International Journal of
Systems Signal Control
and Engineering Application

MEDWELL PUBLICATIONS
SINCE 2000

Partitioned Convolution Analysis for Stereo Inputs Based Three Channel Optimal Source Distribution on Heterogeneous Parallel Computing Platforms using OpenCL

[1]Sreenivasa Rao Chunduri, [1]Venkata Rao Dhulipalla and [2]Lakshmi Narayana Somayajula
[1]Department of ECE, KL University, Vijayawada 522502, India
[2]Narasaraopet Instt. of Tech, Narasaraopet, Guntur 522601, India

Key words: Optimal sound distribution, partitioned convolution, mixed filtering, openCL, data and task parallelism

Corresponding Author:
Sreenivasa Rao Chunduri
Department of ECE, KL University, Vijayawada 522502, India

Page No.: 127-137
Volume: 13, Issue 5, 2020
ISSN: 1997-5422
International Journal of Systems Signal Control and Engineering Application
Copy Right: Medwell Publications

Abstract: Partitioned convolutions are the best methods to address the system performance related issues in 3D virtualization techniques both in terms of latency and computational complexity. General DSP processor architectures are not suitable to implement very long filters due to increase in computational complexity and required on-chip memory. In this study, an efficient method called Mixed Non-uniform partitioned convolution is explained to overcome computational problems for implementing three channel OSD (Optimal Source Distribution) with stereo inputs on heterogeneous parallel computing platforms. With the massive parallel computing architecture, the partitioning scheme used for this method prove that it is possible to implement OSD system containing 6 filters, each filter has a filter length of 65536 (32-bit floating point) on these platforms. The proposed algorithms were implemented on AMD based Bonaire GPU using task parallelism. The advantage of proposed method is that it provides zero output latency which is desired in real-time applications. The computational performance and the system cost of proposed method was compared with existing approaches. The performance comparison clearly provides information that the proposed approach is suitable for implementation of OSD system at very long filter lengths with reasonable system cost in terms of compute units.

INTRODUCTION

The 3D audio virtual techniques are most popular and have several applications in home theatre entertainment, gaming, teleconference and remote control. The aim of these techniques is to reproduce the spatial audio pattern at the ears of the listener so that one would feel that he actually resided in the real audio environment. Initially, Head Related Transfer Function (HRTF) technique was introduced in 1983 in which the listener wears the headphones to enjoy the audio and the incoming signal will be processed using the HRTF filters whose impulse responses were measured based on the environment. Though this technique has excellent channel separation, they are inconvenient to wear, particularly when more number of users are enjoying the audio. Later,

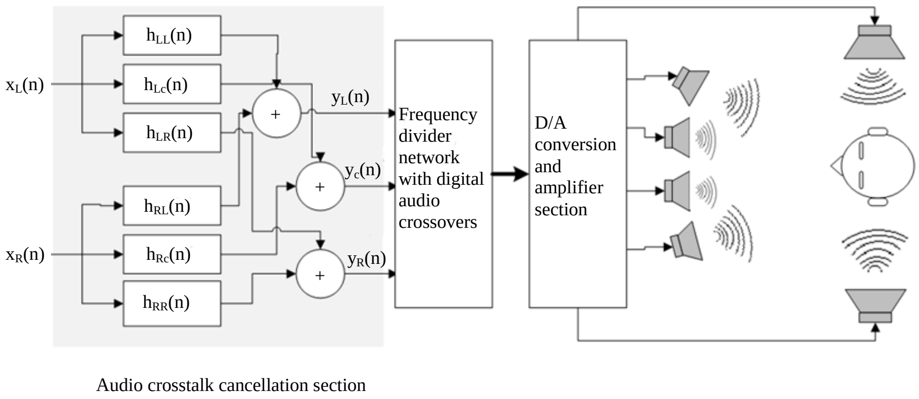
$x_L(n)$
$x_R(n)$
$h_{LL}(n)$
$h_{Lc}(n)$
$h_{LR}(n)$
$h_{RL}(n)$
$h_{Rc}(n)$
$h_{RR}(n)$
+
+
+
$y_L(n)$
$y_c(n)$
$y_R(n)$
Frequency divider network with digital audio crossovers
D/A conversion and amplifier section
Audio crosstalk cancellation section

Fig. 1: Three channel OSD system with stereo source

conventional loudspeaker system was developed. The main disadvantage of this technique is system inversion to cancel out the unintended sounds between loudspeakers and the ears of the listener[1-2].

Takashi in his research found that the condition number of the inverse acoustic matrix is of type ill conditioned. This means a small variation of the listener's head causes wrong impression in identifying the direction of the sound. More importantly, it causes the listener to treat the sound is coming from far though it is actually originating from near and vice versa. Takashi has come up with smart solution to overcome this problem by forming a relationship between operating frequency and the position of the loudspeaker. He divided the audio bandwidth into three regions, namely low, mid and high pass regions so that in each region the condition number should be unity. He called this technique as Optimum Source Distribution (OSD)[3, 4].

For perfect crosstalk cancellation, advantage of a simple phase change is enhanced by introducing one more loudspeaker between binaural loudspeakers. This is referred to as three channel OSD system. Here the word "three channel" means that each input source is processed by three crosstalk cancellation filters. Figure 1 shows three channel OSD system in which each channel is processed by three individual Crosstalk Cancellation (CTC) filters and the output of CTC block is fed to frequency divider network[3, 4].

To reproduce the spatial reverberation characteristics at the desired locations, it is necessary to use long filters in CTC section. This requires more computational power for implementation of these filters on general purpose DSP processors. This study mainly concentrates on describing the computational complexity issues when implementing very long filters in audio CTC and provides a feasible solution called mixed non-uniform partitioned convolution to implement CTC section on heterogeneous parallel computing platforms.

Existing implementations: The original and basic method to proceed is time domain convolution. It is suitable for very shorter lengths in order of 1024 and is not preferred for long filters due to more power consumption. Recently, the DSP processor manufacturers are come up with FIR accelerators to save computational power. These accelerators can run in parallel with core process to reduce the complexity. The ADI SHARC 214xx series processors are best example for this. But restriction on accelerator filter lengths makes them not suitable for long filters[5, 6].

On the other hand, frequency domain techniques provide better computational complexity. The techniques like overlap save and overlap add methods are examples. The drawback of these methods is output latency. Due to appending of zeroes to the original impulse response to match the FFT lengths, latency is introduced at the output. The latency depends on the number of zeroes appended and typically equal to the length of impulse response. For example, if the system is operating at 44.1 kHz and impulse response length is 8192, the output latency is 185.7 msec approximately, which is undesired in real-time applications. Also FFT requires twice the impulse response length as its size. Due to this, the computational power increases drastically[5, 6].

To overcome the latency problems, partitioned filter approach was developed. The idea behind this approach is to partition the impulse response uniformly and apply overlap save method for each partition. The number of partitions become M/L assuming that M and L are lengths of impulse response and processing frame, respectively. Usually M>>L. The length of each partition becomes L so that when overlap save method is applied for each partition, it is enough to append L zeroes to each partition. Hence the latency in this case is equal to L/Fs where Fs is sampling frequency. For example, latency becomes 5.8 msec for L = 256 and Fs = 44.1 kHz which is drastic improvement compared to overlap save method. To

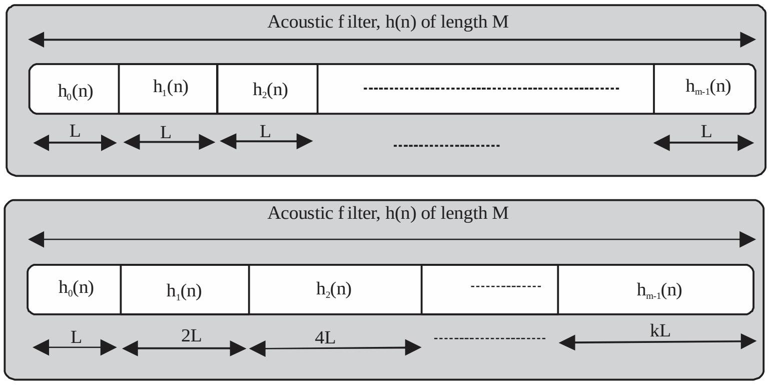

Fig. 2: [Top] Uniform partitioning of impulse response. Each partition length is L, so that, total partitions become m = M/L. [Bottom] Non-uniform partitioning scheme. The total partitions are based on partitioning scheme followed. The values m and k are decided based on impulse response length and partitioning scheme

optimize the computational complexity, the structure of this method can be adjusted in such a way that only single FFT and IFFT would be needed and all the frequency domain contents are delayed for process of further partitions instead of time domain delayed contents. In this way, this method is referred to as uniform partitioned convolution as partitions are uniform and also called Single Frequency Delay Line filter (SFDL) as FFT contents of the input frames are delayed and used in processing of further partitions[7-10].

SFDL approach solves major problems of latency and computational complexity upto certain extent. But when filter length is very long, the complex frequency multiplication blocks will increase and this results into more computational complexity. To avoid these problems, Gardener and Garcia suggested non-uniform partitioning approach. In this method, impulse response is partitioned non-uniformly starting with shorter partition length and gradually increase the partition length. The initial partition length is preferred short to obtain less latency. Even if time domain convolution method is followed for this partition, zero latency would be obtained. The gradual increase in partitions are preferred to obtain low computational complexity. To achieve optimum computational complexity, massive parallel architectures are required, so that, each non-uniform partition can be executed in parallel. Figure 2 shows the partitioning schemes for uniform and non-uniform cases[7-10].

SreenivasaRao *et al.*[11], Rao *et al.*[12], SreenivasaRao *et al.*[13], SreenivasaRao and VenkataRao[14] and SreenivasaRao *et al.*[15] provide an efficient mechanism on how to optimize the implementation of audio CTC filters for various multi-channel inputs. An optimum method called Mixed Uniform partitioned convolution was explained to utilize algorithmic as well as processor level optimization efficiently on DSP processors. When CTC filter lengths are very long i.e. in the order of 32768, DSP processors are inefficient to handle computational complexity and latency issues as these architectures don't have massive parallelism compute units. Lot of memory is required when processing the signal in frequency domain and to store various FFT coefficients and intermediate buffers. DSP processors may not support required on-chip memory. There is a possibility to store the coefficients in external memory and copy these into on-chip using DMA but this takes more cycles. Hence, a suitable architecture called heterogeneous parallel computing platform is required to perform parallel operations. This architecture is explained in section 5.

The audio CTC section in Fig. 1 contains 6 such long filters and if these are implemented separately using non-uniform partitioned convolution on parallel platforms, it is very difficult to handle FFT buffers and partitioned coefficients related to each partition as well as for each filter. In this study, an efficient method called mixed non-uniform partitioned convolution is explained to achieve best optimization in processing of signal with very long filters. This approach initially relies on simplifying overall CTC structure in frequency domain based on method called mixed filtering and applying non-uniform partitioned convolution to simplified result.

MATERIALS AND METHODS

Objectives of current work: In this study, mixed filtering is explained to simplify the 3 channel audio CTC section. This is combined with non-uniform partitioned convolution to obtain optimum computational complexity. The mathematical model of each method is explained in detail. The maximum filter length of 65536 for each filter is experimented using the proposed approach. This method was implemented on AMD Radeon HD 7900 series GPUs. The partitioning scheme followed was explained. Various tests were conducted to support the proposed method such as latency, computational complexity tests using measured impulse responses. The results of the proposed method are compared with those of existing techniques.

Proposed solution-mixed non-uniform partitioned convolution: This study describes the mathematical model of proposed algorithm. Initially, mixed filtering concept was explained and later partitioned convolution was applied to the results obtained in mixed filtering. To proceed with, the outputs of audio CTC section (shown in Fig. 1) can be represented in both time and transform domains, respectively as:

$$y_L(n) = x_L(n) * h_{LL}(n) + x_R(n) * h_{RL}(n) \tag{1}$$

$$y_R(n) = x_L(n) * h_{LR}(n) + x_R(n) * h_{RR}(n) \tag{2}$$

$$y_C(n) = x_L(n) * h_{LC}(n) + x_R(n) * h_{RC}(n) \tag{3}$$

and:

$$Y_L(z) = X_L(z) H_{LL}(z) + X_R(z) H_{RL}(z) \tag{4}$$

$$Y_R(z) = X_L(z) H_{LR}(z) + X_R(z) H_{RR}(z) \tag{5}$$

$$Y_C(z) = X_L(z) H_{LC}(z) + X_R(z) H_{RC}(z) \tag{6}$$

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

$$\begin{aligned} y_L(n) + jy_R(n) = x_L(n) * \left(h_{LL}(n) + jh_{LR}(n)\right) + \\ x_R(n) * \left(h_{LR}(n) + jh_{RR}(n)\right) \end{aligned} \tag{7}$$

In frequency domain, it is represented as:

$$Y_L(z) + jY_R(z) = X_L(z) H_L(z) + X_R(z) H_R(z) \tag{8}$$

where:

$$H_L(z) = H_{LL}(z) + jH_{LR}(z) \tag{9}$$

$$H_R(z) = H_{RL}(z) + jH_{RR}(z) \tag{10}$$

Equation 8 gives the frequency contents of $y_L(n)$ and $y_R(n)$. In other words, real and imaginary parts of IFFT applied to Eq. 8 yield both time domain outputs respectively. While calculating the FFTs of inputs $x_L(n)$ and $x_R(n)$, a single FFT with decomposition can be used to get better optimization instead of using two individual FFTs[5]. Also Eq. 6, provides the frequency domain representation of yC(n). All the components in this equation are real and hence its FFT is symmetric with respect to real axis. Equation 6 and 8 are collectively referred to as mixed filtering approach because there is no need to perform filtering operations for each filter separately. Two equations will do the job of 6 filters.

This approach is better suitable for implementation of filters in the order of 1024-2048 using overlap save method because it is easy to manage the required memory and implementation complexity on general purpose DSP processors. But when the filter lengths are in the order of 16384, it is very difficult to handle computational complexity. As mentioned in section 2,[11-15] explained the mixed uniform partitioned convolution on how to manage implementation complexity using external memory and DMA on SHARC DSP processors. The aim of this paper is to support more filter lengths than 16384 and to derive better algorithm to achieve this. To address this, the impulse responses in Eq. 3 and 7 are partitioned non-uniformly. The impulse responses $h_L(n)$, $h_R(n)$, $h_{LC}(n)$ and $h_{RC}(n)$ are partitioned as:

$$\begin{aligned} & h_L(n) = \left\{h_{L,0}(n),\ h_{L,1}(n),\ h_{L,2}(n),\ \ldots,\ h_{L,l-1}(n)\right\} = \\ & \left\{\begin{matrix} h_{LL,0}(n),\ jh_{LR,0}(n),\ h_{LL,1}(n) + \\ jh_{LR,1}(n),\ \ldots,\ h_{LL,l-1}(n) + jh_{LR,l-1}(n) \end{matrix}\right\} \end{aligned}$$

$$\begin{aligned} & h_R(n) = \left\{h_{R,0}(n),\ h_{R,1}(n),\ h_{R,2}(n),\ \ldots,\ h_{R,l-1}(n)\right\} = \\ & \left\{\begin{matrix} h_{RL,0}(n),\ jh_{RR,0}(n),\ h_{RL,1}(n) + \\ jh_{RR,1}(n),\ \ldots,\ h_{RL,l-1}(n) + jh_{RR,l-1}(n) \end{matrix}\right\} \end{aligned}$$

$$\begin{aligned} & h_{LC}(n) = \left\{h_{LC,0}(n),\ h_{LC,1}(n),\ h_{LC,2}(n),\ \ldots,\ h_{LC,l-1}(n)\right\} \\ & h_{RC}(n) = \left\{h_{RC,0}(n),\ h_{RC,1}(n),\ h_{RC,2}(n),\ \ldots,\ h_{RC,l-1}(n)\right\} \end{aligned}$$

where, each impulse response has partition lengths in the order of L_1, L_2, ..., L_l. Equation 8 can be written in terms of partitions as:

$$\begin{aligned} & Y_L(z) + jY_R(z) = X_L(z) \begin{bmatrix} H_{L,0}(z) + z^{-L_1} H_{L,1}(z) \\ +,\ \ldots,\ + z^{-L_{l-1}} H_{L,l-1}(z) \end{bmatrix} + \\ & X_R(z) \left[H_{R,0}(z) + z^{-L_1} H_{R,1}(z) +,\ \ldots,\ + z^{-L_{l-1}} H_{R,l-1}(z)\right] = \\ & X_L(z) H_{L,0}(z) + z^{-L_1} X_R(z) H_{L,1}(z) \\ & +,\ \ldots,\ z^{-L_{l-1}} X_L(z) H_{L,l-1}(z) + X_R(z) H_{R,0}(z) + \\ & z^{-L_1} X_R(z) H_{R,1}(z) +,\ \ldots,\ z^{-L_{l-1}} X_R(z) H_{R,l-1}(z) = \\ & \left[X_L,\ H_{L,0},\ 0,\ L_1\right]_{UPC} + \left[X_L,\ H_{L,1},\ L_1,\ L_2\right]_{UPC} \\ & +,\ \ldots,\ + \left[X_L,\ H_{L,l-1},\ L_{l-1},\ L_1\right]_{UPC} + \left[X_R,\ H_{R,0},\ 0,\ L_1\right]_{UPC} + \\ & \left[X_R,\ H_{R,1},\ L_1,\ L_2\right]_{UPC} +,\ \ldots,\ + \left[X_R,\ H_{R,l-1},\ L_{l-1},\ L_l\right]_{UPC} \end{aligned} \tag{11}$$

Similarly, Eq. 6 can be expanded as:

$$\begin{aligned} & Y_C(z) = \left[X_L,\ H_{LC,0},\ 0,\ L_1\right]_{UPC} + \left[X_L,\ H_{LC,1},\ L_1,\ L_2\right]_{UPC} \\ & +,\ \ldots,\ + \left[X_L,\ H_{LC,l-1},\ L_{l-1},\ L_1\right]_{UPC} + \left[X_R,\ H_{RC,0},\ 0,\ L_1\right]_{UPC} + \\ & \left[X_R,\ H_{RC,1},\ L_1,\ L_2\right]_{UPC} +,\ \ldots,\ + \left[X_R,\ H_{RC,l-1},\ L_{l-1},\ L_l\right]_{UPC} \end{aligned} \tag{12}$$

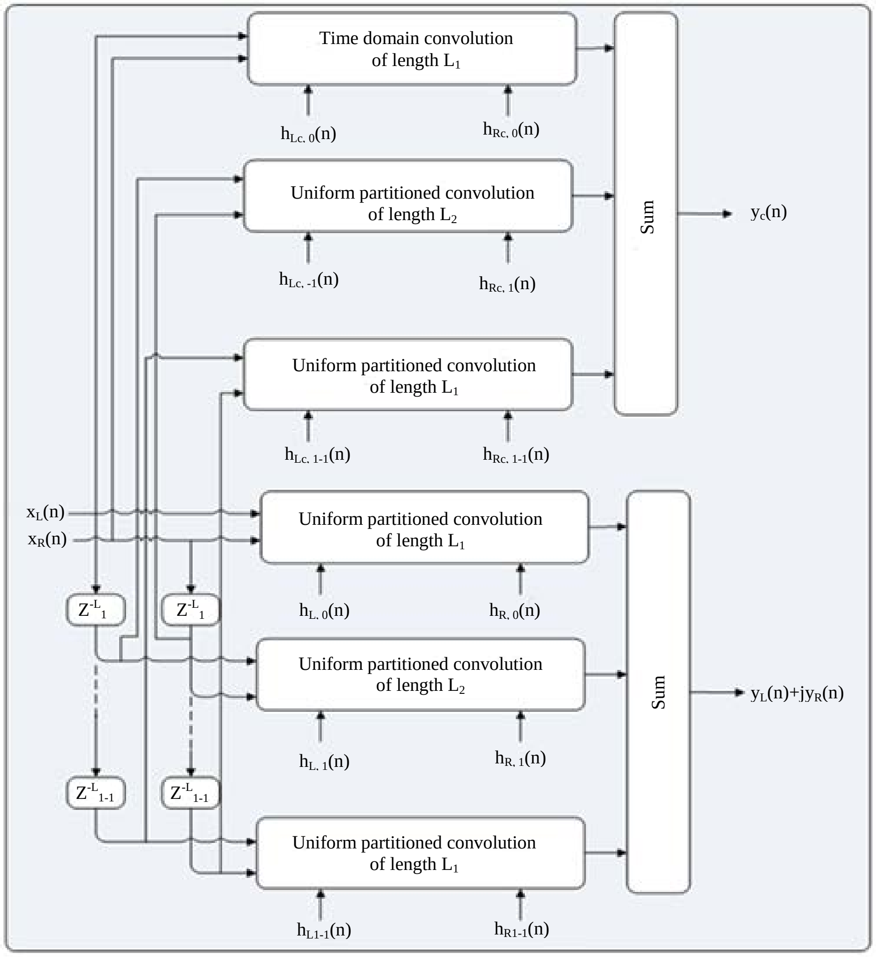
Time domain convolution of length L_1
$h_{Lc,0}(n)$
$h_{Rc,0}(n)$
Uniform partitioned convolution of length L_2
$h_{Lc,-1}(n)$
$h_{Rc,1}(n)$
Uniform partitioned convolution of length L_1
$h_{Lc,1-1}(n)$
$h_{Rc,1-1}(n)$
Sum
$y_c(n)$
$x_L(n)$
$x_R(n)$
Uniform partitioned convolution of length L_1
$Z^{-L}{}_1$
$Z^{-L}{}_1$
$h_{L,0}(n)$
$h_{R,0}(n)$
Uniform partitioned convolution of length L_2
$h_{L,1}(n)$
$h_{R,1}(n)$
Sum
$y_L(n)+jy_R(n)$
$Z^{-L}{}_{1-1}$
$Z^{-L}{}_{1-1}$
Uniform partitioned convolution of length L_1
$h_{L1-1}(n)$
$h_{R1-1}(n)$

Fig. 3: Block diagram of proposed algorithm-Mixed Non-uniform partitioned convolution

Here the term $[A, B, C, D]_{UPC}$ means that C samples delayed input frame A is processed by the partitioned filter B of length L1. The suffix UPC is attached to each term because each partition is implemented using uniform partitioned convolution. Finally the time-domain outputs of CTC section are given by taking inverse z-transform for Eq. 11 and 12. Even though the impulse response is partitioned non-uniformly, each non uniform partition, in turn, implemented using uniform partitioned convolution in order keep the output latency low. For example, assume impulse response of length 8192 is partitioned into 4 non-uniform partitions, each of length 1024 and 2 non-uniform partitions, each of length 2048 ($8192 = 4\times1024+2\times2048$). The first partition of length 1024 still requires appending of 1024 zeroes to the partitioned length in order to use overlap save method that results latency problem. To overcome this, it is better to use uniform partitioned convolution for implementing each non-uniform partition. If frame length is 128, obviously each non-uniform partition yields output latency of 2.9 msec (128/44.1 kHz). But we discussed that parallel computing platforms are suitable for implementing these algorithms, the output latency is not sum of all latencies. So 2.9 msec is the output latency of overall CTC section. Still there is possibility to obtain zero latency with this approach. As all partitions are executed in parallel, it is better if first partition is implemented using time domain convolution because time domain yields zero latency. But the computational complexity of first partition should not exceed that of any other partitions. This is guaranteed because the partitioning scheme makes sure that gradual increase in partition lengths from lower partition to possible higher partitions.

Figure 3 shows the block diagram of proposed algorithm. When first frame of L input samples arrive, they are processed by the 1st non-uniform partition in time domain. For frames more than L_1/L, the previous frames are stored in delay buffer for the 2nd non-uniform partitioned filter to process these. The 2nd non-uniform partitioned filter will be implemented using uniform partitioned convolution. This process will be repeated for each partition. The delay buffers will be needed to store input frames for each partition based on the partition length. These buffers are common for calculating $y_L(n)+jy_R(n)$ and $y_C(n)$ as input channels are common for both. The uniform partitioned convolution contains single FFT, single IFFT and complex frequency multiplication blocks. The number of these blocks are derived based on partition length and frame lengths. As all UPC blocks are executed in parallel, the computational complexity of overall system becomes maximum of computations needed for each UPC block. Obviously the implementation complexity of this approach is high. Also more memory is required to store long filter coefficients, their FFTs and to store input frames in delayed buffers. This can be managed with GPUs as the memory in the order of GigaBytes is available in these platforms.

RESULTS AND DISCUSSION

This study explains the architecture of heterogeneous parallel computing platforms, partitioning scheme used in implementing proposed algorithm and results achieved on AMD based GPUs.

Parallel computing architecture: Heterogeneous computing involves the use of various computational units, usually a general purpose processing unit such as CPU or GPU or DSP processors. To obtain optimal performance, suitable hardware is required to schedule various tasks based on the developer's choice. OpenCL (Open Computing Language) is an open and royalty free parallel computing API used to implement kernel programs on GPUs and these programs can be enabled by Host software running on CPU. As shown in Fig. 4, the OpenCL Device consists of one or more compute units. Each compute unit in turn contains many processing elements, usually called SIMD units. These SIMD units are responsible for processing of data provided by Host [16-20].

In general, OpernCL execution model contains two components, i.e., Host and kernel. Host creates the kernel context and based on this this context, it creates the required program objects, command queues and memory objects. It sets the kernel argument list and prepares the command queue either in in order or out of order based on kernel execution. If command queue is on out of order, then synchronization is required while calling the kernels. After this, Host calls the kernels NDRangeKernel or EnqueueTask based on requirement of data parallelism or task parallelism. The other component Kernel contains the actual openCL programs which are optimized using the global work group size[16-20].

OpenCL memory is divided into 4 regions shown in Fig. 4. Global memory is accessible to both host and OpenCL device. All host allocated memory objects resided in global memory. A part of global memory is dedicated for constants and kernel has only read access for it but host has read and write access because host allocates these objects. Local memory is a region of memory used for data sharing across work items in work group. Private memory is a region that is accessible to only one work item. Generally, data must be explicitly move from host to global to local and back. Host will wait until the kernel executes entire program to read the output memory objects in data parallelism. But in case of task

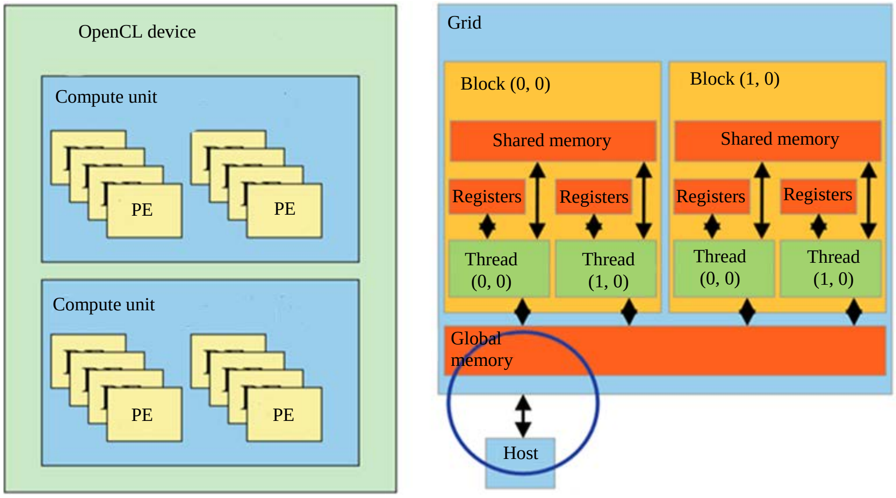

Fig.4: [Left] OpenCL device model and [Right] memory model

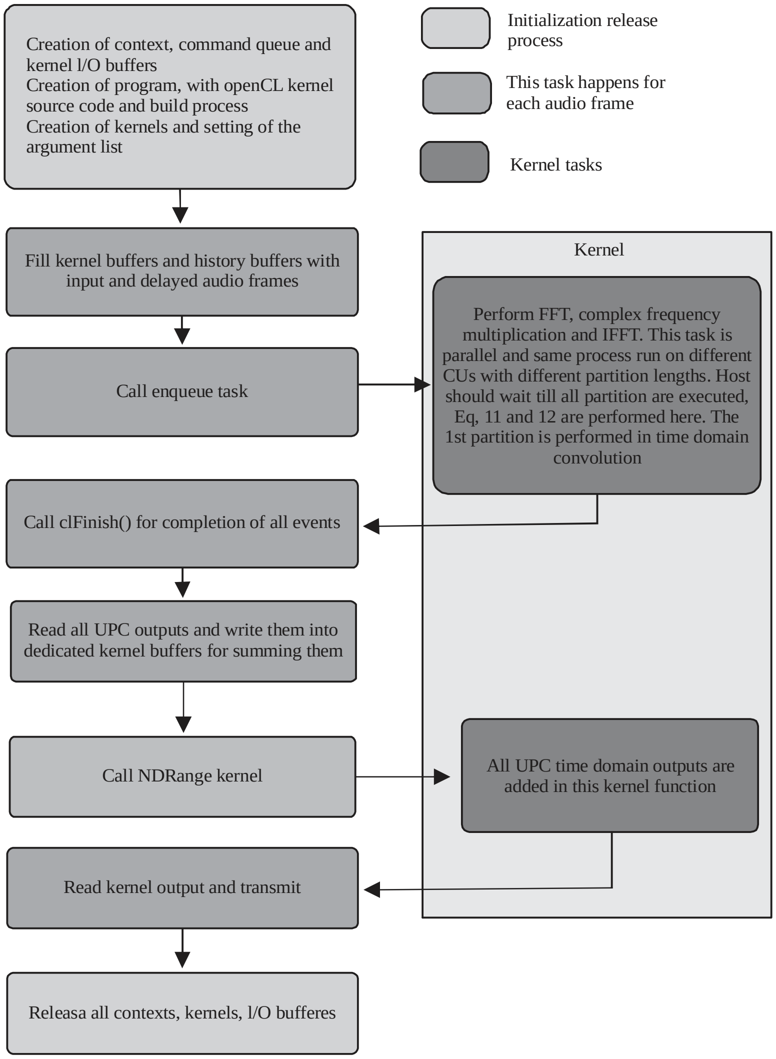
Creation of context, command queue and kernel I/O buffers
Creation of program, with openCL kernel source code and build process
Creation of kernels and setting of the argument list
Initialization release process
This task happens for each audio frame
Kernel tasks
Fill kernel buffers and history buffers with input and delayed audio frames
Kernel
Perform FFT, complex frequency multiplication and IFFT. This task is parallel and same process run on different CUs with different partition lengths. Host should wait till all partition are executed, Eq, 11 and 12 are performed here. The 1st partition is performed in time domain convolution
Call enqueue task
Call clFinish() for completion of all events
Read all UPC outputs and write them into dedicated kernel buffers for summing them
Call NDRange kernel
All UPC time domain outputs are added in this kernel function
Read kernel output and transmit
Releasa all contexts, kernels, I/O bufferes

Fig. 5: Flow diagram for the design of proposed approach

parallelism, these can be used as non-blocked if required based on the order and synchronization of tasks[16-20].

Partitioning scheme: To reduce the computational complexity and to keep output latency low, the partitioning scheme should follow some systematic way from lower partition length to possible higher partition length. Gardener and Garcia have proposed 128*2, 256*2,512*2,1024*2,2048*2,4096*2 and 128*14, 1024*14 for filter length of 16128, respectively[9, 10]. As initial partition length is 128, this ensures that latency is low. But there is difference in system cost. The solution of Gardener needs 12 CUs (compute units) whereas Garcia's method needs 28 CUs. As against this, Garcia's solution consumes less power as maximum partition length is 1024 but Gardener solution needs 4096 length. So there is always tradeoff between system cost and computational power.

Table 1 gives partitioning scheme followed to implement proposed method for filter lengths from 4096-65536 with step size of 4096. For example, filter length of 16384 requires 4 UPCs of 512 each, 6 UPCs of 1024 length each and 4 UPCs of 2048 each (16384 = 4*512+6*1024+4*2048). This design was done to make sure that each non-uniform partition is still partitioned into uniform partition with filter length of L = 128 during implementation. This ensures that the latency of the system is obtained as 128/44.1 = 2.9 msec at sampling frequency of 44.1 kHz. Also with this approach, the cost of the system is low as required CUs are low.

Design of proposed approach: Figure 5 shows the flow diagram of high level design of the proposed approach. In this, Host basically initializes all the kernels, required OpenCL kernel I/O buffers, the OpenCL context and the builds the kernel source code. Then it sets all the required

Table 1: Partitioning of impulse response length in mixed non-uniform partitioned convolution

Filter length (M)	Partitions ×512	Partitions ×1024	Partitions ×2048	Partitions ×4096	Total partitions
4096	4	2	0	0	6
8192	4	6	0	0	10
12288	4	6	2	0	12
16384	4	6	4	0	14
20480	4	6	6	0	16
24576	4	6	8	0	18
28672	4	6	8	1	19
32768	4	6	8	2	20
36864	4	6	8	3	21
40960	4	6	8	4	22
45056	4	6	8	5	23
49152	4	6	8	6	24
53248	4	6	8	7	25
57344	4	6	8	8	26
61440	4	6	8	9	27
65536	4	6	8	10	28

kernel argument list. After this, it writes the input frame contents into kernel Input buffers and calls the clEnqueueTask function. Upon calling this function, the OpenCL Kernel runs on GPU. On GPU, the kernel function is implemented basically to perform FFT of input audio frame, complex frequency multiplication and IFFT. These blocks are part of uniform partitioned convolution. The initial partitioned filter is implemented using time domain convolution.

The count of non-uniform partitions are based on length of impulse response and the partitioning scheme followed as per Table 1. So, all non-uniform partitions are executed on different CUs in parallel within GPU. The execution time of kernel function running on each CU depends on the partition length. Host waits using clFinish function till execution of all kernels are completed. The clEnqueueTask basically performs the task parallelism operation and after completion of this, it is required to add all outputs of kernel functions, i.e., all UPC outputs. To do this, Host writes all UPC outputs in one dedicated kernel buffer and calls clNDRangeKernel, basically meant for data parallelism operation. This kernel function does the summing of all UPC outputs and time domain output to produce the final outputs, $y_L(n)+jy_R(n)$ and $y_C(n)$. In this case, Host does not need to wait for Kernel completion because data parallel kernel returns after completion of kernel execution only. Then Host transmits the CTC outputs for rendering. Once, all the audio frames are processed successfully, host releases all the kernels, allocated kernel I/O buffers and various contexts.

To proceed with performance tests, impulse responses from various source locations to receiver positions are needed. An audio room of size 5×4×3.5 m^3 was used in this experiment. A dummy head of width 30 cm was used to record impulse response. Three receiver positions fixed to receive sounds.

Receiver locations (Dummy Head):

- Left ear position: 2.35×2×1
- Center position: 2.5×2×1
- Right ear position: 2.65×2×1

Source locations:

- Source 1: 4×3.5×1
- Source 2: 2.5×3.5×1

The audio room was fully covered with absorption material. The measurements were done at 44.1 kHz sampling frequency. The impulse response from source 1 to left ear is corresponding to $h_{LL}(n)$ and response from source 1 to center position is corresponding to $h_{LC}(n)$ and so on. The responses were measured for all filter lengths ranging from 4096-65536. The inverse of these responses were calculated using frequency domain method and were used in experiments. The details of hardware used in experiments are given below:

- Processor: AMD FX-4100 Quad-core processor, 3.6GHz
- GPU: Bonaire (AMD Radeon Graphics Processor-R7 2000 series)
- Approximate memory: 3GB
- Active CUs: 12
- Debugging tools: Microsoft Visual C++and CodeXL

AMD Radeon 7900 series based BONAIRE GPU is used to develop proposed algorithm. This GPU has 12 active CUs. Win 10 operating system was used for experiments. Microsoft Visual C++ and CodeXL were used for Host and Kernel developments, respectively.

Latency tests: One of the main advantages of using proposed method is zero output latency. Always the original method, i.e., time domain convolution provides zero delay and this is reference approach. To measure this latency, a multi-channel sweep signal of 44.1 kHz was provided as input to implemented CTC system shown in Fig.1. The 1st CTC output $y_L(n)$ was implemented using time domain convolution and proposed method. The outputs of both methods were recorded and compared as shown in Fig. 6. The filters used in this experiment are of length 4096. Here the 1st partition was implemented using uniform partitioned convolution with frame size of 128 samples.

As shown in Fig 6a, the proposed method provides a latency of 2.9 msec approximately when compared to original method, i.e., time domain convolution. This is due to appending of 128 zeroes to the 1st non-uniform partitioned impulse response. This results 128/44.1 kHz = 2.9 msec.

In another experiment, 1st non-uniform partition was implemented using time domain convolution

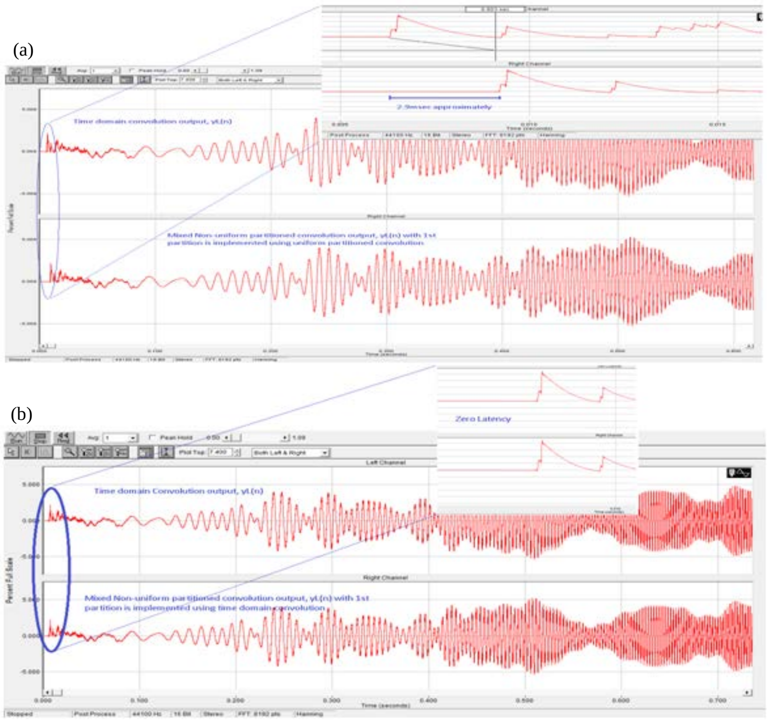
(a)
2.9msec approximately
Right Channel
Time (seconds)
Time domain convolution output, yL(n)
Mixed Non-uniform partitioned convolution output, yL(n) with 1st partition is implemented using uniform partitioned convolution
(b)
Zero Latency
Left Channel
Time domain Convolution output, yL(n)
Right Channel
Mixed Non-uniform partitioned convolution output, yL(n) with 1st partition is implemented using time domain convolution
Percent Full Scale
5.000
0.000
-5.000
0.000
0.100
0.200
0.300
0.400
0.500
0.600
0.700
Time (seconds)
Stopped
Post Process
44100 Hz
16 Bit
Stereo
FFT: 8192 pts
Hanning

Fig.6: Latency tests for (a) [Top] Time domain vs mixed non-uniform partitioned convolution with 1st partition being implemented using time domain (b) [Bottom] Time domain vs mixed non-uniform partitioned convolution with 1st partition being implemented using uniform partitioned convolution. The CTC output, $y_L(n)$ was shown in the comparison

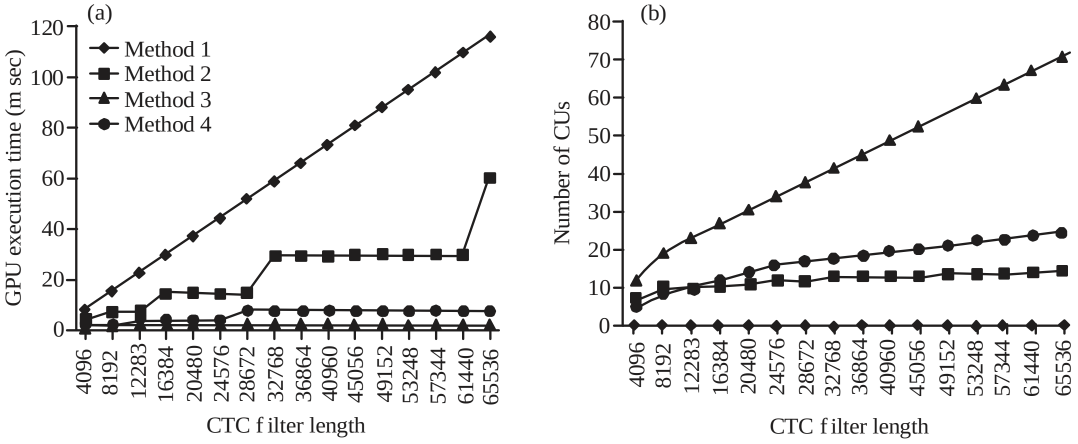

Fig. 7: (a)[Left] Comparison of GPU Execution Time in msec. (b) [Right] Comparison of Cus, Method 1: Uniform Partitioned Convolution, Method 2: Gardener approach; Method 3: Garcia approach, Method 4: Mixed Non-uniform partitioned convolution (Proposed approach)

for 1st CTC output $y_L(n)$ and outputs were recorded. The comparison was shown in Fig. 6b. The output latency is zero which is actually desired result. This won’t result additional increase in computational complexity because all partitions are executed in parallel and the computational complexity of higher partition is more compared to that of initial partition (Fig. 7).

Performance tests: The proposed algorithm was implemented as described in sub-section Design of Proposed approach. Host fills the kernel buffers and FFT delayed frames for each frame and initiates the kernel call using clEnqueueTask and clNDRangeKernel. The kernel functions were implemented in optimum way using vector based OpenCL approach. The execution time in msec was measured at various filter lengths using proposed algorithm. The same measurements were done for reference methods and the comparison was shown in Fig.7 (left hand side). The partitioning scheme mentioned in Table 1 was followed for implementation of proposed algorithm and the partitioning of reference methods is based on approaches provided in references[9, 10]. The computational cost for each method in terms of computations units is also compared in Fig.7 (right hand side).

The 1st method is uniform partitioned convolution. In this, the parallel implementation of all complex frequency multiplication blocks are is not done due to uniform partitioned approach. Additional complexity comes into picture to add all complex multiplications outputs, which is serial anyhow. Due to these dependencies, the computational complexity of this approach is high. So it is suitable at medium filter lengths but it is not preferable to use at very long filter lengths. On the other hand, single CU is sufficient to implement this method. Hence this method is most suitable to implement on general DSP processors itself.

The 2nd method is Gardener’s approach. In this method, the partitioning was done like for every two partitions, the partition size becomes doubled, starting from lower partition length of 128 coefficients. Due to this, at very long filter lengths, the higher partition size becomes increase so that the execution time will get increase at these lengths. But the number of CUs are quite low due to higher size in partitions. The 3rd method is based on Garcia’s approach. Garcia proposed partitioning scheme in which each impulse response is weighted sum of 128 and 1024 lengths. Due to this, execution time is less but the required number of Compute units are more to support very long filter lengths.

The proposed partitioning scheme contains an increase in partition lengths as well as increase in number of such partitions. The benefit of this approach is less execution time will be obtained and at the same time, there is scope to get less number of compute units. Clearly, the maximum partition length in proposed approach is 4096 and 28 CUs are needed for 65536 filter length in worst case.

The results are clearly resembling that the proposed method is best suitable for implementing audio CTC section at very long filter lengths, both in terms of execution time and number of compute units.

CONCLUSION

To address computational complexity and latency issues of very long filters in OSD audio CTC section, an efficient algorithm calledm mixed non-uniform partitioned convolution was proposed in this study. The ability of this approach is to provide zero output latency for long filters. The approach was implemented on AMD based Bonaire GPU platform. The performance was measured for various filter lengths from 4096-65536. The design and the way of implementation was described in detail on how to utilize the openCL platforms to bring down the computations. The computational performance comparison with existing methods clearly indicate that the proposed method is very good for long filters in audio CTC section. This work could be extended to multi-channel based audio CTC systems. In this study, it is interesting to see the variation of computational power as the channel count increases.

REFERENCES

01. Otani, M. and S. Ise, 2006. Fast calculation system specialized for head-related transfer function based on boundary element method. J. Acoust. Soc. Am., 119: 2589-2598.
02. Wang, L., F. Yin and Z. Chen, 2010. A stereo crosstalk cancellation system based on the common-acoustical pole/zero model. EURASIP J. Adv. Signal Process., Vol. 1,.
03. Akeroyd, M.A., J. Chambers, D. Bullock, A.R. Palmer, A.Q. Summerfield, P.A. Nelson and S. Gatehouse, 2007. The binaural performance of a cross-talk cancellation system with matched or mismatched setup and playback acoustics. J. Acoust. Soc. Am., 121: 1056-1069.
04. Yang, J., W.S. Gan and S.E. Tang, 2004. Development of virtual sound imaging system using triple elevated speakers. IEEE. Trans. Consum. Electron., 50: 916-922.
05. Lyons, R.G., 2010. Understanding Digital Signal Processing. 3rd Edn., Prentice Hall Publications Inc., New Jersey, USA.,.
06. Analog Devices Inc., 2010. ADSP-214xx SHARC processor hardware reference manual. Analog Devices Inc., Norwood, Massachusetts.
07. Battenberg, E. and R. Avizienis, 2011. Implementing real-time partitioned convolution algorithms on conventional operating systems. Proceedings of the 14th International Conference on Digital Audio Effects, September 19-23, 2011, Paris, France, pp: 248-235.

08. Torger, A. and A. Farina, 2001. Real-time partitioned convolution for Ambiophonics surround sound. Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575), October 24, 2001, IEEE, New Platz, New York, pp: 195-198.
09. Guillermo, G., 2002. Optimal filter partition for efficient convolution with short input/output delay. Proceedings of the AES 113th International Conference, October 1, 2002, Stanford University, Stanford, California, pp: 2660-2660.
10. Gardener, W.G., 1995. Efficient convolution without input-output delay. J. AES., 43: 127-136.
11. SreenivasaRao, C., N.V.K. Mahalakshmi and D. VenkataRao, 2012. Real-time DSP implementation of audio crosstalk cancellation using mixed uniform partitioned convolution. Signal Process. Int. J. (SPIJ.), 6: 118-127.
12. Rao, C.S., D.V. Rao and S. Lakshminarayana, 2015. Design and implementation analysis of OSD based audio crosstalk cancellation with multi-channel inputs on DSP processors. Indian J. Sci. Technol., 8: 419-431.
13. SreenivasaRao, C., D. VenkataRao and S. Lakshminarayana, 2015. An efficient implementing solution for three channel OSD based audio crosstalk cancellation with stereo inputs. Int. J. Applied Eng. Res., 10: 1995-2011.
14. SreenivasaRao, C. and D. VenkataRao, 2013. Real-time implementation of multi-channel audio crosstalk cancellation using mixed single frequency delay line filtering algorithm. Int. J. Mod. Eng. Res., 3: 1088-1096.
15. SreenivasaRao, C., R. Udayalakshmi and P. Jeyasingh, 2012. Fast implementation of audio crosstalk cancellation of audio crosstalk cancellation on DSP processors. Proceedings of the AES 45 Conference, March 1-4, 2012, Helsinki, Finland, pp: 56-63.
16. Gaster, B.R. L. Howes, D. Kaeli, P. Mistry and D. Schaa, 2012. Heterogeneous Computing with OpenCL. 2nd Edn., Morgan Kaufmann Publisher Inc., San Francisco, CA, USA., ISBN: 978-0-12-387766-6.
17. Kim, D., J. Lee, J. Lee, I. Shin, J. Kim and S.E. Yoon, 2013. Scheduling in heterogeneous computing environments for proximity queries. IEEE. Trans. Visualization Comput. Graphics, 19: 1513-1525.
18. Lee, C., W.W. Ro and J.L. Gaudiot, 2012. Cooperative heterogeneous computing for parallel processing on CPU/GPU hybrids. Proceedings of the 2012 16th Workshop on Interaction between Compilers and Computer Architectures (INTERACT), February 25, 2012, IEEE, New Orleans, Louisiana, pp: 33-40.
19. Nakamura, T., T. Izuka and A. Asahara, 2012. The OpenCL Programming Book Revised for OpenCL 1.2. Fixstars Publishers, Sunnyvale, California,.
20. Proakis, J.G., 1996. Digital Signal Processing: Principles Algorithms and Applications. 3rd Edn., Pearson Education India, India,.