ISSN (Online) : 0974-5645
ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 8(5), 419–431, March 2015* DOI : 10.17485/ijst/2015/v8i5/60708

Design and Implementation Analysis of OSD based Audio Crosstalk Cancellation with Multi-channel Inputs on DSP Processors

Ch. Sreenivasa Rao[1*], D. Venkata Rao[2] and S. Lakshminarayana[3]

[1]Department of ECE, KL University, India; chsrinivas19800305@rediffmail.com
[2]NIT, Narsaraopet, Guntur (Dist.), India; dvenky221101@rediffmail.com
[3]Department of ECE, KL University, India; drslakshminarayana@gmail.com

Abstract

To overcome the basic system inversion problems in conventional crosstalk cancellation systems, the three channel Optimum Source Distribution (OSD) was developed by Takashi et al., that provides balance between in-phase and out of phase components. The main difficulty in this system is real-time implementation of cross talk cancellation filters on DSP platforms, particularly when filter lengths are very long with system operates on multichannel inputs. This paper discusses the implementation complexities and proposes an efficient approach to reduce the power consumption. It also discusses efficient usage of available processor memory. By utilizing the processor resources and existing frequency domain techniques, this approach would be one of best methods for long filters. The mathematical model, with efficient design for floating point DSP processors, clearly explains the optimization methods at algorithmic and instruction levels. The computational complexity of the proposed method was measured for various multi-channel input sources and the comparison was shown. The results indicate that the proposed method provides efficient computations than existing methods.

Keywords: Convolution, Crosstalk Cancellation, Optimal Source Distribution, Overlap Save Method, Uniform Partitioned Convolution

1. Introduction

3D audio CrossTalk Cancellation (CTC) systems are used in many spatial audio applications such as home theatre entertainment, gaming, teleconference and remote control. With the aim of reproducing spatial reverberation characteristics and spatial audio pattern at the listeners' ears, this technology has tremendous changes over the years such as binaural synthesis using headphone processing and conventional loudspeaker systems. The process of system inversion of audio room simulation and cancellation of sounds at listeners' ears using conventional loudspeaker system, usually called crosstalk cancellation, suffer from loss of dynamic range, more severe non-linear distortion due to spectral coloration and low signal to noise ratio[1–4].

Takeuchi et al., developed a smart solution to overcome these problems with the concept of Optimum Source Distribution (OSD) by forming a relationship between the position of loudspeaker and the operating frequency. The audio bandwidth is divided into three frequency bands such as low pass, band pass and high pass frequency regions and a frequency divider network was introduced after CTC block[5–10]. The two channel OSD systems use this relationship to balance the in-phase and out of phase components of acoustic inversion matrix. Inverse filters of OSD have unique property that simple phase change is required for perfect crosstalk cancellation. This advantage is further enhanced by introducing one more loudspeaker between the binaural loudspeakers, which is referred to as three channel OSD system[1–4]. Figure 1 shows the audio CTC and frequency divider network with multi-channel

*Author for correspondence

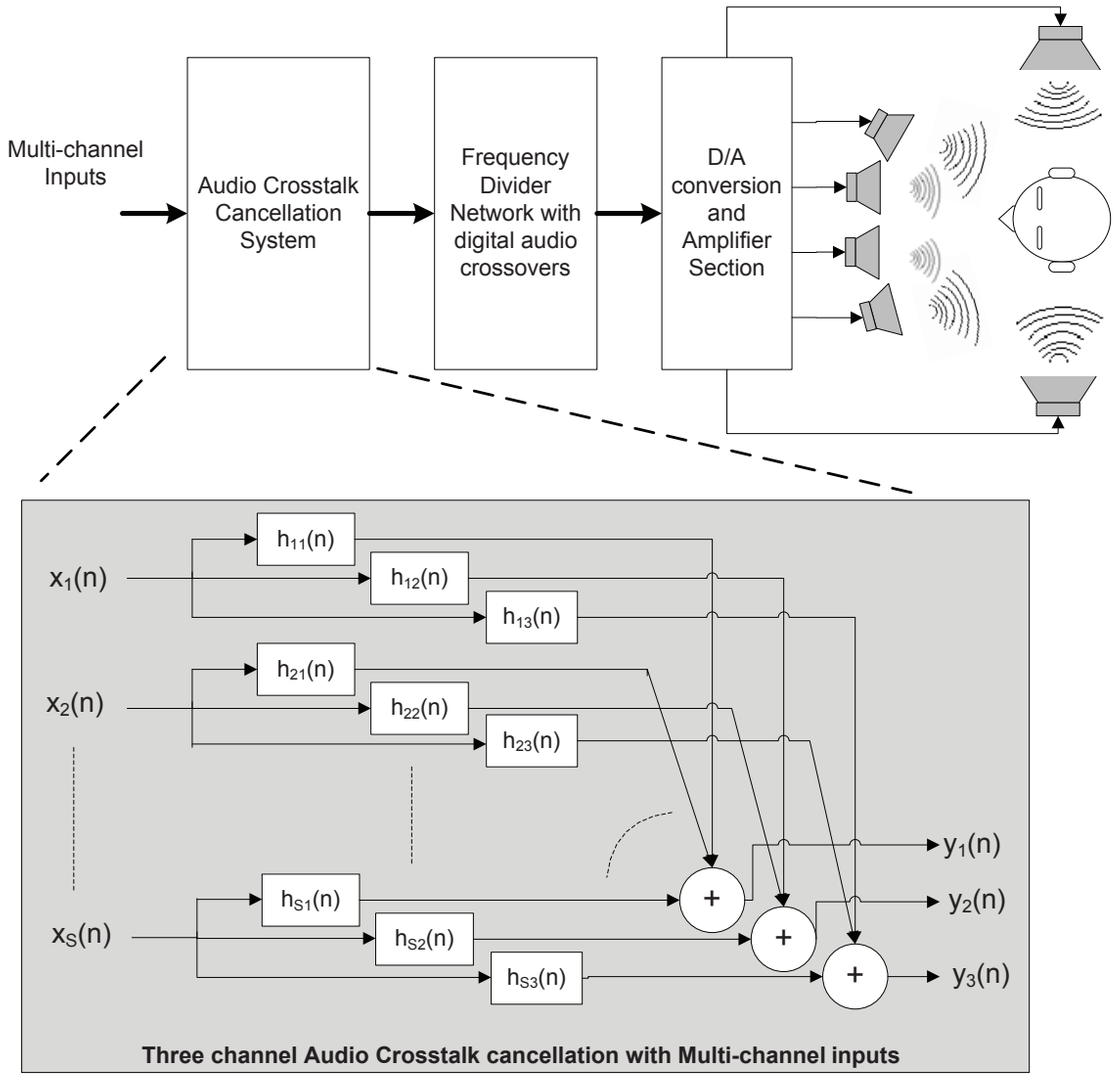

Figure 1. Three channel OSD system with multi-channel input source. Here S is number of multi-channel inputs.

inputs. Here the word "three channel" means that each input source is processed by three CTC filters.

To produce the spatial reverberation characteristics and sound patterns at the listener's ears, audio CTC requires long filters. This paper mainly focuses on the real-time implementation complexity of audio CTC in OSD system for multi-channel inputs. The basic challenge is to develop a feasible low cost implementation method to implement all long filters in audio CTC system with reasonable power consumption and memory on a single DSP.

The original approach is time domain based filtering that requires more computational power, which is not suitable for real-time implementation of long filters. The recent DSP processors (such as ADI SHARC 214xx) are manufactured with FIR accelerators, which can run in parallel with core process to reduce the complexity[11]. But restriction on accelerator filter lengths makes them not suitable for long filters. The frequency domain based overlap save & overlap add methods are ideal choice for handling computational complexity problems in real-time implementation. But these techniques require twice

the filter length as the FFT size, which must be a power of 2. Due to this, latency (equal to the length of filter coefficients) is introduced at the output[12–15].

Uniform partitioned convolution is a kind of technique where computational complexity and latency issues are resolved. A complex signal is formed with the outputs of three channel OSD system and these are simplified for implementation point of view by using FFT properties[16–19].

This paper provides the implementation analysis involved in simplifying the three channel CTC system with multi-channel inputs initially in mathematical model. With this analysis, the CTC system was designed so as to fully utilize processor resources at instruction level. It deals with the complexities involved and how to reduce these for better computational power. The Analog Devices 32 bit SHARC

The efficient implementation of shaded filters on DSP processor is the aim of this paper. 21469 floating point processor was used for development and the experimental results of computational complexity and memory requirements for the proposed approach were shown for various filter lengths (maximum of 16384 considered)at a fixed frame size.

This paper is organized as follows. Section 2 provides the existing work for implementing all acoustic filters in CTC system. Section 3 gives mathematical approach in simplifying the CTC structures that suits for implementation. It explains the theoretical computational complexity of the proposed method. Section 4 gives the details of the design on how to implement the proposed method on SHARC 32-bit floating point processor. Section 5 deals with the experimental results and comparison of computational complexity and memory requirements with the better existing approach. Finally section 6 provides the conclusion and future scope.

2. Related Work

This section provides the details of previous work in the area of CTC implementation.

2.1 Mixed Filtering

Reference[20] deals with the simplifying the two channel CTC structure for stereo inputs. In this structure, each input channel is processed by two acoustic filters instead of three filters as in Figure 1. By forming a complex sequence with real-time outputs $y_1(n)$, $y_2(n)$, one could arrive at the frequency domain equivalent of output complex sequence as

$$\begin{aligned} Y(k) &= Y_1(k) + jY_2(k) \\ &= X_1(k)H_1(k) + X_2(k)H_2(k) \end{aligned} \quad (1)$$

where

$$H_1(k) = H_{11}(k) +_j H_{12}(k)$$
$$H_2(k) = H_{21}(k) +_j H_{22}(k)$$

The computational complexity of equation (1) includes one FFT computation with decomposition (to evaluate $X_1(k)$ *and* $X_2(k)$, complex frequency multiplication and one IFFT. The real and imaginary components of IFFT output yield $y_1(n)$ and $y_2(n)$ respectively.

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[21–24]. Figure 2 shows the signal processing involved in this method. Before providing these details, the mathematical interpretation is explained here.

Let x(n), y(n) and h(n) be the input, output and impulse response sequences of an LTI system respectively. The impulse response h(n) is characterized by the filter coefficients of length M. The frame size of input signal is L. The impulse response in z-domain can be simplified as

$$\begin{aligned} H(z) &= \sum_{n=0}^{M-1} h(n) z^{-n} \\ &= \sum_{n=0}^{L-1} \left(\sum_{i=0}^{m-1} z^{-iL} h_i(n) \right) z^{-n} \end{aligned} \quad (2)$$

where

$$\left.\begin{aligned} &h_0(n) = h(n) \\ &h_1(n) = h(n+L) \\ &-- \\ &-- \\ &h_{m-1}(n) = h(n+M-L) \\ &n = 0,1,2,\ldots L-1 \end{aligned}\right\} \& \; m = M/L$$

In equation (2), we assumed that M length sequence was partitioned into m = M/L parts so that each partition has L coefficients. The output can be derived as

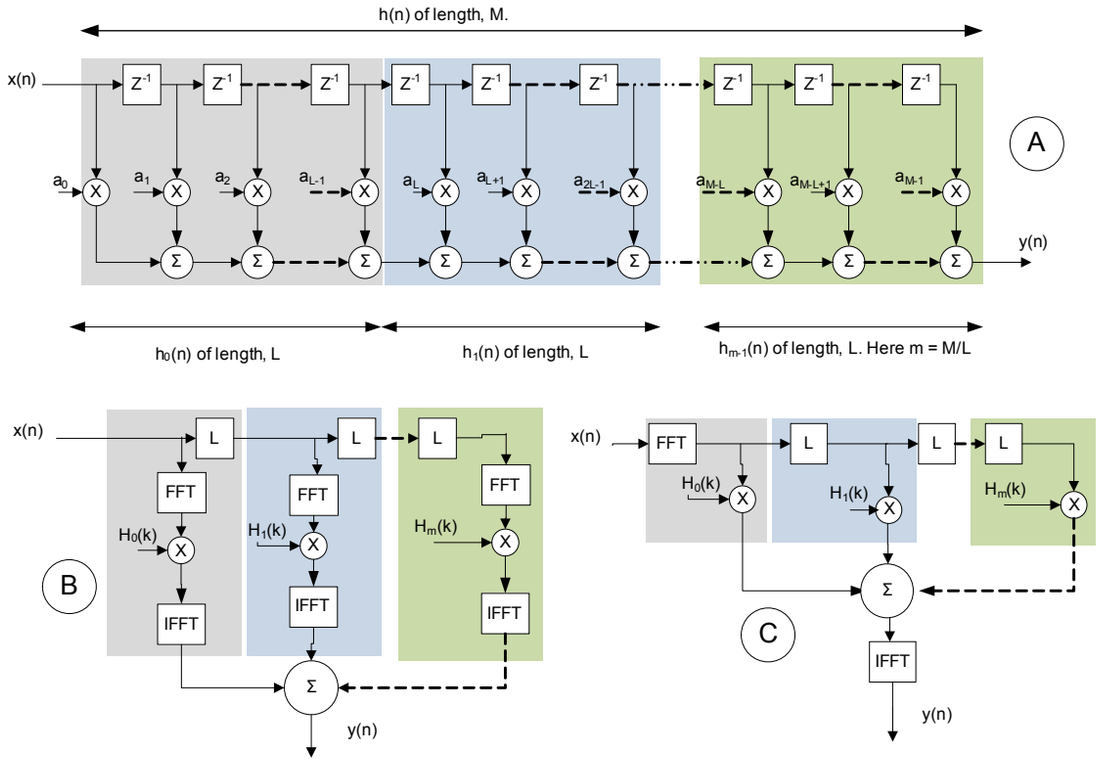

Figure 2. Interpretation of Uniform Partitioned Convolution using block diagram representation.

$$
\begin{aligned}
Y(z) &= H(z)X(z) \\
&= X(z)\sum_{n=0}^{L-1} h_0(n)z^{-n} + z^{-L}X(z)\sum_{n=0}^{L-1} h_1(n)z^{-n} \ldots \\
&\quad + z^{-(m-1)L}X(z)\sum_{n=0}^{L-1} h_{m-1}(n)z^{-n} \\
&= H_0(z)X(z) + z^{-L}X(z)H_1(z) \\
&\quad + \ldots z^{-(m-1)L}X(z)H_{m-1^{(z)}} \\
&= \sum_{i=0}^{m-1} z^{-il}X(z)H_i(z) \qquad (3)
\end{aligned}
$$

Figure 2 describes the interpretation of this equation. Figure 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 (shown in equation 2). Figure 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 Figure 2C just by delaying the FFT outputs. When 2nd frame arrives, FFT output of 1st frame becomes the input 2nd partitioned filter (Refer to equation 3). 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 derived 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.

3. Implementation Analysis of Audio CTC in 3 Channel OSD System

This section describes the mathematical model of proposed algorithm. The computational complexity of this method was given at the end. Note that the frequency index notation z and k are used interchangeably. The meaning is same wherever they used.

The basic idea of this proposed method is to form complex signals with outputs and simplify the outputs in frequency domain. After this, uniform partitioned convolution is applied to simplified CTC structure.

The outputs of audio CTC in OSD model (shown in Figure 1) can be represented in both time and transform domains respectively, as

$$y_j(n)=\sum_{i=1}^{S} x_i(n)*h_{ij}(n) \tag{4}$$

$$\text{and } Y_j(z)=\sum_{i=1}^{S} X_j(z)H_{ij}(z) \tag{5}$$

where j = 1,2,3

By forming complex signal with first two outputs, one can obtain

$$y_1(n)+j\,y_2(n)=\sum_{i=1}^{S} x_i(n)*\left(h_{i1}(n)+jh_{i2}(n)\right) \tag{6}$$

In frequency domain, it is represented as

$$Y_1(z)+jY_2(z)=\sum_{i=1}^{S} X_i(z)H_i(z) \tag{7}$$

where

$$H_i(z)=H_{i1}(z)+jH_{i2}(z)=Z\left[h_{i1}(n)+jh_{i2}(n)\right] \tag{8}$$

To overcome the problems of overlap save method like processing delay and power of 2 FFT length requirements due to appending of zeros in impulse response, the filter impulse responses of audio CTC are partitioned such that total partitions become m = M/L. In this case, each partition has L length of coefficients as shown in Figure 3.

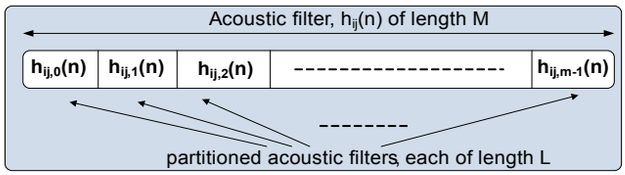

Figure 3. Uniform partitions of acoustic filters.

Let us assume that each acoustic filter length in equation (4) is partitioned in equal length, say, L. The corresponding partitioned coefficients are given by

$$h_{ij}(n)=\left\{h_{ij,0}(n),h_{ij,1}(n),\ldots h_{ij,m-1}(n)\right\} \tag{9}$$

and their transforms are given by $\left\{H_{ij,0}(z), H_{ij,1}(z),\ldots \mathrm{H}_{ij,m-1}(z)\right\}$ where i = 1, 2, 3, 4, 5 and j = 1,2,3. Also, note that z-transform of partitioned coefficients of $h_i(n)$ (= $h_{i1}(n) + j\, h_{i2}(n)$) is represented by $\left\{H_{i,0}(z),H_{i,1}(z),\ldots H_{i,m-1}(z)\right\}$. On substituting the respective partitioned z-transform equivalents in equations (5) and (7), one would get

$$Y_1(z)+jY_2(z)=\sum_{i=0}^{m-1} z^{-iL}\left[\sum_{j=1}^{5} X_j(z)H_{j,i}(z)\right] \tag{10}$$

$$Y_3(z)=\sum_{i=0}^{m-1} z^{-iL}\left[\sum_{j=1}^{5} X_j(z)H_{j3,i}(z)\right] \tag{11}$$

The procedure given in equation (3) was reused here to obtain above expressions. Note that the frequency coefficient sets in equations (10) and (11) are asymmetric and symmetric respectively based on their time domain coefficients. Equation (10) contains real and imaginary time domain coefficients (from equation (8)) whereas equation (11) has real time domain coefficients only. Due to this symmetry, DSP memory could be saved by storing half of the coefficients.

The final time domain outputs of audio CTC system are derived by taking inverse transform for equations (10) and (11).

$$y_1(n)+j\,y_2(n)=z^{-1}\left\{\sum_{i=0}^{m-1} z^{-iL}\left[\sum_{j=1}^{5} X_j(z)H_{j,i}(z)\right]\right\} \tag{12}$$

$$y_3(n)+z^{-1}\left\{\sum_{i=0}^{m-1} z^{-iL}\left[\sum_{j=1}^{5} X_j(z)H_{j3,i}(z)\right]\right\} \tag{13}$$

The realization of above two equations was provided in Figure 4 with maximum of 5 channel inputs. To reduce the complexity, two input channels are combined to

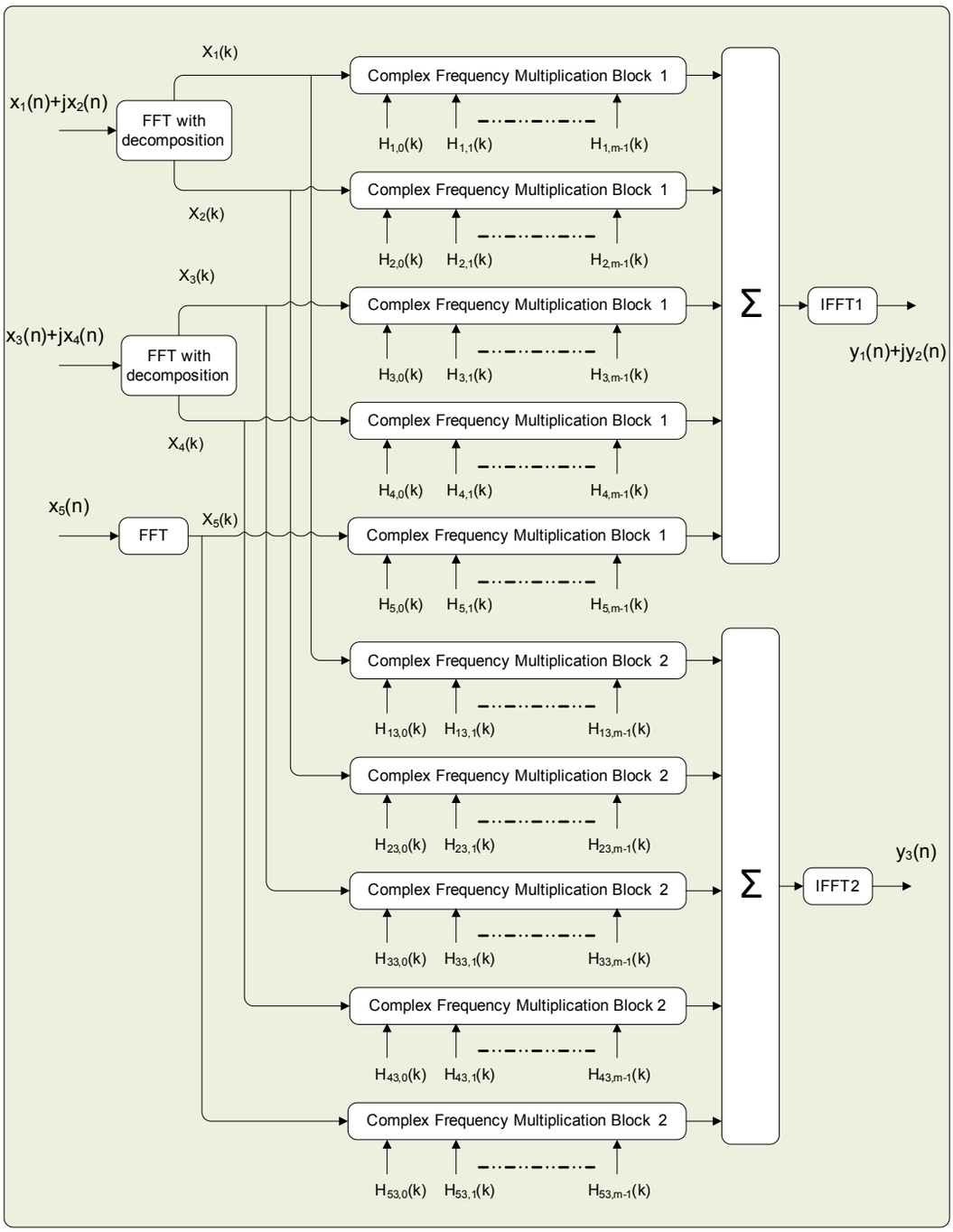

Figure 4. Block diagram for realization of three channel OSD based audio CTC system with multi-channel (maximum of 5) inputs.

form complex signal, which will be applied to FFT and FFTs of individual channels can be extracted using decomposition[12,13]. The complex frequency multiplication block basically does the multiplication of the frequency delayed input FFTs and the FFTs of partitioned filter coefficients. Finally all partitioned outputs are summed before feeding the sum to IFFT for obtaining the filtered outputs. The block diagram shows two sections separately for equation (12) and (13) but the input FFT sections are common for both convolution blocks.

3.1 Theoretical Computational Complexity

The following notation is followed to represent theoretical computational complexity.

Frame Size = L, Coefficients Length = M, FFT Size = N, O(N) = N $\log_2$ N

Theoretical computations contain only the arithmetic operations, such as additions and multiplications required to implement three channel audio CTC system. On real-time DSP processor, these computations vary due to manipulation/copying of the I/O data buffers, coefficients, etc. depending on the operations that need to be performed. Above table provides the computational complexity details

for a pair of input channels in block 1 and single channel in block 2. This could be generalized for S number of channels as shown below

If S is odd,

Total Complex Multiplications:

0.5*(S-1)*(2M + O(2L)) + 0.75 O(2L) + M

Total Complex Additions:

(S-1) *(O(2L) + 2M+L) + 2M – L +0.5 * O(2L)

If S is even,

Total Complex Multiplications: 0.5*S*(2M + O(2L))

Total Complex Additions: S *(O(2L) + 2M+L)

4. Design of 3 Channel OSD Based Audio CTC for Implementation

The mathematical model in previous section provides clear cut analysis for the OSD system on how uniform partition convolution could be applied to long filters for optimum computations. When acoustic filter lengths are long, this approach still looks to be not efficient due to limited on-chip DSP memory. This could be avoided by the efficient use of external and on-chip memories and available hardware DSP resources.

The FFTs of partitioned coefficients were calculated initially and stored in external RAM. Based on the requirement, these could be copied into dedicated buffers of on-chip DSP memory using DMA (Direct Memory Access). For all partitioned coefficients, complex signals (equations (8) and (9)) are formed to feed as inputs to

Table 1. Computational Complexity of 3 channel OSD based audio CTC for a pair of input channels in block 1 & single channel in block 2

To Calculate	Computational complexity		Remarks
	Complex Multiplications	Complex Additions	
Input FFTs for two channels	0.5 O(2L)	O(2L)+2. 2L = O(2L) + 4L	Refer to [11] & [12] for this calculation
Complex Frequency Multiplication Block 1	2L.M/L=2M	2(M-L)	Each Partitioned frequency multiplication requires 2L multiplications. So, 2M complex multiplications are needed for total partitions of M/L. All partitioned multiplier outputs are to be added, which requires 2(M-L) complex additions
Addition of outputs from above block 1	-	2L.m =2M	The outputs of each multiplication block are complex sequences are of length 2L each. The addition of these outputs requires 2L complex additions. This has to be performed for m partitions.
IFFT1	0.5 O(2L)	O(2L)	Complexity of IFFT with size equal to 2L
Complex Multiplication Block 2	M	(M-L)	Due to symmetry of FFTs, these computations are become half from that of block 1.
Addition of outputs from above block 2	-	M	Same as above
IFFT2	0.25 O(2L)	0.5 O(2L)	Same as above

FFT. The obtained FFT real and imaginary buffers are transferred to external RAM using DMA for storage during initialization.

Before going into the details of design, the required memory details of on-chip DSP and external RAM are provided below. The details are given for maximum channel count of 5. These will vary for less number of channels accordingly.

For FFT calculation, real and imaginary buffers along with two temporary buffers, each of size 2L, are needed. Two sets of this buffer list are dedicated in on-chip DSP RAM. The twiddle factors of length L are needed. (real twiddle factor of size L and imaginary buffer of size L). Total memory is 4L + 4L + 2L = 10L (32 bits)

The on-chip memory is dedicated for FFTs of partitioned filter coefficients. That means memory for two sets of real and imaginary buffers (for example $H_{1,0}$, $H_{2,0}$, $H_{13,0}$ and $H_{23,0}$) is dedicated. Total memory here is 2*(2L + 2L + 2L + 2L) = 16L (32bits).

The buffers are required for storing real and imaginary outputs of complex frequency multiplication. For this, 4L (32 bits) size memory is needed for Y1 and Y2. Similarly, 4L size of memory is needed for Y3.

Total Required On-chip DSP Memory = 10L + 16L + 4L + 4L = 34L (32 bits)

The delayed FFTs of all input channels and FFTs of all partitioned coefficients are stored in external memory. Each input channel requires memory of 4M (32bits) where M is filter length. This includes both real and imaginary buffers. For coefficient FFTs, 2M size of memory is needed for real and imaginary buffers separately. In 3 channel OSD system with 5 channel inputs, the total memory required would be 4M *5 * 2 = 40M (32bits).

Total Required External Memory = 4M *5 + 40M = 60M (32 bits)

The main idea of this design is utilization of the external RAM for storing FFTs of all partitioned coefficients and delayed FFTs of input channels, copying them into dedicated internal buffers using DMA (Direct Memory Access) that is running in parallel with core process. When core is performing any processing such as FFT/IFFT/frequency multiplication, DMA copies those The flow chart shown in Figure 5 explains the usage of background DMA in parallel with core process for one frame. When background DMA is used, the consumed cycles are calculated as maximum of DMA cycles and core process cycles. There should not be any buffer conflict between background DMA and core processes. It is always necessary to wait till background DMA completes at the end of core process.

Having achieved algorithmic optimization, it is also required to optimize the code at processor level. This is achieved by utilizing the processor instruction set efficiently. As pointed out in this section, SHARC processor has efficient floating point instruction set to handle parallel operations. It is able to perform one multiplication, one addition, one subtraction and two data move operations in a single instruction with proper use of register set[11]. With efficient use of SIMD (Single Instruction Multiple Data) with parallel instructions, more optimization can be achieved. An equivalent C function was shown in Figure 6 for implementing complex frequency multiplication. When SIMD mode is enabled, the performance can be improved by 50%.

5. Experimental Results and Discussions

The proposed method was implemented on SHARC 32 bit floating point DSP processor ADSP-21469 Ez-Kit Lite board[25]. SHARC is chosen due to the following efficient features that suit audio applications. Though the design is based on SHARC, these features are available on most of the floating point DSP processors.

- Super Harvard Architecture Computer with 32/40-Bit IEEE floating-Point Math.
- No Arithmetic Pipeline; All Computations Are Single-Cycle.
- 32 Address Pointers Support Circular Buffer Addressing Supported in Hardware.
- Six Nested Levels of Zero-Overhead Looping in Hardware.
- Single Instruction Multiple Data (SIMD) Support.
- Instruction Set Supports Conditional Arithmetic, Bit Manipulation, Divide & Square Root, Bit Field Deposit and Extract.
- DMA Allows Zero-Overhead Background Transfers at Full Clock Rate Without Processor Intervention. External DDR2 runs at clock rate of 225MHz, which is half of the DSP speed, 450MHz.
- SHARC-21469 processor has 5M bits of internal RAM and supports 64M (16bits) of DDR2 (external RAM). The frame size used in this experiment is L = 512. The computational measurements were done for acoustic

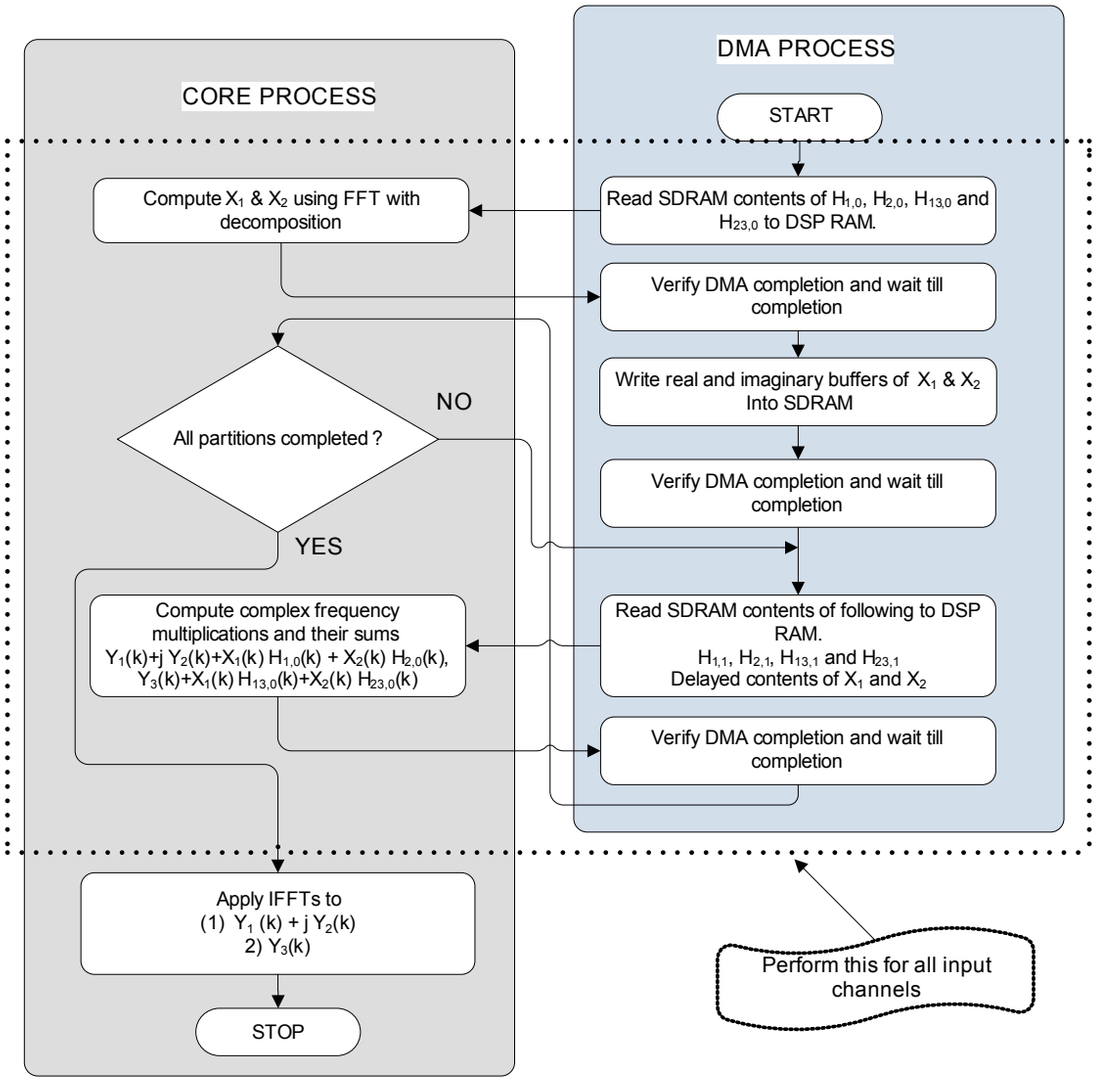

Figure 5. Flow chart that shows design/data flow of 3 channel OSD based audio CTC system for multi-channel inputs.

filter lengths ranging from 1024 to 16384 at step size of 1024. The memory requirements (in 32 bit words) for L = 512 and M = 16384 are shown below as per discussion in Section 3.

Total internal RAM	:	17408
Total DDR2 usage	:	983040

The internal memory of 5Mb is divided into 4 memory blocks. The arrangement of FFT buffers and coefficient buffers are made in such a way that any two buffers can be accessed from different blocks at the same time using parallel instructions. The optimized FFT/IFFT routines are implemented based on derivations given in references[8,9].

During the core process, DMA routines are performed by using two DMA channels DMAC0 and DMAC1. It is necessary to copy two buffers stored at different locations in external RAM into separate locations in internal RAM. For this purpose, DMA is configured in chaining mode with two DMA channels and DMA copying will happen in parallel to core process.

An audio signal of 44.1 kHz sampling frequency with multi-channels is provided as input to the system for mea-

```
void    ComplexFrequencyMultiplication(float    *pReal1,
                                       float    *pImag1,
                                       float    *pReal2,
                                       float    *pImag2,
                                       float    *pReal3,
                                       float    *pImag3,
                                       float    *pRealOut1,
                                       float    *pImagOut1,
                                       float    *pRealOut2,
                                       float    *pImagOut2,
                                       unsigned int bufSize)
{
    unsigned int index;

    float   *pTempReal3 = pReal3 + bufSize/2 + 1;
    float   *pTempImag3 = pImag3 + bufSize/2 + 1;

    /*  Copying 1st half of data buffer to the 2nd half of
    data buffer due to symmetry*/
    for(index = 1;index < bufSize/2 ; index++)
    {
        *pTempReal3++ = pReal3[bufSize/2 - index];
        *pTempImag3++ = pImag3[bufSize/2 - index];
    }
    //Enabling SIMD mode
    asm("bit set MODE1 0x200000;");
    asm("nop;");
    /* loop count becomes half of the buffer size due to
    enabling of SIMD */
    for(index = 0; index < bufSize/2; index++)
    {
        *pRealOut1 += *pReal1 * *pReal2 - *pImag1 * *pImag2;
        *pImagOut1 += *pReal1 * *pImag2 + *pImag1 * *pReal2;

        *pRealOut2 += *pReal1 * *pReal3 - *pImag1 * *pImag3;
        *pImagOut2 += *pReal1 * *pImag3 + *pImag1 * *pReal3;

        pReal1 += 2;
        pImag1 += 2;
        pReal2 += 2;
        pImag2 += 2;
        pRealOut1 += 2;
        pImagOut1 += 2;
        pRealOut2 += 2;
        pImagOut2 += 2;
    }
    //Disabling SIMD mode
    asm("bit clr MODE1 0x200000;");
    asm("nop;");
}
```

Figure 6. Equivalent C code of SHARC Assembly to implement complex frequency multiplication.

suring the mega peak cycle counts. The computational complexity was measured with hardware cycle count registers. The header file cycle_count. h needs to be included and it contains the definitions for cycle_t datatype. *START_CYCLE_COUNT* and *STOP_CYCLE_COUNT* measures the current cycle count and final cycle count for *OSD_MultiChannel_Processing* function respectively. Finally peak_count contains the peak cycle count, which is the maximum of final cycle counts for all frames.

The 3 channel OSD based audio CTC was implemented with mixed overlap save method and individual uniform partitioned convolution for 2 channel (stereo) and 3 channel inputs. The comparison of mega peak cycle counts was provided in Figure 7 & Figure 8 at various filter lengths. The proposed method is more efficient and the computational savings vary based on filter length. The overlap save approach basically suffers from more FFT lengths due to appending of zeros. Since FFT needs its size must be power of 2 and zeros should be appended to filters, its size become 16384 for filter lengths ranging from 8192 to 15360, for example. This difficulty was resolved efficiently with proposed method with combined use of mixed filtering and uniform partitioned convolution. If uniform partitioned convolution is applied to individual filters i.e. without mixed filtering, the computational complexity of frequency multiplication and IFFT stages increase. If input channels increase, more memory is required for overlap save method and real-time was not met for higher acoustic filter lengths (greater than or equal to 8192 for 3 input channels and equal to 16384 for stereo inputs) as power consumption at these lengths is exceeding 450 MHz (Power consumption is given by expression: mega peak cycle count * sampling frequency/ frame length).

For different input channels, the peak cycle count was measured and potted this for CTC filter lengths ranging from 1024 to 16384. The comparison of these results for various input channels is provided in Figure 9. This shows proportional increase in computational complexity for various filter lengths and various channels. The mega peak cycle count for 5 channels at M = 16384 is 2.71504. For this, the power consumption is found as

233.85MHz. Here operating sampling frequency and frame length are 44.1 kHz and 512 respectively. This power consumption is 51.9% of the available power (450MHz). The existing design was unique for all filter lengths. By storing some of the filter coefficients in on-chip memory (for filter lengths less than 8192), the power consumption still could be reduced.

```
cycle_t start_count, final_count, peak_count;

// start_count contains current cycle count
START_CYCLE_COUNT(start_count);

// function call for which performance measurement to be done
OSD_MultiChannel_Processing(...);

//  final_count contains performance cycle count
STOP_CYCLE_COUNT(final_count,start_count);

//  peak_count contains maximum of final_count for all frames
peak_count   =  (peak_count < final_count)? final_count: peak_count;
```

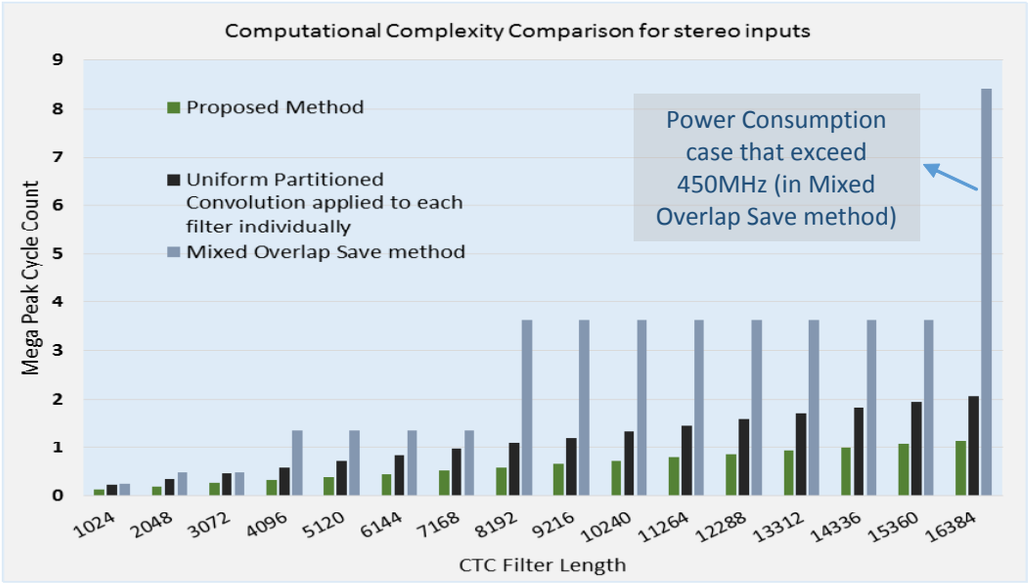

Figure 7. Computational complexity comparison between proposed method and conventional methods for stereo channel inputs. X-axis represents filter length, M. Y-axis represents Mega Peak cycle count. The above results are valid for frame length of L = 512.

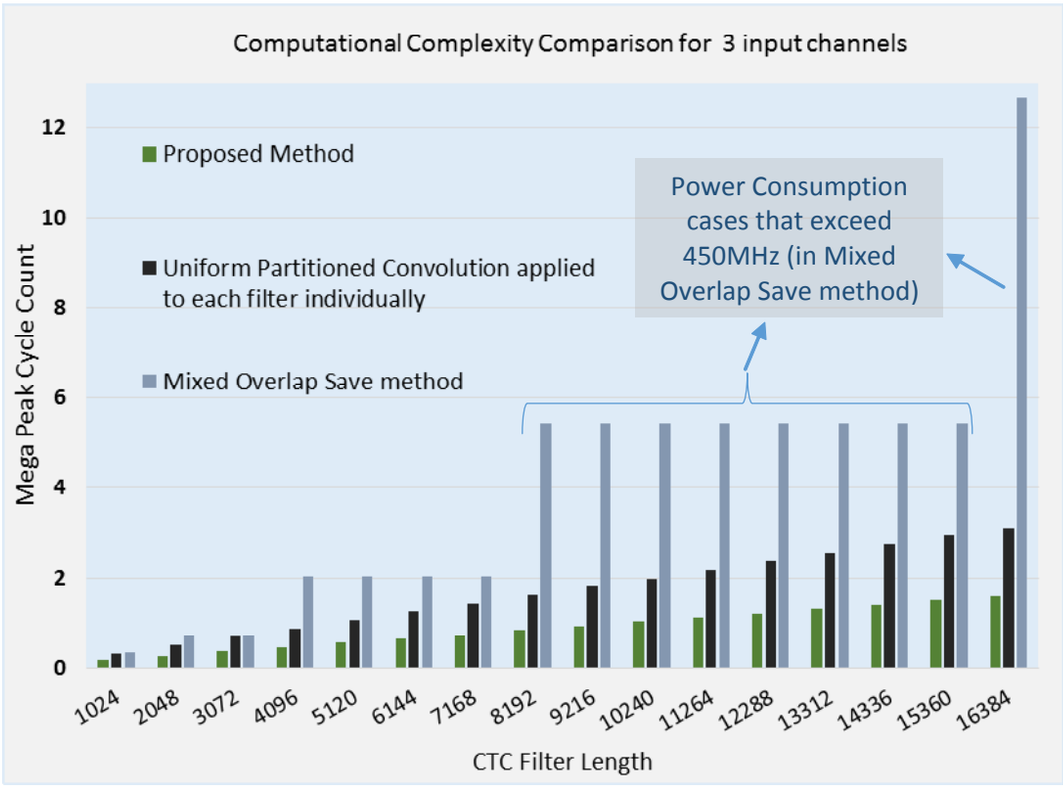

Figure 8. Computational complexity comparison between proposed method and conventional methods for 3 input channels. X-axis represents filter length, M. Y-axis represents Mega Peak cycle count. The above results are valid for frame length of L = 512.

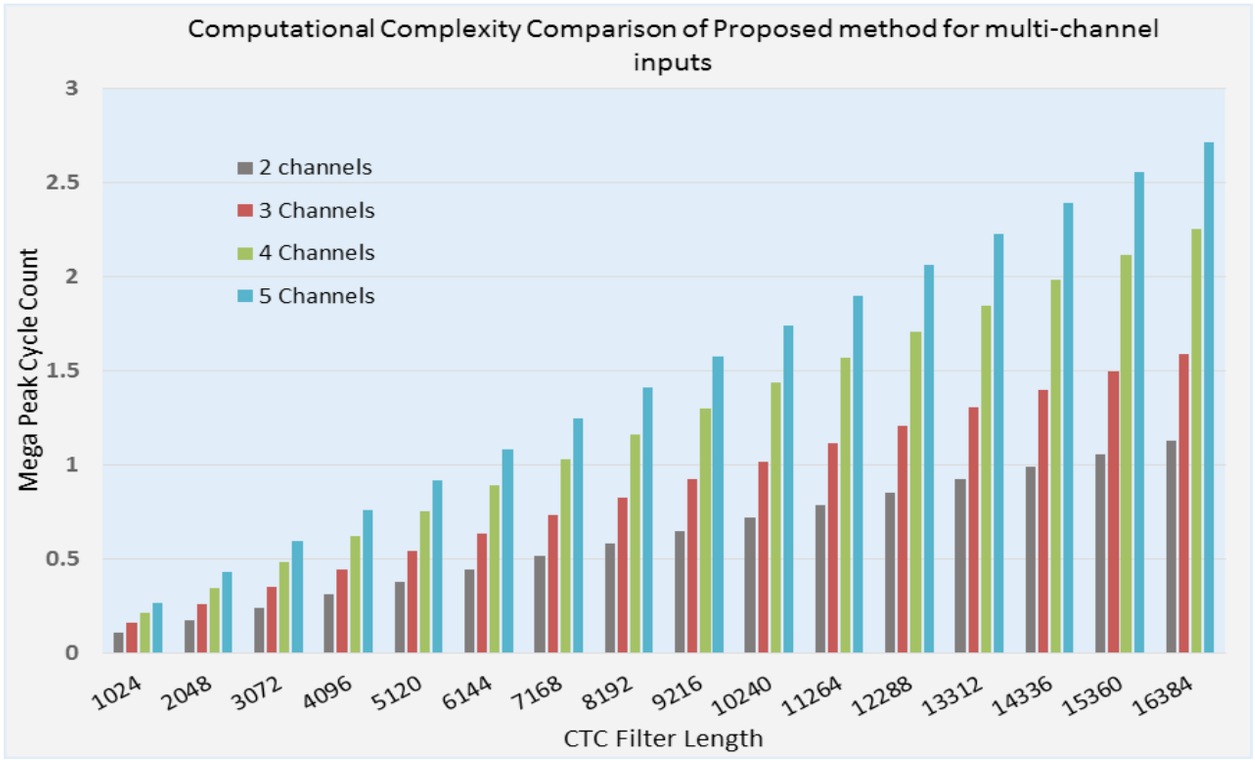

Figure 9. Computational complexity comparison. X-axis represents filter length, M. Y-axis represents Mega Peak cycle count. The above results are valid for frame length of L = 512

6. Conclusion and Scope of Future Work

An efficient and low cost solution is proposed to implement three channel OSD system with multi-channel source on DSP processors. This method not only reduces the processing power but also the delay that overlap-save method suffers from. The computational results indicate that the proposed method is good technique for implementing long filters in OSD system. The advantage here is the usage of single DSP to implement all filters, each of maximum length 16384.

The alternate technique to uniform partitioned convolution is non-uniform partitioned convolution that requires multiple cores to perform complex frequency multiplication stage in efficient way. By scheduling the partitioned filters efficiently, the latency would be reduced to almost equal to zero and at the same time, the computations would be reduced drastically for more filter lengths with the usage of multiple cores.

7. References

1. Otani M, Ise S. Fast calculation system specialized for head-related transfer function based on boundary element method. J Acoust Soc Am. 2006; 119(5):2589–98.
2. Ole K, Per R, Philip NA, Angelo F. Design of crosstalk cancellation networks by using fast deconvolution. Audio Engineering Society. 1999, May; p. 9900–5.
3. Tobias L, Oliver S. Adaptive Cross-talk cancellation system for a moving listener. Proceedings of 21st International Conference on Audio Engineering Society. 2002, June. p. 00134.
4. Linwang, Yin F, Chen Z. A stereo crosstalk cancellation system based on common-acoustical pole/zero model. Audio Engineering Society. 2010, Aug. p. 719197.
5. Available from: http://resource.isvr.soton.ac.uk/FDAG/VAP/html/osd.html
6. Available from: http://*resource.isvr.soton.ac.uk/ FDAG/ VAP/html/***OPSODIS***.pps*
7. Available from:http://resource.isvr.soton.ac.uk/FDAG/VAP/html/vasi.htm
8. Nelson PA et al. The binaural performance of a cross-talk cancellation system with matched or mismatched setup

and playback acoustics. Journal of Acoustical Society of America. 2013, Feb.
9. Choueiri EY. Optimal crosstalk cancellation for binaural audio with two loudspeakers. 2004, Aug.
10. Yang J et al. Development of virtual sound imaging system using triple elevated speakers. IEEE Transactions on Consumer Electronics. 2004, Aug; 50(3):916–22.
11. Analog Devices Inc. ADSP-214xx SHARC Processor Hardware Reference Manual. 2010 Jul.
12. Proakis JG, Manolakis DG. Digital Signal Processing Principles, Algorithms and Applications. 3rd ed. Prentice Hall Publications; 2006.
13. Lyons RG. Understanding Digital Signal Processing, 3rd ed. Prentice Hall; 2010 Nov.
14. VandeKieft JR. Computational improvements to linear convolution with multi-rate filtering methods. 1998, Apr. Available from: http://mue.music.miami.edu/thesis/jvandekieft/jvtitle.htm.
15. Vetterli M. Running FIR and IIR filters using multi-rate filter banks. IEEE Transactions on Acoustics, Speech and Signal Processing. 1998, May; 36(5):730–8.
16. Battenbaerg E, Avizienis R. Implementing real-time partitioned convolution algorithms on conventional operating systems. Proceedings of 14th International Conference on Digital Audio Effects (DAFX 2011). 2011 Sep.
17. Torger A, Farina A. Real-time partitioned convolution for ambiophonic surround sound. IEEE Workshop on applications of Digital Signal Processing to Audio and Acoustics. 2001. p. 195–8.
18. Guillermo G. Optimal filter partition for efficient convolution with short input/output delay. Proceedings on 113th International Conference on Audio Engineering Society. 2002 Oct. p. 2660.
19. Gardiner WG. Efficient convolution without input-output delay. J Audio Eng Soc. 1995; 43(3);127–36.
20. SreenivasaRao Ch, Udalaylakshmi R, Jeyasingh P. Fast implementation of audio crosstalk cancellation on dsp processors. 45th International Conference on Audio Engineering Society. 2012 Mar. p. 2.
21. Egelmeers G, Sommen P. A new method for efficient convolution in frequency domain by nonuniform partitioning for adaptive filtering. IEEE Transactions on Signal Processing. 2001; 44(12):3123–9.
22. Cownas B, Karpalos B. Real-time GPU based convolution. Proceddings of 2009 Conference on Future Play. 2009.
23. Muller-TomFelde C. Time varying filter in non-uniform block convolution. Conference on Digital Audio Effects. 2001.
24. Toger A, Farina A. Real-time partitioned convolution for ambiophonics surround sound. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. 2001. p. 195–8.
25. Analog Devices Inc. ADSP-21469 Ez-Kit Lite Evaluation Board Manual. 2012.