Real-time DSP Implementation of Audio Crosstalk Cancellation Using Mixed Uniform Partitioned Convolution

Chunduri SreenivasaRao *chsrinivas19800305@rediffmail.com*
Research Scholar, ECE Department,
KL University
Vijayawada, 522502, INDIA

NVK Mahalakshmi *mahalakshminvk.nvk@gmail.com*
Assistant Professor, ECE Department
SRK Institute of Technology,
Vijayawada, 521108, INDIA

Dr. Dhulipalla VenkataRao *dvenky221101@rediffmail.com*
Principal, Narasaraopet Instt. of Tech.
Narasaraopet, Guntur, 522601, INDIA

Abstract

For high fidelity sound reproduction, it is necessary to use long filter coefficients in audio crosstalk cancellation. To implement these long filters on real-time DSP processors, conventional overlap save technique suffers from more computational power as well as processing delay. To overcome these technical problems, *mixed uniform partitioned convolution* technique is proposed. This method is derived by combining uniform partitioned convolution with mixed filtering technique. With the proposed method, it is possible to perform audio crosstalk cancellation even at the order of ten thousand filter taps with less computations and short processing delay. The proposed technique was implemented on 32-bit floating point DSP processor and design was provided with efficient memory management to achieve optimization in computational complexity. The computational comparison of this method with conventional methods shows that the proposed technique is very efficient for long filters.

Keywords: Convolution, Crosstalk Cancellation, FFT, Mixed filtering, Partitioned Convolution, Overlap Save Method.

1. INTRODUCTION

3D audio systems have the potential to be used in many spatial audio applications such as home theatre entertainment, gaming, teleconference and remote control. To reproduce the realistic spatial audio, the challenging task of any 3D audio system is to have the ability to reproduce spatial reverberation characteristics and spatial audio pattern at the desired locations. This could be achieved by binaural synthesis and audio cross-talk cancellation (CTC). In 1983, head related transfer function (HRTF) technology was developed to transform the sound field of a particular location to the head by convolving the sound with appropriate pair of HRTF functions. Headphones have excellent spatial characteristics such as channel separation and equalization, but they are inconvenient and little bit cumbersome to use when more number of listeners is enjoying the audio. An alternative to HRTF technology is conventional stereo loudspeaker system located exactly in front of the listener. In this case, transmission path equalization is obtained by inverting acoustic transfer function matrix between the two loudspeakers and the two ears of listener, which is called crosstalk cancellation and is particularly required to cancel the unwanted crosstalk from each speaker to the opposite ear [1][2][3][4].

To obtain such equalization for transmission path, the impulse responses of 2x2 system inversion matrix ($h_{aa}(n)$, $h_{ab}(n)$, $h_{ba}(n)$, $h_{bb}(n)$ as shown in Fig.1) may last for several hundreds of milliseconds, which leads to the requirement of thousands of FIR filter coefficients as impulse responses [5]. Due to this, the implementation of these long filters on real-time DSP processors requires more computational power. To overcome the complexity issues, it is essential to develop new implementation techniques without compromising for performance.

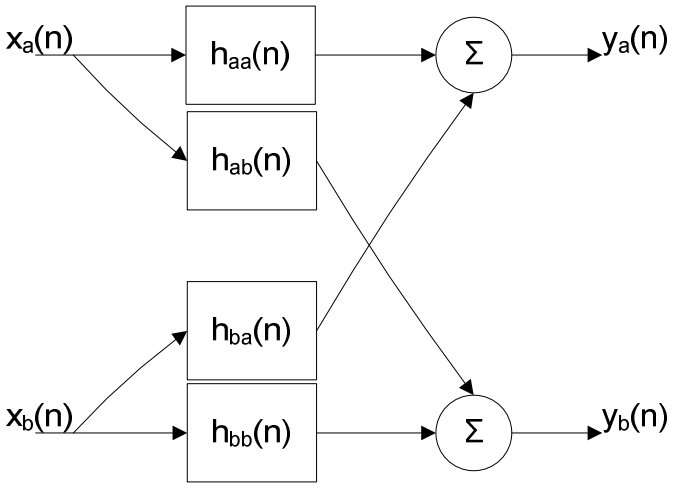

FIGURE 1: Audio Crosstalk Cancellation (CTC) for stereo source.

Historically, time domain convolution is well known technique. Even though this method is the original method, it won't be preferred for long filters, in general, as it suffers from more computational power. On other hand, overlap save & overlap add methods are frequency domain methods and are efficient to handle the computational complexity problems in real-time implementation. In these methods, the length of FFT is derived as N = L+M-1, which must be a power of 2 due to the usage of FFT, where L and M are frame size and filter length respectively. For the case of L=256 & M=8192, N becomes 8447 and could be chosen as 16384 by adding additional zeros. Due to additional zeros, FFT size increases and hence, the increase in computational power. In addition to this, the additional zeros cause the delay in output response at least by M [5][6][7].

To overcome delay issues, in 1988, Vetterli proposed running convolution based on multi-rate methods, in which impulse response is divided into bi-orthonormal filter banks continuously till minimum FFT Size reaches such as equivalent to frame size, L. After bi-orthonormal filtering process, required interpolation techniques will be applied to obtain the final filtered signal. In this, delay issue was solved but it suffers from computational complexity as filtering process could be performed for every sub filter bank and this technique involves more buffering of data[8][9].

Uniform partitioned convolution is a kind of technique where computational complexity as well as delay issues is resolved. 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 [10][11][12][13][14][15][16][17].

To avoid such problems, uniform partitioned convolution is combined with mixed filtering 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 for long filters.

This paper is organized as follows. Section 2 provides the review of mixed filtering and uniform partitioned convolution. Later the combination of these two techniques is explained as proposed

method in section 3. It also discusses theoretical computational complexity, design to achieve efficient memory management and 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.

2. REVIEW OF PREVIOUS WORK

In this section, Mixed Filtering and uniform partitioned convolution methods are reviewed.

2.1 Mixed Filtering

The name implies that this method is able to perform all filtering operations of CTC in a single equation. This is possible by forming a complex sequence with real-time outputs $y_a(n)$, $y_b(n)$. By doing so, one could arrive at the frequency domain equivalent of output complex sequence as

$$Y(k) = Y_a(k) + jY_b(k) = X_a(k)H_a(k) + X_b(k)H_b(k) \quad \rightarrow \quad (1)$$

where

$$H_a(k) = H_{aa}(k) + j\,H_{ab}(k)$$
$$H_b(k) = H_{ba}(k) + j\,H_{bb}(k)$$

The computational complexity of equation (1) includes one FFT computation with decomposition (to evaluate $X_a(k)$ and $X_b(k)$), complex frequency multiplication and one IFFT. The real and imaginary components of IFFT output yield $y_a(n)$ and $y_b(n)$ respectively [5].

2.2 Uniform Partitioned Convolution

In this method, the length of impulse response is uniformly partitioned into small lengths so that overlap save method is applied to each partitioned impulse response and finally adding all outputs of partitioned filters yield the convolved output. Fig. 2 shows the signal processing involved in this method [10][11].

Let $h(n)$ of length M be the impulse response and frame length be L. Fig. 2A shows the time delay line filter. Let $h_0(n), h_1(n),..., h_{m-1}(n)$ where $m = M/L$ be the partitioned impulse responses, which are obtained by dividing the impulse response length by L so that the length of each partitioned impulse response becomes L. Fig. 2B shows the application of overlap save method to each partitioned impulse response, where FFT/IFFT size is equal to 2L. After first frame is processed, L samples of IFFT output are transmitted as filtered output. For 2nd frame, first frame will be delayed by L samples and provided as input to 2nd partitioned filter. Now overlap save method is applied to both partitioned filters and IFFT outputs are summed to yield filtered output of 2nd frame. This process is continued till last partitioned filtering process. Instead of finding FFT and IFFT for time delayed frames and frequency multiplied outputs, it is better to optimize the structure with single FFT and IFFT as shown in Fig. 2C just by delaying the FFT outputs. When 2nd frame arrives, FFT output of 1st frame becomes the input 2nd partitioned filter. The complex outputs of all partitioned frequency multipliers are added and a single IFFT is applied to the complex sum. From the IFFT output, L samples are transmitted as overall filtered output.

FFT size of 2L means that the appended zero samples are L in size so that the processing delay is L samples in worst case whereas overlap save method for original impulse response produces at least M samples delay. Hence this method provides less delay compared to that of overlap save method.

Computational complexity of this method is explained as follows. 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.

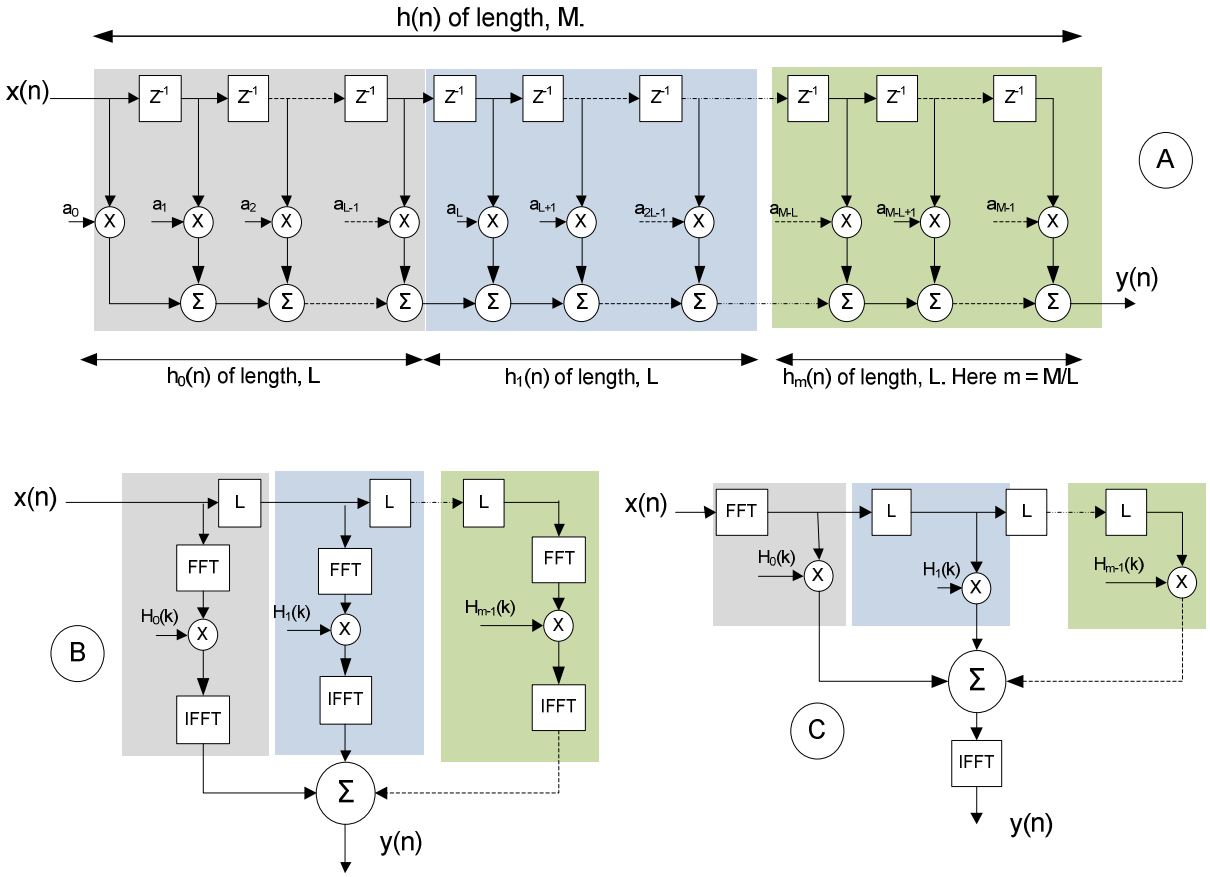

FIGURE 2: Interpretation of Uniform Partitioned Convolution using block diagram representation.

3. PROPOSED ALGORITHM

The proposed algorithm combines both the methods mentioned in Section 2. To proceed for the proposed algorithm, let us partition the impulse responses, $h_a(n)$ & $h_b(n)$ of equation (1). The time domain equivalents of $H_a(k)$ & $H_b(k)$ are given by $h_{aa}(n) + j\, h_{ab}(n)$ & $h_{ba}(n)+j\, h_{bb}(n)$ respectively. The length of these impulse responses are M. By partitioning these into m parts, where m = M/L, the resultant partitioned impulse responses are given by

$$\begin{aligned} h_a(n) &= \{h_{a,0}(n),\ h_{a,1}(n),\ h_{a,2}(n),\ldots,\ h_{a,m-1}(n)\} \\ &= \{h_{aa,0}(n)+ j\, h_{ab,0}(n),\ h_{aa,1}(n)+ j\, h_{ab,1}(n),\ \ldots,\ h_{aa,m-1}(n)+ j\, h_{ab,m-1}(n)\} \end{aligned}$$

$$\begin{aligned} h_b(n) &= \{h_{b,0}(n),\ h_{b,1}(n),\ h_{b,2}(n),\ldots,\ h_{b,m-1}(n)\} \\ &= \{h_{ba,0}(n)+ j\, h_{bb,0}(n),\ h_{ba,1}(n)+ j\, h_{bb,1}(n),\ \ldots,\ h_{ba,m-1}(n)+ j\, h_{bb,m-1}(n)\} \end{aligned}$$

The length of each partitioned sequence now becomes L. As per overlap save method, FFTs of these partitioned responses found out by appending L zeros to each impulse response. The frequency equivalents, $H_{a,0}(k)$, $H_{a,1}(k)$,... $H_{a,m-1}(k)$ and $H_{b,0}(k)$, $H_{b,1}(k)$,... $H_{b,m-1}(k)$ are obtained in this way. Once partitioned FFT coefficients are found, the rest of the algorithm is based on the application of uniform partitioned convolution approach as per equation (1). Instead of using two IFFTs, it is better to apply single IFFT to the complex frequency sum, $[X_a(k)H_a(k)+X_b(k)H_b(k)]$. IFFT output provides the outputs $y_a(n)$ & $y_b(n)$ in complex form.

The z-domain equivalent of equation (1) is given by

$$
\begin{aligned}
Y(z) &= Y_a(z) + j\,Y_b(z) = X_a(z)H_a(z) + X_b(z)H_b(z) \\
&= X_a(z)\left[H_{a,0}(z) + z^{-L}H_{a,1}(z) + \ldots + z^{-(m-1)L}H_{a,m-1}(z)\right] + \\
&\qquad X_b(z)\left[H_{b,0}(z) + z^{-L}H_{b,1}(z) + \ldots + z^{-(m-1)L}H_{b,m-1}(z)\right] \\
&= X_a(z)\sum_{i=0}^{m-1} z^{-iL}H_{a,i}(z) + X_b(z)\sum_{i=0}^{m-1} z^{-iL}H_{b,i}(z) \\
&= \sum_{i=0}^{m-1}\left[X_a(z)z^{-iL}\right]H_{a,i}(z) + \left[X_b(z)z^{-iL}\right]H_{b,i}(z) \qquad \rightarrow (2)
\end{aligned}
$$

$$
y(n) = y_a(n) + j\,y_b(n) = z^{-1}\left\{\sum_{i=0}^{m-1}\left[X_a(z)z^{-iL}\right]H_{a,i}(z) + \left[X_b(z)z^{-iL}\right]H_{b,i}(z)\right\} \qquad \rightarrow (3)
$$

The block diagram of the proposed algorithm was shown in Fig. 3. The steps involved in proposed algorithm are as follows.

a. Partition the long impulse responses and find the complex frequency equivalents as stated above.
b. Receive the 1st frames of inputs $x_a(n)$ & $x_b(n)$ and store them in memory buffers of length 2L each.
   Note: Initially memory buffers contain zeros and filling of input frames into memory buffers is based on overlap save method.
c. Find out FFT of overlapped 1st frame and store complex FFT outputs, $X_a(k)$ & $X_b(k)$ in separate buffers.
   Note: Reference [7] could be followed to find $X_a(k)$ & $X_b(k)$.
d. Perform complex frequency multiplication between frequency partitioned coefficients and frequency delayed input frames. Each time one frequency multiplication is performed, the resultant complex output is added to previous multiplier output so that complex sum will be provides as input to IFFT.
e. Evaluate step (d) for both of inputs $x_a(n)$ & $x_b(n)$ and add the corresponding complex sums to yield $[X_a(k)H_a(k)+X_b(k)H_b(k)]$.
f. Now apply IFFT to the output in step (e) and transmit real & imaginary parts of complex IFFT output as $y_a(n)$ & $y_b(n)$ respectively.
   Note: As per overlap save method, only L valid samples will transmitted from IFFT output.
g. Repeat steps (b) to (f) for each new frame.

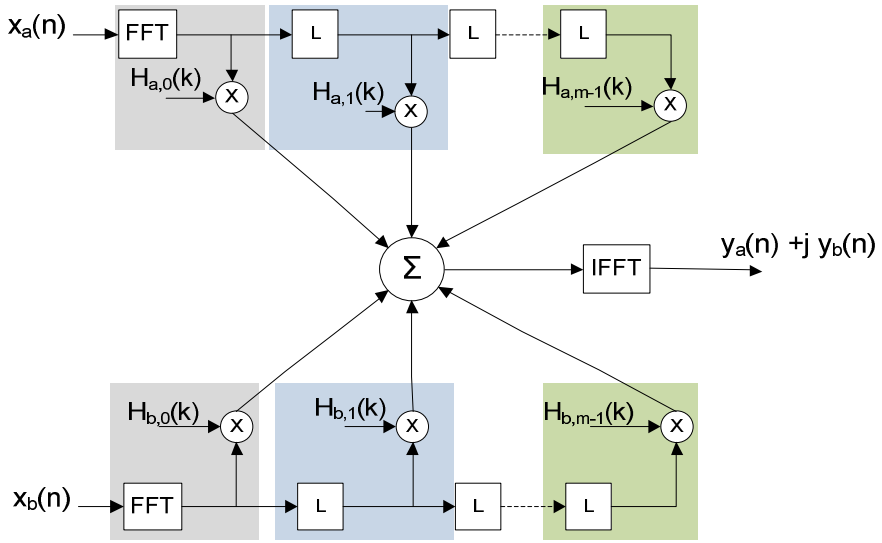

FIGURE 3: Block diagram of Mixed Uniform Partitioned Convolution to obtain CTC outputs

3.1 Theoretical Computational Complexity

The following table provides the details of computations required for proposed algorithm. Here $O(N)=N.\log_2 N$

To Calculate	Computational complexity		Remarks
	Complex Multiplications	Complex Additions	
$X_a(k)$ & $X_b(k)$.	0.5 O(2L)	O(2L)+2. 2L = O(2L) + 4L	FFTs of $x_a(n)$ & $x_b(n)$ could be found with single FFT and decomposition[7]
$X_a(k)H_a(k)$	2L.M/L=2M	2(M-L)	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
$X_b(k)H_b(k)$	2L.M/L=2M	2(M-L)	"
$Y(k)$	-	2L	The outputs of $X_a(k)H_a(k)$ & $X_b(k)H_b(k)$ are complex sequences are of length 2L each. As per equation (1), the addition of these outputs index by index requires 2L complex additions
$y(n)$	0.5 O(2L)	O(2L)	Complexity of IFFT with size equal to 2L

TABLE 1: Computational complexity of proposed algorithm

Total Complex Multiplications : 4M + O(2L)
Total Complex Additions : 4M + 2 O(2L) + 2L

3.2 Efficient Memory Management

After going through the steps of the proposed algorithm, one can believe that the proposed method requires more copying routines (and hence more processing cycles) to perform FFT, frequency multiplication and IFFT evaluation. But by storing FFT output buffers in systematic way, these copying routines could be avoided. Fig. 4 depicts the approach that was followed for implementation. The diagram is showing only the real buffers of FFT. Such kind of buffers is needed for imaginary buffers also, which was not shown in the diagram.

A dedicated memory of size 2M is allocated for real & imaginary parts of $X_a(k)$ & $X_b(k)$ to store FFT values of current frame as well as delayed frames. Also the real & imaginary parts of partitioned coefficients are also arranged in memory buffers of size 2M in sequence i.e. $H_{a,0}(k)$, $H_{a,1}(k),...H_{a,m-1}(k)$, etc. Initially all input buffers hold zeros. When 1st frame arrives, FFT of input buffers could be stored in last 2L locations of the dedicated 2M length buffer directly. The pointer for the last 2L location in dedicated 2M buffer could be passed as argument into FFT function evaluation. Now, these 2L values are multiplied directly with first 2L values of coefficient buffer index by index. The resultant 2L values could be added to another 2L length dedicated real & imaginary buffers. These dedicated buffers hold the final complex frequency multiplication output of size 2L and will be directly provided as input to IFFT evaluation. The next 2L locations of real & imaginary buffers are accessed in circular fashion and multiplied with associated coefficient buffers index by index and finally added to the real & imaginary dedicated IFFT input buffers. This process is continued till m = M/L stages of multiplication are performed. IFFT is evaluated with 2L size dedicated real & imaginary buffers. IFFT output produces 2L complex values of which L real & imaginary values are transmitted as $y_a(n)$ & $y_b(n)$ respectively.

When 2nd frame arrives, the FFT values of 1st frame need not be disturbed. FFT of 2nd frame will be stored in 2L locations previous to 1st frame. The buffers are accessed in circular fashion and complex frequency multiplication could be performed in same way as explained earlier with the associated partitioned complex frequency coefficients. This procedure will be continued in this way for every new frame received.

By assuming real & imaginary buffers as circular buffers, intermediate copying routines could be minimized for complex frequency multiplication as well as FFT and IFFT evaluation.

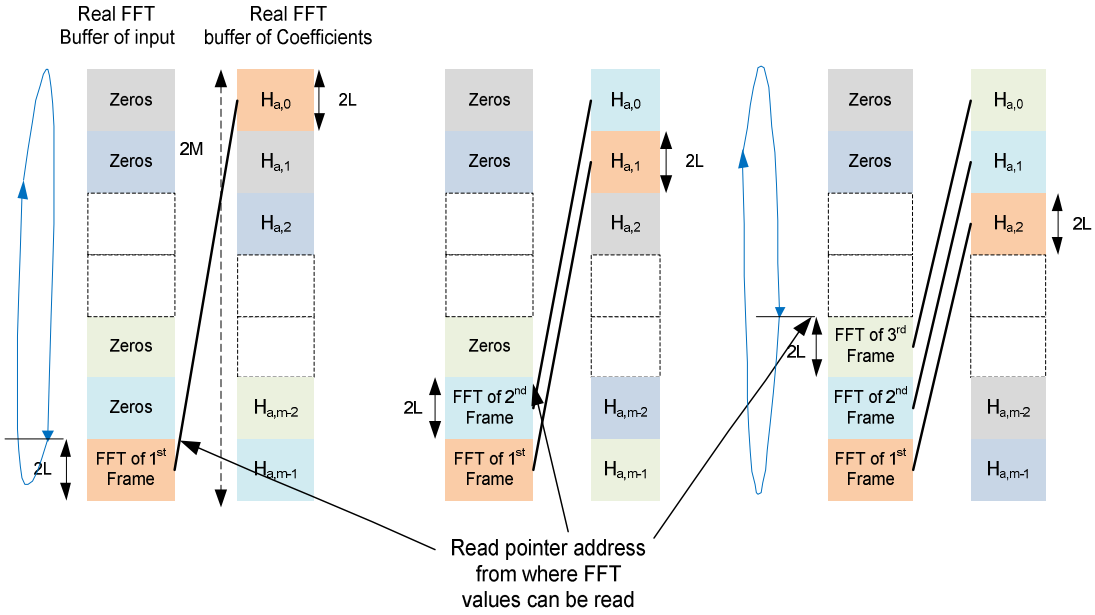

FIGURE 4: Efficient memory management in SHARC processors for Mixed Uniform Partitioned Convolution.

3.3 Efficient Complex Frequency Multiplication

The sum of complex multiplications, i.e. $[X_a(k)H_a(k)+X_b(k)H_b(k)]$, was implemented with multiplication and add instructions in parallel with data move operations efficiently on SHARC processor. This piece of code was provided here with 8 instructions inside the loop. The loop counter size is 2L, which means that this code is valid for one partitioned filter and the same code is called m times for all partitioned filters. The following analogy was made for easy understanding.

$X_a(k)$ → a1+jb1
$H_a(k)$ → c1+jd1
$X_b(k)$ → a2+jb2
$H_b(k)$ → c2+jd2

```
/*********************************************

(a1 + j b1)(c1 + j d1) + (a2 + j b2)(c2 + j d2)  + (y1 + j y2)

=   ( a1c1 - b1d1 + a2c2 - b2d2 + y1 ) + j (  a1d1 + b1c1 + a2d2 + b2c2 + y2 )

*********************************************/

                              f0 = dm(i0,m0), f4 = pm(i8,m8);     // Read a1, c1
f8 = f0*f4,                   f1 = dm(i1,m0), f5 = pm(i9,m8);     // a1c1, Read   b1, d1
f12= f1*f5,                   f2 = dm(i2,m0), f6 = pm(i10,m8);    // b1d1, Read   a2, c2
f13= f2*f6,     f8 = f8-f12,  f3 = dm(i3,m0), f7 = pm(i11,m8);    // a2c2, a1c1-b1d1, Read    b2, d2

lcntr = FFTSIZE/2, do Mul_Add until lce;              // SIMD mode was set. Hence loop count is FFT_Size/2

    f12= f3*f7, f8 = f8+f13,    f14=dm(i5,m5);                    // b2d2, a1c1-b1d1+a2c2 , Read Y_Real
    f9 = f0*f5, f8 = f8-f12,    f15=pm(i13,m13);                  // a1d1, a1c1-b1d1+a2c2-b2d2 , Read y_imag
    f12= f1*f4, f14= f8+f14,    f0 = dm(i0,m0), f4 = pm(i8,m8);   // b1c1, a1c1-b1d1+a2c2-b2d2+ Y_Real
    f13= f2*f7, f9 = f9+f12,    dm(i5,m0)=f14;                    // a2d2, a1d1+b1c1, Write Y_Real
    f12= f3*f6, f9 = f9+f13,    f1 = dm(i1,m0), f5 = pm(i9,m8);   // b2c2, a1d1+b1c1+a2d2
    f8 = f0*f4, f9 = f9+f12,    f2 = dm(i2,m0), f6 = pm(i10,m8);  // a1d1+b1c1+a2d2+b2c2
    f12= f1*f5, f15 = f9+f15,   f3 = dm(i3,m0), f7 = pm(i11,m8);  // a1d1+b1c1+a2d2+b2c2+Y_imag
Mul_Add: f13= f2*f6,    f8 = f8-f12,    pm(i13,m8)=f15;           // Write Y_imag
```

In above piece of code, y_1 & y_2 are of size 2L each, which hold the real and imaginary outputs obtained by summing the complex frequency multiplication outputs of all partitioned filters. These two buffers are provided as inputs to IFFT evaluation.

4. EXPERIMENTAL RESULTS & DISCUSSION

The proposed method "Mixed Uniform Partitioned Convolution" was implemented on SHARC ADSP-21469, 32 bit floating point DSP processor [18] with EZ-Kit Lite to measure the computational complexity. A dedicated buffer of size 2L is used for each input frames $x_a(n)$ & $x_b(n)$. Overlap save method is followed to overlap previous frames for calculation of FFT for each frame. Fixed memory buffers of size 2L are allocated to store real & imaginary FFT coefficients of all partitioned impulse responses. The complex frequency multiplication of equation (2) was implemented very efficiently 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.

Filter Length, M	Mega Peak Cycle count	Filter Length, M	Mega Peak Cycle count
512	0.02137	9216	0.19573
1024	0.02628	9728	0.20579
1536	0.0312	10240	0.21565
2048	0.03611	10752	0.22561
2560	0.04103	11264	0.23762
3072	0.04594	11776	0.24698
3584	0.05086	12288	0.25532
4096	0.05577	12800	0.26490
4608	0.06069	13312	0.27398
5120	0.0656	13824	0.28536
5632	0.07052	14336	0.29621
6144	0.07543	14848	0.30453
6656	0.08035	15360	0.31762
7168	0.08526	15616	0.32203
7680	0.09018	16128	0.33018
8192	0.09509	16384	0.33987
8704	0.18577		

TABLE 2: Proposed Algorithm - Mega Peak Cycle counts for frame length, L=256 and variable filter lengths.

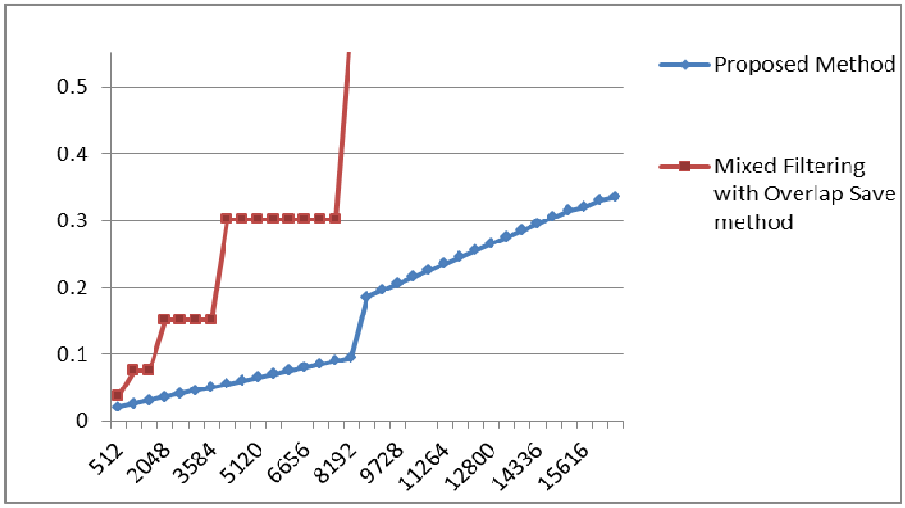

FIGURE 5: Computational complexity comparison. X-axis represents filter length, M. Y-axis represents Mega Peak cycle count. The above results are valid for frame length of L=256.

The cycle counts were calculated on SHARC processor by varying the filter length for fixed block size, L=256. Table 2 shows the computational complexity details along with that of Mixed filtering with overlap save method. Fig. 5 shows the comparison of computation cycles between Mixed filtering with overlap save method (Results were taken from Reference [7] for this method) and proposed method.

In the graph, one can observe that slight increase in mega peak cycle count for filter length of M=8704. For the cases of more than M=8704, the required internal DSP memory is not able to hold all the required buffers such as delayed FFT values of input as well as FFT coefficients. To handle this, all the contents of these buffers will be written into external storage device such as SDRAM. From SDRAM, these buffers could be read whenever needed using Direct Memory Access. To avoid too many cycles due to this memory transfer, DMA was performed in background with DSP core process. Due to all these processing, some extra cycles are needed compared to the actual signal processing and hence the mega peak cycle count was increased for these cases.

Similarly, mega peak cycle count is constant in Mixed filtering with overlap save method for few of the cases, for example M=2048 to 3584. This is obviously expected because in all these cases, FFT length is 4096 and all the operations are based on this factor. This is where the proposed method is advantageous than the overlap save method.

The experimental results clearly indicate that the proposed method provides more savings in computational complexity by following the efficient design as explained sub-sections 3.2 and 3.3. The advantage of proposed algorithm is that FFT computational complexity is a function of frame length unlike on coefficient length as in overlap save method. By segmenting filtering process into M/L parts, one can achieve attractive computational savings and also savings in memory usage.

5. CONCLUSION & SCOPE OF FUTURE WORK

To reduce the processing delay for long filters in audio crosstalk cancellation, an efficient method with combination of mixed filtering and uniform partitioned convolution is proposed to implement on real-time DSP processors. With efficient internal DSP memory management, it is possible to perform audio CTC with less computational complexity even at longer filter lengths. The results clearly indicate that the proposed method is highly dominant in both terms of computational complexity and processing delay.

This work could be extended to mixed non-uniform partitioned convolution, with which, one can obtain more efficient results. But this method is basically useful in applications related to operating systems and involves more complex process. Also instead of concentrating on FFT, it is better to use Fast Hartley Transform to reduce the computational complexity. Because FHT is always real in nature and operates directly on real signal as opposed to FFT, which always operates on complex signal, in general. The major area where more computations needed in the proposed method is complex frequency multiplication. By either utilizing the DSP architecture core instruction set effectively or simple mathematical equations, it is possible to work on this for effective computational complexity.

6. REFERENCES

[1] 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, pp 2589-2598

[2] Kirkeby ole, Rubak Per, Nelson Philip A. and Farina Angelo, “Design of Crosstalk cancellation Networks by using Fast deconvolution” in AES 15, May 1999, pp 9900-9905

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

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

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

[6] John G. Proakis and Dimitris G. Manolakis, "Digital Signal Processing Principles, Algorithms and Applications", 3rd Edition, Page No. 430 to 476

[7] Richard G Lyons, "Understanding Digital Signal Processing", 3rd Edition, published on November 11, 2010.

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

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

[10] Eric Battenbaerg and Rimas Avizienis. "Implementing Real-time Partitioned Convolution Algorithms on Conventional Operating Systems", Proc. of 14th Int. Conference on Digital Audio Effects, Paris, France, Sept 19-23, 2001.

[11] Anders Torger and Angelo Farina, " Real-time Partitioned Convolution for Ambiophonic Surround Sound", IEEE Workshop on applications of Digital Signal Processing to Audio and Acoustics 2001, New Paltz, New York, W2001-4.

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

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

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

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

[16] E. Armelloni, C. Giottoli and A. Farina, "Implementation of Real-time partitioned convolution on a DSP board", IEEE Workshop on Applications of Signal processing to Audio and Acoustics, October 19-22, 2003, New Paltz, NY.

[17] Eric Battenberg, David Wessel & Juan Colmenares, "Advances in the Parallelization of Music and Audio Applications", ebookbrowse.com/wessel-parlab-retreat-winter-2010-ppt-d59199484

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