OPTIMAL IMPLEMENTATION METHODS FOR AUDIO CROSSTALK CANCELLATION ON DSP PROCESSORS

Chunduri SreenivasaRao

ECE Department, KLUniversity, Green Fields,
Guntur, Andhra Pradesh 522 502, India
chsrinivas19800305@rediffmail.com
http://www.kluniversity.in

Dhulipalla VenkataRao

Principal, PPDCET, Nunna,
Vijayawada, Andhra Pradesh 521 212, India
dvenky221101@rediffmail.com
http://www.ppdcet.ac.in

Abstract

In general, frequency domain based techniques are preferred to time domain techniques for the implementation of audio crosstalk cancellation due to time domain techniques suffer from more computations. In this paper, the computational difficulty of real FFT on DSP processors is analyzed and to overcome the disadvantages, two optimization methods (1. Parallel Filtering, 2. Mixed Filtering) are proposed for the implementation of Audio Crosstalk cancellation. These methods are developed based on *fast convolution using overlap save method & Finding DFTs of two real sequences using single Complex DFT*. By applying these properties to two real outputs of stereo crosstalk cancellation structure, valid equations are derived for implementation. The proposed methods are simulated in real-time on 32 bit floating point processor. The results show that the proposed methods can significantly reduce the computational complexity when compared to conventional time domain convolution & frequency domain based individual filtering without affecting the performance.

***Keywords*:** Audio Crosstalk Cancellation; Complex FFT; Computational Complexity.

1. Introduction

Over the decades, 3D virtualization technique has got demand in many audio applications where audio crosstalk cancellation (CTC) is one of them. Initially, HRTF (Head Related Transfer Function) technology was developed to sense the direction of the source for head phone processing [2][3]. The same technology was extended later to the home entertainment applications using loud speakers. With loudspeakers, the delivery of binaural signals at the listeners' ears leads to the unintended reception of so called crosstalk component, which needs to be cancelled out for proper sound reproduction. CTC algorithms can be classified into direct and adaptive implementations. In direct implementation, fixed filter coefficients will be used as the position of the listener is fixed whereas adaptive algorithm dynamically derives the CTC filter coefficients during run-time based on the movement of listener [4][5][6].

To achieve high fidelity sound reproduction, it is desired to maintain the filter taps of CTC as long as possible in these algorithms. Due to long taps, implementation of these filters demands high computational complexity on DSP processors. Traditionally, frequency domain based overlap save approach is one of the best methods that provide less computational complexity for long data sequences [1][14]. However, implementation of overlap save method for all filters individually yields more computational complexity. A typical CTC structure for stereo source was shown in Fig. 1 [4]. The filters indicated by $h_{12}(n)$ and $h_{21}(n)$ are designed in such a way that the crosstalk due to $x_1(n)$ and $x_2(n)$ needs to be cancelled at $y_2(n)$ and $y_1(n)$ respectively [7][10][11]. This structure needs 2 FFTs, 4 Complex Multiplications and 2 IFFTs (one IFFT for frequency sum $Y_{11}(k)+Y_{21}(k)$ and another for $Y_{12}(k)+Y_{22}(k)$) for individual filter implementation using overlap save method.

In general, FFT of any real signal, say *x(n)*, is always symmetric about real axis i.e. $X(N-k) = X^*(k)$ for $k = 0,1,..\ N-1$. When implementing real FFT on DSP processors, it is sufficient to calculate $X(0)$ to $X(N/2-1)$ and symmetric property shall be used to find out $X(N/2+1)$ to $X(N-1)$. But the real FFT values $X(0)$ & $X(N/2)$ are not equal. As the calculation of $X(N/2)$ depends on stage by stage calculations in Radix-2 butterfly FFT, the

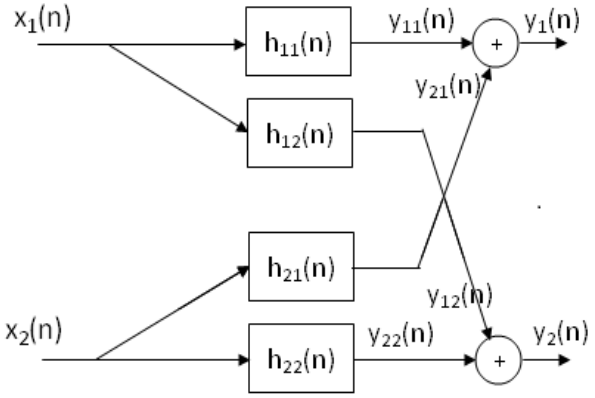

Fig.1. Audio CTC Structure for Stereo Input

required computations of real FFT are more than half of computations needed for complex FFT. Also the full capacity of parallel instruction performance may not be utilized because of avoiding stage by stage butterfly operations. Instead, complex FFT could be preferred for two signals simultaneously rather than two individual real FFT calculations. The proposed methods in this paper were developed based on FFT calculation of two real sequences using single complex FFT [1]. This property was applied to output signals of the CTC structure of Fig. 1 and a set of valid equations were derived for real time outputs. Based on the derived equations, the system was realized for implementation.

These methods are not published in any journal till now and are the basic methods, could be applied with not only overlap save method but also any kind of frequency domain technique.

This paper is organized into 5 sections. Section 1 explains introduction. Section 2 provides the mathematical model for optimized methods (a) *Parallel filtering* (b) *Mixed filtering*. Section 3 compares the performance of proposed methods with that of time domain method. Section 4 provides the computational results and gives the advantages of these methods. Finally Section 5 provides conclusion of the paper.

2. Optimal Implementation Methods

The basic idea lies behind the proposed methods is the zero phase of real signals. When evaluating FFT of two real-time signals of equal length, say N, a complex sequence could be formed with 1st signal as real buffer and 2nd signal as imaginary buffer. Then N-point FFT could be applied for the complex sequence and individual DFTs of two real-time signals will be derived using decomposition as given in Reference [1]. The inputs and outputs of the CTC structure are real in nature. By taking advantage of this property, the implementation could be optimized in which the computational complexity is far more better than if the filters were implemented individually just by using overlap save method. Before going into the analysis, few of the mathematical related equations were given here for better understanding. The time domain based equations of CTC structure are given by

$$y_{11}(n) = x_1(n) * h_{11}(n) \quad (1)$$
$$y_{12}(n) = x_1(n) * h_{12}(n) \quad (2)$$
$$y_{21}(n) = x_2(n) * h_{21}(n) \quad (3)$$
$$y_{22}(n) = x_2(n) * h_{22}(n) \quad (4)$$

The equivalent frequency domain based equations of the above set are given by

$$Y_{11}(k) = X_1(k)\, H_{11}(k) \quad (5)$$
$$Y_{12}(k) = X_1(k)\, H_{12}(k) \quad (6)$$
$$Y_{21}(k) = X_2(k)\, H_{21}(k) \quad (7)$$
$$Y_{22}(k) = X_2(k)\, H_{22}(k) \quad (8)$$

where $k = 0,1,...N-1$, the frequency domain index. Here $N = L+M-1$ should be selected as power of 2 with the assumption that Radix 2 FFT is used for DFT calculation. L and M are the input frame length and filter lengths respectively. If N is not a power of 2, sufficient zeros should be padded. Also the final outputs were derived as

$$y_1(n) = y_{11}(n) + y_{21}(n) \quad (9)$$
$$y_2(n) = y_{12}(n) + y_{22}(n) \quad (10)$$

2.1. *Parallel Filtering*

The word '*Parallel*' is used because filtering operation can be performed for two filters in parallel. This could be achieved by forming a complex signal with the real output signals $y_{11}(n)$ & $y_{12}(n)$. With this assumption, the real & imaginary terms of the formed complex signal can be considered as individual filtering operations. The frequency domain analysis for this was given below. Let us form two complex signals with real outputs of CTC structure as

$$y_L(n) = y_{11}(n) + jy_{12}(n) \quad (11)$$
$$y_R(n) = y_{21}(n) + jy_{22}(n) \quad (12)$$

The equivalent frequency domain representation can be given as

$$Y_L(k) = Y_{11}(k) + jY_{12}(k) \quad (13)$$
$$Y_R(k) = Y_{21}(k) + jY_{22}(k) \quad (14)$$

After substituting equations (5) to (8) in above equations, one can obtain

$$Y_L(k) = X_1(k)H_{11}(k) + jX_1(k)H_{12}(k) = X_1(k)[H_{11}(k) + jH_{12}(k)] = X_1(k)H_1(k) \quad (15)$$
$$Y_R(k) = X_2(k)H_{21}(k) + jX_2(k)H_{22}(k) = X_2(k)[H_{21}(k) + jH_{22}(k)] = X_2(k)H_2(k) \quad (16)$$

where

$$H_1(k) = H_{11}(k) + jH_{12}(k) = \{Re[H_{11}(k)] - Im[H_{12}(k)]\} + j\{Im[H_{11}(k)] + Re[H_{12}(k)]\}$$
$$H_2(k) = H_{21}(k) + jH_{22}(k) = \{Re[H_{21}(k)] - Im[H_{22}(k)]\} + j\{Im[H_{21}(k)] + Re[H_{22}(k)]\}$$

In above equations, *Re[.]* & *Im[.]* are the real & imaginary parts of coefficient FFTs. From these equations, one can conclude that it is sufficient to store real & imaginary parts of $H_{11}(k) + jH_{12}(k)$ term in DSP memory instead of storing real & imaginary parts of $H_{11}(k)$ & $H_{12}(k)$ separately. Similarly for the case of $H_2(k)$ also. For example, if filter taps of $h_{11}(n)$ is $M=2048$ & frame size is $L = 256$, then $N = L+M-1 = 2033$ as per the overlap save method. Since FFT was used to find out $H_{11}(k)$, N should be considered as next higher power of 2 i.e. 4096, which is FFT length. In this case, if individual FFTs of $h_{11}(n)$ & $h_{12}(n)$ are stored in memory, the memory required is 16384 considering both real & imaginary parts, whereas *Parallel Filtering* requires 8192 only.

Computations point of view, this approach requires single FFT (for inputs $x_1(m, n)$ and $x_2(m, n)$) with decomposition [1], 2 frequency multiplications (as per equations 15 & 16) and 2 IFFTs (applied to equations 15 & 16) to obtain filtered outputs. The block diagram for this implementation and the theoretical computational complexity are provided in Fig. 2 and table 1 respectively. Assumption was made for $O(N) = Nlog_2N$.

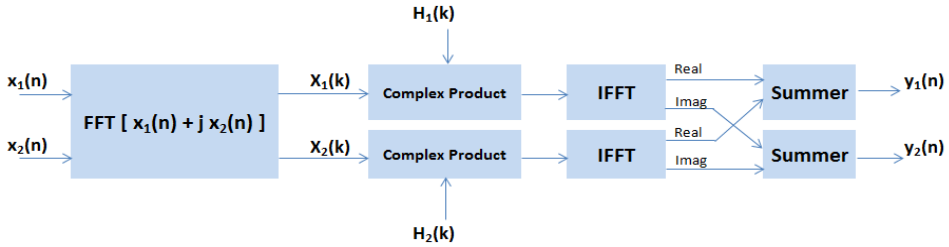

Fig.2. Block Diagram - *Parallel Filtering* Implementation

Table 1. Computational Complexity of Parallel Filtering.

To Calculate	Complex Multiplications	Complex Additions
$X_1(k)$ & $X_2(k)$	$0.5O(N)$	$O(N)+2N$
Eqns (15) & (16)	$2N$	-
IFFT of (15) & (16)	$O(N)$	$2O(N)$
Total	$2N + 1.5\ O(N)$	$2N + 3O(N)$

2.2. *Mixed Filtering*

Mixed Filtering means that all filtering operations of CTC structure are evaluated simultaneously. This is possible by forming a complex sequence with final outputs of CTC i.e. $y_1(n)$ & $y_2(n)$, assuming both outputs could be evaluated in parallel. In this method, equations (11) & (12) will be evaluated but IFFT won't be applied. Instead the multiplied results of both equations are added & single IFFT will be applied to the sum. A complex signal with final outputs $y_1(n)$ & $y_2(n)$ is could be assumed as follows.

$$y(n) = y_1(n) + jy_2(n)$$

Its frequency domain representation is given by

$$Y(k) = Y_1(k) + jY_2(k)$$

This could be simplified as

$$\begin{aligned} Y(k) &= [X_1(k)H_{11}(k)+X_2(k)H_{21}(k)]+j[X_1(k)H_{12}(k)+X_2(k)H_{22}(k)] \\ &= X_1(k)[H_{11}(k)+ jH_{12}(k)]+X_2(k)[H_{21}(k)+ jH_{22}(k)] \\ &= X_1(k)H_1(k)+X_2(k)H_2(k) \end{aligned} \qquad (17)$$

where the same assumption was made for $H1(k)$ & $H2(k)$ as that in Parallel Filtering.

From eqn (17), one can clearly say that the complexity of *Mixed Filtering* includes single FFT (for inputs $x_1(n)$ & $x_2(n)$) with decomposition, 2 frequency multiplications & single frequency addition (as per equation 17) and single IFFT (applied to equation 17) to obtain $y(n)$. The real & imaginary parts of $y(n)$ yield $y_1(n)$ & $y_2(n)$ respectively. With this approach, one IFFT can be reduced at the cost of complex frequency addition when compared to *Parallel Filtering*. The software flow graph for *Mixed Filtering* implementation was given in Fig 3. The theoretical computational complexity is provided in table 2. Assumption was made for $O(N)= Nlog_2N$.

In surround channel audio processing applications, more than two filtered outputs could be summed at $y_1(n)$ & $y_2(n)$ in order to cancel the crosstalk required for each channel. Due to the single IFFT usage, *Mixed Filtering* yields computational savings for such applications. A general frequency domain representation of equation (17) for n surround audio channels could be expressed as

$$Y(k) = X_1(k)\,H_1(k)+X_2(k)\,H_2(k)+ X_3(k)\,H_3(k)+ \ldots+X_n(k)\,H_n(k)$$

where

$$\begin{aligned} H_1(k) &= H_{11}(k)+ jH_{12}(k) \\ H_2(k) &= H_{21}(k)+ jH_{22}(k) \\ &\ldots\ldots \\ H_n(k) &= H_{n1}(k)+ jH_{n2}(k) \end{aligned}$$

The CTC structure of surround channel processing could be imagined based on this general equation and its description is out of the scope of this paper.

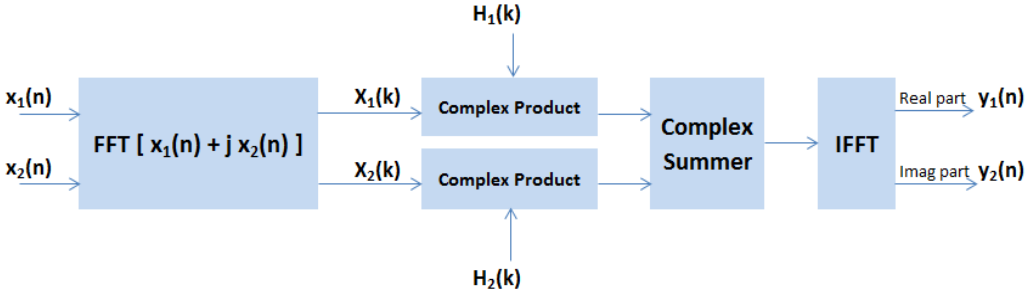

Fig.3. Block Diagram - *Mixed Filtering* Implementation

Table 2. Computational Complexity of Mixed Filtering.

To Calculate	Complex Multiplications	Complex Additions
$X_1(k)$ & $X_2(k)$	0.5O(N)	O(N)+2N
Eqn (17)	2N	N
IFFT of (17)	0.5O(N)	O(N)
Total	2N + O(N)	3N + 2O(N)

3. Performance Evaluation

Due to the quantization effects in DSP processors, the noise power of the frequency domain methods may vary from that of time domain outputs even though the signal power is same for both of the methods. Since FFT is used in these methods, the noise floor depends on FFT Size & the data width (no of bits) of the processor.

The performance of the proposed methods was evaluated by comparing the outputs with time domain convolution method. A MATLAB based code was developed for *Parallel Filtering* and *Mixed Filtering* approaches with filter taps, M=2048 and L=256 [13]. The cutoff frequencies of FIR filters were selected randomly for simulation. An audio signal of stereo source was provided as input to both methods. The outputs were recorded. The spectrum (Spectrum was plotted for FFT Size of 512) of the recorded outputs was shown in Fig 4. From the plot, it is understood that the outputs of proposed methods were matching with time domain convolution results and the difference could be negligible.

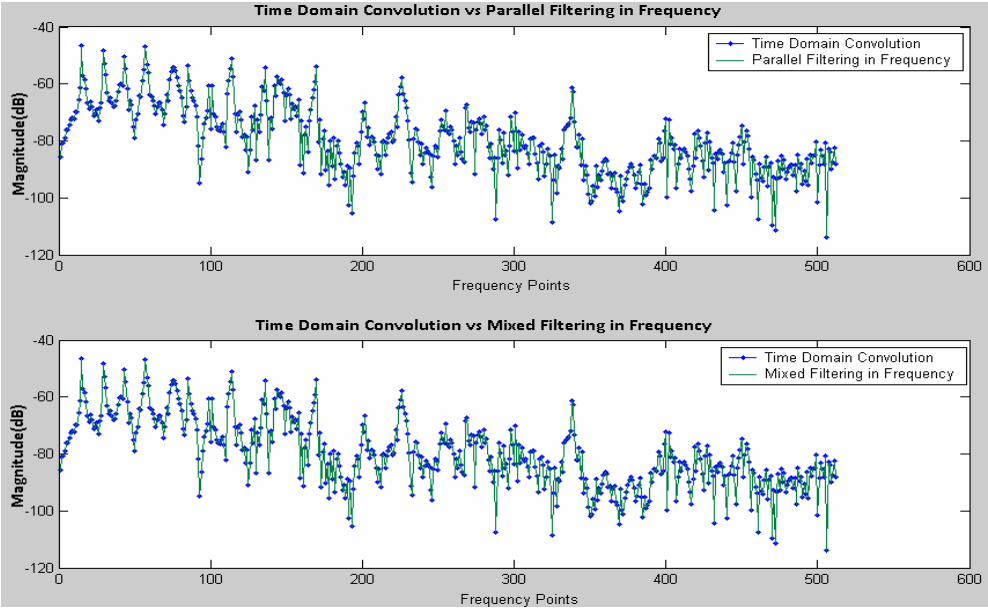

Fig.4. MATLAB based Performance Evaluation

4. Experimental Results

The proposed methods of *Parallel Filtering* and *Mixed Filtering* were implemented on Analog Devices SHARC EZ-Kit Lite with ADSP-21469, 32 bit floating point DSP processor [12] to measure the computational complexity. The time domain convolution & overlap save method for the individual filters were also implemented for comparison of computational complexity. For DFT evaluation, Radix-2 FFT decimation in time algorithm was used. Frequency multiplication process was implemented effectively using SIMD of SHARC processor. For IFFT calculation, FFT algorithm was reused by swapping the real & imaginary buffers and the scaling factor of 1/N was applied to the FFT outputs after swapping real & imaginary buffers again.

4.1. Variable Frame Size & Fixed filter taps

The computational complexity in terms of cycles was measured for all the approaches. A 48kHz frequency sweep audio signal was provided as input to the implemented codes. The outputs were observed for the random FIR cutoff values. The table & associated graphs given below contain cycle count for variable *L* (frame size) & fixed *M* (filter taps) values. Two cases are realized, one is for *M=2048* and one is for *M=1024*. From graphs, it is clear that *Mixed Filtering* provides less computational complexity. This is very useful in surround audio applications due to the usage of single IFFT after the frequency multiplication & summation. *Parallel Filtering*

provides good computational complexity than time domain based convolution & individual filtering with overlap save method. But this method requires more computations than *Mixed Filtering* approach. At lower values of *L* (frame size), individual filtering suffers from more computations when compared with time domain based convolution. This is due to the FFT size. Since FFT was used for DFT calculation, the DFT size, i.e. $N = L+M-1$, must be a power of 2. For example, if $M=1024$ & $L = 96$, then $N = 1024 + 96 - 1 = 1119$. Since *N* is not a power of 2, next higher power of 2 could be chosen as *N* i.e. $N = 2048$. In other way, whether *L* is either 96 or 256, $N = 2048$ provided that *L* is less than or equal to *M*. Due to this, FFT computational complexity is same for all *L*, which is less than or equal to *M*. Because of this, individual filtering approach with overlap save method is showing more computations at lower values of *L*. The notation for A, B, C & D is shown in the graphs.

Table 3. Mega Peak cycle count for M=2048 & M=1024.

M=2048				
Frame Size, L	**A**	**B**	**C**	**D**
96	0.2426	0.2835	0.2016	0.1557
112	0.2761	0.2837	0.2019	0.1559
128	0.3096	0.2840	0.2022	0.1561
160	0.3765	0.2845	0.2028	0.1564
192	0.4435	0.2850	0.2033	0.1568
224	0.5104	0.2855	0.2039	0.1571
256	0.5773	0.2860	0.2045	0.1575
320	0.7112	0.2870	0.2056	0.1582
384	0.8451	0.2880	0.2068	0.1589
448	0.9790	0.2891	0.2079	0.1596
512	1.1129	0.2900	0.2091	0.1603
640	1.3807	0.2920	0.2114	0.1617
768	1.6485	0.2942	0.2137	0.1631
896	1.9162	0.2962	0.2160	0.1645
1024	2.1758	0.2983	0.2183	0.1660

M=1024				
Frame Size, L	**A**	**B**	**C**	**D**
96	0.1230	0.1358	0.0968	0.0753
112	0.1400	0.1361	0.0971	0.0754
128	0.1569	0.1363	0.0973	0.0756
160	0.1908	0.1368	0.0979	0.076
192	0.2247	0.1374	0.0985	0.0763
224	0.2587	0.1379	0.0991	0.0767
256	0.2926	0.1384	0.0996	0.0770
320	0.3604	0.1394	0.1008	0.0777
384	0.4283	0.1404	0.1019	0.0784
448	0.4961	0.1415	0.1031	0.0791
512	0.5639	0.1425	0.1043	0.0798
640	0.6996	0.1445	0.1066	0.0813
768	0.8353	0.1466	0.1089	0.0827

4.2. Comparison of Computational Savings

The Mega peak cycle counts provided in sub-section 4.1 are measured with SHARC 21467, 400MHz processor with an audio input of 48kHz Sampling frequency. The total MIPS (Million Instructions Per Second) consumed

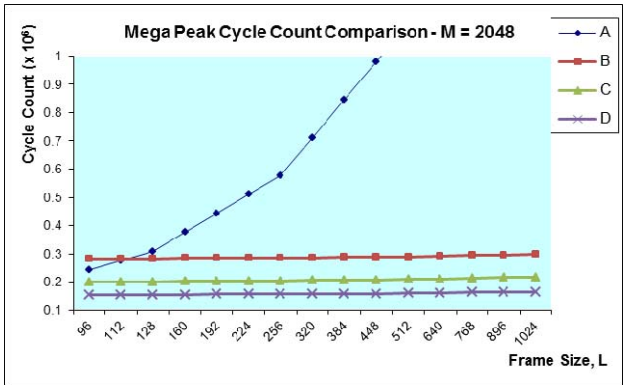

Fig.5. Mega Peak Cycles Comparison for filter taps, M=2048 & variable frame size, L

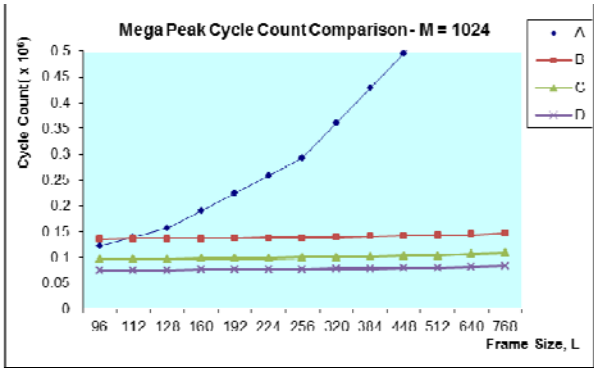

Fig.6. Mega Peak Cycles Comparison for filter taps, M=1024 & variable frame size, L

by each method are evaluated and from this, the percentage of remaining MIPS is calculated as computational savings. Fig 7 provides the comparison of computational savings for all methods.

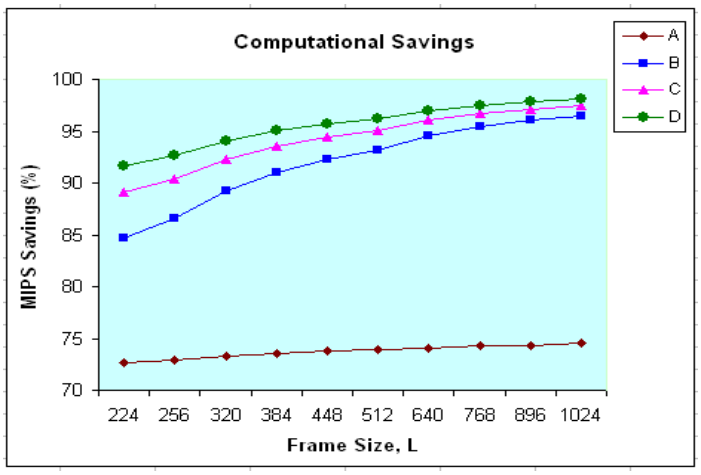

Fig.7. Comparison of Computational Savings

5. Conclusion

In this paper, the performance of (a) *Parallel Filtering* (b) *Mixed Filtering* was measured on MATLAB platform & compared with that of time domain convolution. The computational complexity was also compared based on measurements of ADSP-21469 EZ-Kit Lite SHARC DSP processor. The results show that proposed methods provide great computational savings and the performance deviation is insignificant. By forming a complex sequence with two real outputs of CTC and working towards the fast implementation leads to great savings in computational complexity. Since frequency domain based filtering contains more FFT calculations, some more optimization is still achievable if Radix 4 or Split Radix FFTs were used in implementation..

References

[1] John G. Proakis & Dimitris G. Manolakis, *Digital Signal Processing Principles, Algorithms and Applications, 3rd edition*, Page nos. 430 to 476

[2] W. G. Gardner and K. D. Martin, "HRTF measurements of a KEMAR", Journal of the Acoustical Society of America, vol. 97, no. 6, pp. 39073908, 1995

[3] M. Otani and S. Ise, Fast calculation system specialized for head-related transfer function based on boundary element method, Journal of the Acoustical Society of America, vol. 119, no. 5, pp. 25892598, 2006.

[4] Kirkeby Ole, Rubak Per, Johansen Lars G., Nelson and Philip A, Implementation of Cross-Talk Cancellation Networks Using Warped FIR Filters, AES paper Number: 16-031, March 1999

[5] Lentz Tobias and Schmitz Oliver, Adaptive Cross-talk Cancellation System for a moving listener, AES paper Number: 000134, June 2002

[6] LinWang, Fuliang Yin and Zhe Chen, A Stereo Crosstalk Cancellation System Based on the Common-Acoustical Pole/Zero Model, Audio Engineering Society, August 2010

[7] Lee Junho, Park Young-Cheol, Youn Dae-Hee, Robust Crosstalk Cancellation Based on Energy- Based Control, AES 34, August 2008

[8] 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

[9] Jeong Jae-woong, Kim Jeong-Tae, Lee Junho, Park Young-cheol and Youn Dae-hee, Design and Implementation of IIR Crosstalk Cancellation Filters Approximating Frequency Warping, AES 118, May 2005

[10] Behler Gottfried and Lentz Tobias, Dynamic Crosstalk Cancellation for Binaural Synthesis in Virtual Reality Environments, AES 117, October 2004

[11] Bettarelli Ferruccio, Cecchi Stefania, Palestini Lorenzo, Peretti Paolo and Piazza Francesco, Sub band Adaptive Crosstalk Cancellation: A Novel Approach for Immersive Audio, AES 124, May 2008

[12] *ADSP-214xx SHARC Processor Hardware Reference* Revision 0.3, July 27, 2010, Part Number 82-000469-01.

[13] *MATLAB, The Language of Technical Computing*, Version 7.9.0.529 (R2009b) 32-bit (win32), August 12, 2009.

[14] Mark Borgerding, Turning Overlap-Save into a Multiband Mixing, Downsampling Filter Bank, IEEE SIGNAL PROCESSING MAGAZINE, March, 2006