-in FFT software modules (available with installation package) were used.

During DMA process, it is required to make sure that the execution times of DMA and core process (i.e. complex frequency multiplication) are almost same. DMA execution is dependent on the buffer size to be copied (Here it is 4L in size i.e. for real and imaginary buffers) and the clock ratio of DSP and external RAM. The core process was optimized to make sure that its execution time should not exceed that of DMA process by making use of multiply add and SIMD (Single Instruction multiple data) instructions in SHARC architecture. The acoustic filter lengths from 1024 to 16384 with a step of 512 were considered as filter lengths. The computational complexity in terms of Mega Peak cycle count was recorded for different input channels. The Mega Peak cycle count at various filter lengths for different input channels was given in Table 2.

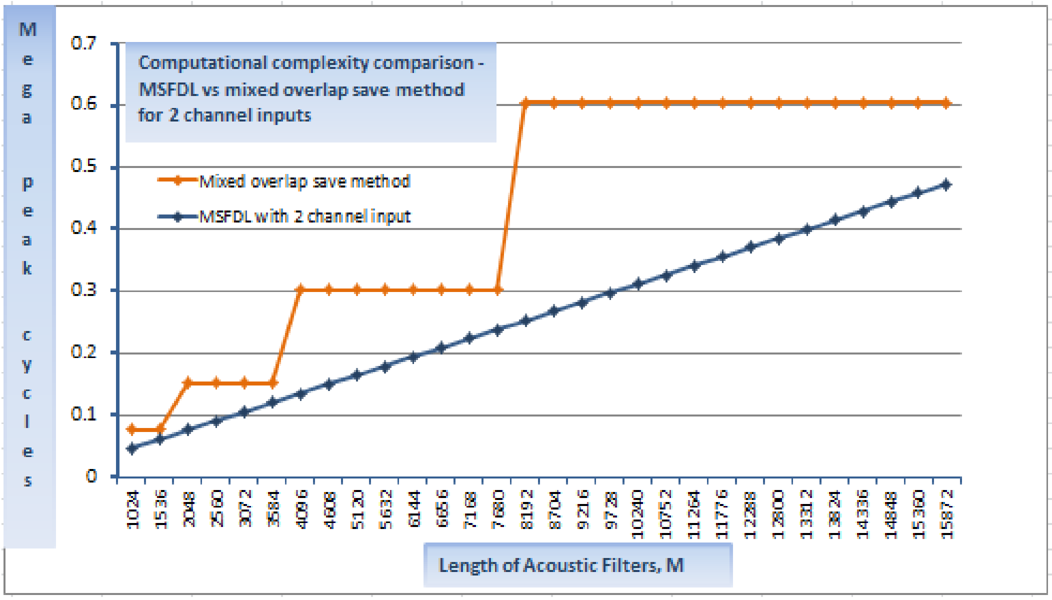

Fig.5. Graph showing the comparison of computational complexity between Mixed overlap save method and MSFDL algorithm for different filter lengths with stereo channel inputs. The frame length of the experiment is 256.

The same experiment was done with mixed overlap save method for stereo inputs. The comparison of computational complexity of this method and that of proposed method was shown in Fig. 5. From the graph, the mixed overlap save method obviously needs more computations than that of proposed method. Also, the curve resembles the staircase and sudden increase in computational complexity at power of 2 filter lengths and maintained the same complexity till next power of 2. This is expected because FFT/IFFT size is derived as N = L+M-1 in overlap save method. If N is not a power of 2, it will be chosen as next immediate power of 2.

Table 2**:** MSFDL algorithm - Mega Peak cycle count at different filter lengths with various input channels. The frame length was taken as 256.

Length of Acoustic filters, M	Mega Peak cycle count (rounded to 5 digits after decimal point)			
	2ch input	3ch input	4ch input	5ch input
1024	0.04594	0.06693	0.08393	0.10491
1536	0.06069	0.08905	0.11342	0.14178
2048	0.07543	0.11117	0.14291	0.17864
2560	0.09018	0.13329	0.17241	0.21550
3072	0.10492	0.15540	0.20189	0.25237
3584	0.11967	0.17752	0.23138	0.28923
4096	0.13442	0.19964	0.26087	0.32609
4608	0.14916	0.22176	0.29036	0.36296
5120	0.16391	0.24388	0.31986	0.39982
5632	0.17865	0.26605	0.34935	0.43669
6144	0.1934	0.28811	0.37884	0.47355
6656	0.20814	0.31023	0.40833	0.51042
7168	0.22289	0.33235	0.43782	0.54728
7680	0.23764	0.35447	0.46731	0.58414
8192	0.2524	0.37659	0.49680	0.62101
8704	0.26713	0.39870	0.52629	0.65787
9216	0.28187	0.42082	0.55579	0.69474
9728	0.29662	0.44294	0.58528	0.7316
10240	0.31136	0.46506	0.61477	0.76846
10752	0.32611	0.48718	0.64426	0.80533
11264	0.34085	0.5093	0.67375	0.84219
11776	0.3556	0.53141	0.70324	0.87906
12288	0.37035	0.55355	0.73273	0.91592
12800	0.38509	0.57565	0.76222	0.95278
13312	0.39984	0.59777	0.79171	0.98965
13824	0.41458	0.61989	0.82121	1.02651
14336	0.42933	0.64201	0.85069	1.06338
14848	0.44407	0.66412	0.88019	1.10024
15360	0.45882	0.68624	0.90968	1.13710
15872	0.47356	0.70836	0.93917	1.17397
16384	0.48831	0.73048	0.96866	1.21083

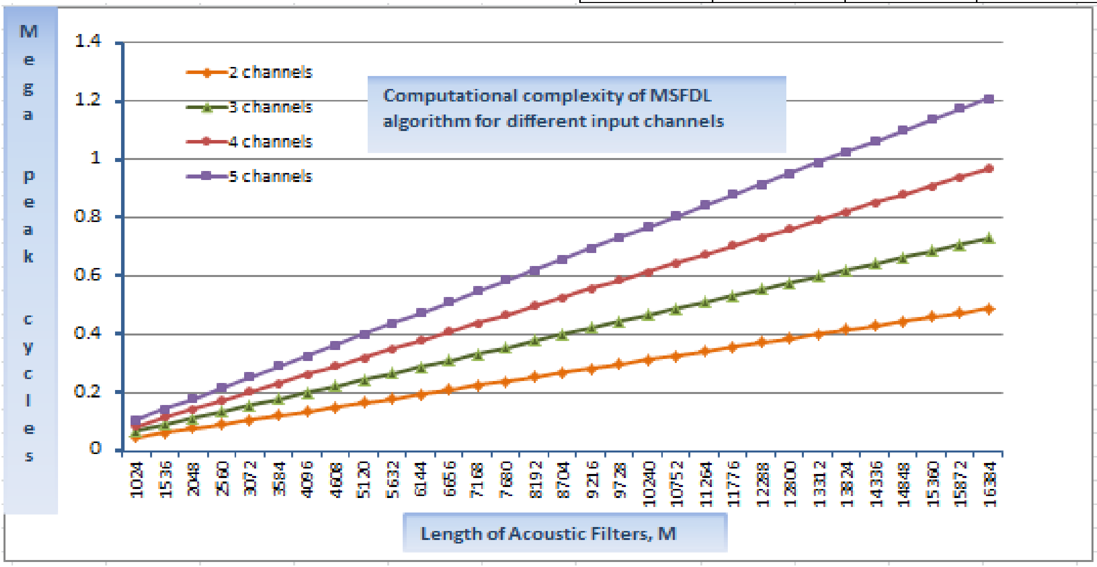

Fig.6. Graph showing the comparison of computational complexity of MSFDL algorithm for different filter lengths with various input sources. The frame length of the experiment is 256.

The variation of computational complexity for different input sources was provided in Fig.6. The graph shows linear relationship between the filter length and the computational complexity with increase in slope as number of sources is increased. This is expected because as number of sources is increased, the computations in complex frequency multiplication as well as number of FFTs will be increased.

The reason behind choosing floating point processor for implementation is because of the algorithm implementation complexity and computations. In high end applications such as audio surround systems, fixed point processors are not encouraged due to the lack of output quality.

VI. CONCLUSION

An efficient algorithm for the implementation of multichannel audio crosstalk cancellation was presented in this paper by combining the techniques of mixed and single frequency delay line filtering techniques as MSFDL algorithm. To reduce memory issues on DSP processors during implementation, efficient design was provided by utilizing the resources of DSP processor. The results were compared against mixed overlap save method for various filter lengths and these indicate that the proposed technique yields better results, particularly, at long filter lengths. Also the variation of computational complexity for different input sources were shown clearly.

The future scope of this work is mixed multiple frequency delay line filtering, in which the impulse response is partitioned non-uniformly and all non-uniform filters run in parallel. This requires multicore DSP for implementation point of view and suitable for filter lengths more than 16384. The main computations of the proposed algorithm are in complex frequency computation block. One can investigate on the methods related to this area to reduce computations. Also it is better to investigate on suitable frequency domain methods other than FFT based techniques. Not only algorithm optimization is enough but processor level optimization is also very important to achieve good computational complexity. For this, suitable DSP processor should be chosen to handle parallel instructions.

REFERENCES

[1] Blauert J., Spatial Hearing: The Psychophysics of Human Sound Localization, MIT Press, Cambridge, 1983.

[2] M. Otani and S. Ise, Fast calculation system specialized for head-related transfer function based on boundary element method, Journal of Acoustical Society of America, Vol. 119, 2006, No. 5, 2589-2598

[3] Kirkeby ole, Rubak Per, Nelson Philip A. and Farina Angelo, Design of Crosstalk cancellation Networks by using Fast deconvolution, Audio Engineering Society 15, May 1999, 9900 – 9905

[4] Lentz Tobias and Scmitz Oliver, Adaptive Cross-talk cancellation system for a moving listener, AES 21st International Conference Proc., June 2002. Paper No. 00134

[5] Linwang, Fuliang Yin and zhe Chen, A Stereo Crosstalk cancellation system based on common- acoustical pole/zero model, AES, August 2010

[6] William G. Gardner, 3-D Audio Using Loudspeakers, School of Architecture and Planning, Massachusetts Institute of Technology, September, 1997

[7] John G. Proakis and Dimitris G. Manolakis, Digital Signal Processing Principles, Algorithms and Applications, (New Jersey, Prentice Hall International, 3rd Edition, Page No. 430 to 476)

[8] Richard G Lyons, Understanding Digital Signal Processing, (New Jersey, Prentice Hall International, 3rd Edition, published on November 11, 2010).

[9] M.Vetterli, Running FIR and IIR Filters using Multi-rate Filter Banks, IEEE transactions on Acoustics, Speech and Signal Processing, Vol. 36, No.5, May 1988.

[10] Jason R. VandeKieft, Computational improvements to linear convolution with multi-rate filtering methods, April 30, 1998, http://mue.music.miami.edu/thesis/jvandekieft/jvtitle.htm.

[11] Garcia Guillermo, Optimal Filter Partition for efficient Convolution with short input/output Delay, AES 113th International Conference Proc., October 2002, pp. 2660.

[12] WG Gardiner, Efficient Convolution without input-output delay, Journal of AES, Vol. 43, No. 3, 1995, pp. 127-136.

[13] J. Hurchalla, A time distributed FFT for efficient low latency convolution, AES Convention 129, November 2010, Paper No.8257

[14] J. Hurchalla, Low latency convolution in one dimension via two dimensional convolutions-An intuitive approach, AES Convention 125, October 2008, Paper No. 7634

[15] SreenivasaRao. Ch, R. Udayalakshmi and Jeyasingh P., Fast implementation of audio crosstalk cancellation of audio crosstalk cancellation on DSP processors, AES 45 Conference Proc., March 1-4, 2012, Paper No. 2-2

[16] Analog Devices Inc., "ADSP-214xx SHARC Processor Hardware Reference Manual", Rev 0.3, Part Number 82-000469-01, July 27, 2010.