Lorente Giner, J.; Ferrer Contreras, M.; Diego Antón, MD.; Gonzalez, A. (2014). GPU Implementation of multichannel adaptive algorithms for local active noise control. IEEE Transactions on Audio, Speech and Language Processing. 22(11):1624-1635. doi:10.1109/TASLP.2014.2344852.

GPU Implementation of Multichannel Adaptive Algorithms for Local Active Noise Control

Jorge Lorente, *Student Member, IEEE,* Miguel Ferrer, *Member, IEEE,* Maria de Diego, *Senior Member, IEEE,* and Alberto González, *Senior Member, IEEE*

***Abstract*—Multichannel active noise control (ANC) systems are commonly based on adaptive signal processing algorithms that require high computational capacity, which constrains their practical implementation. Graphics Processing Units (GPUs) are well known for their potential for highly parallel data processing. Therefore, GPUs seem to be a suitable platform for multichannel scenarios. However, efficient use of parallel computation in the adaptive filtering context is not straightforward due to the feedback loops. This paper compares two GPU implementations of a multichannel feedforward local ANC system working as a real-time prototype. Both GPU implementations are based on the filtered-x Least Mean Square algorithms; one is based on the conventional filtered-x scheme and the other is based on the modified filtered-x scheme. Details regarding the parallelization of the algorithms are given. Finally, experimental results are presented to compare the performance of both multichannel ANC GPU implementations. The results show the usefulness of many-core devices for developing versatile, scalable, and low-cost multichannel ANC systems.**

***Index Terms*—Active Noise Control, Graphics Processing Unit, filtered-x Least Mean Square**

I. Introduction

ACTIVE noise control (ANC) [1], [2], [3] is a field that combines digital signal processing techniques with traditional acoustics. ANC systems are based on the principle of destructive interference between a disturbance sound field called primary noise and a secondary sound field that is generated by controlled secondary sources called actuators. The goal is to cancel, or at least minimize, the primary noise signal. In order to cancel the primary noise, the ANC system commonly uses adaptive algorithms [4] to generate the secondary sound field from a reference signal that is correlated with the primary noise. For this purpose, the noise signal is monitored at a specific spatial point by a sensor called error sensor. Therefore, cancellation is only achieved around that error sensor with the spatial limit being approximately $\lambda/10$, where λ is the wavelength of the highest undesired frequency. ANC systems can be extended to multichannel ANC systems by employing multiple error sensors and multiple secondary sources, thereby extending the control zone [5].

The filtered-x Least Mean Squares (FxLMS) algorithm [6] and its multichannel version [5] are the most widely used adaptive filtering strategies applied to single or multiple-channel adaptive noise cancellers for ANC applications. Two of the best-known applications of the multichannel filtered-x LMS algorithm are the control of noise inside enclosures such as in cars [7], [8] and in flight cabin interiors [9]. These multichannel systems require a high computational capacity and an even greater capacity when massive control systems are considered (a high number of channels, defining one channel for each pair of error sensor - secondary source). Moreover, the number of filtering operations increases significantly with the number of channels. Thus, in practice the computational cost is one of the main bottlenecks of these multichannel ANC systems. On the other hand, Graphics Processing Units (GPUs) are highly parallel programmable co-processors that provide massive computation when the needed operations are properly parallelized. Hence, GPUs seem suitable for multichannel ANC applications where the processing of each channel could be done in parallel. Therefore, in contrast to traditional implementations that are based on conventional hardware devices such as Digital Signal Processors (DSPs) [10] and dedicated hardware, this paper presents a multichannel ANC prototype over a GPU platform.

Using the NVIDIA programming language CUDA (Compute Unified Device Architecture) [11], GPUs are being employed in most engineering fields that require intensive computation. In [12], [13] some signal processing applications take advantage of these opportunities. A general overview of audio signal processing on the GPU is given in [14] and [15]. Moreover, there are many recent contributions that leverage GPUs to accelerate acoustic and audio simulations or real-time applications like: room acoustics [16], acoustics likelihood computation [17], speech recognition [18], RIR (Room Impulse Response) reshaping [19], beamforming [20], sound localization [21] or wave-field synthesis [22]. Furthermore, the filtering on GPU where real-time filtering of multiple data is carried out concurrently has recently been introduced in [23], [24]. However, very few publications [25], [26], [27] deal with the GPU implementation of real-time acoustic applications based on adaptive filtering. This is because the data transactions among GPU, CPU, and the audio card are critical for the real-time performance. Specifically, in [25], a multichannel acoustic echo canceller on GPU was proposed. In [26] and [27], a single-channel and a multichannel ANC systems were implemented on a GPU.

In this work, we compare two different GPU implementations of a multichannel ANC prototype based on the FxLMS algorithm [28]. The Frequency-domain Partitioned Block LMS

The authors are with the Institute of Telecommunications and Multimedia Applications (iTEAM), Universitat Politècnica de València, València, Spain (e-mail: {jorlogi, mferrer, mdediego, agonzal}@iteam.upv.es).

This work has been supported by European Union ERDF and Spanish Government through TEC2012-38142-C04 project, and Generalitat Valenciana through PROMETEO/2009/013 project.

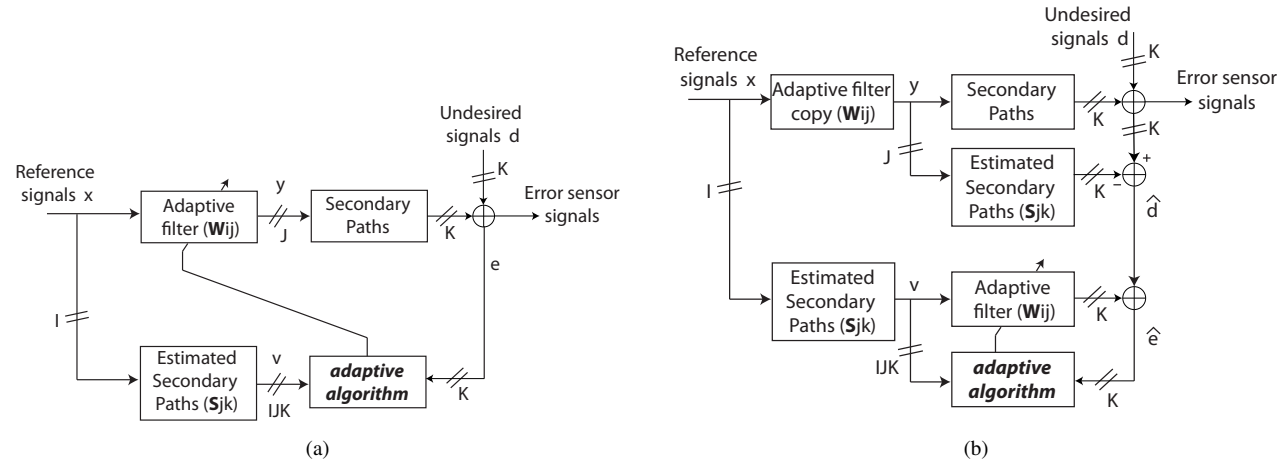


Fig. 1: Block diagrams of the multichannel ANC system based on the (a) FPBFxLMS and (b) FPBMFxLMS algorithms.

algorithm (FPBLMS) [28], [29] is implemented using two different filtering schemes: the conventional scheme and the modified filtered-x scheme. Throughout this paper, the FPBLMS algorithm based on the conventional filtered-x scheme will be referred to as FPBFxLMS, whereas the FPBLMS algorithm based on the modified filtered-x scheme will be referred to as FPBMFxLMS. The FPBFxLMS algorithm was discussed in [26] and [27] for a single-channel and a multichannel ANC system, respectively. A similar algorithm was introduced in [30] for a single-channel ANC system. The FPBMFxLMS was presented in [31] for a multichannel ANC system.

The use of the Frequency-domain Block-based filtered-x LMS [32], [33], is motivated by the following reasons. It allows a fast implementation because the parallel resources of a GPU are better exploited when working with blocks of samples instead of sample-by-sample. Moreover, most of the common audio cards work with block data buffers. In a second stage, and taking into account that the adaptive filters could be larger than the block size, the adaptive filters are partitioned [34], [35]; therefore, the delay is reduced and the parallelization is improved by performing the adaptation of each partition of the filters simultaneously. This leads to the use of the FPBLMS algorithm with a suitable filtering scheme for ANC.

It is well known that the modified scheme provides better convergence performance than the conventional filtering scheme [36], but it is more demanding from a computational cost point of view. However, making use of the parallelism of GPU computing, the proposed implementation of the modified scheme can meaningfully deal with the increase of computational burden. Therefore, this work discusses the advantages and disadvantages of the GPU implementation for both schemes.

This paper is organized as follows: section 2 outlines the two algorithms. The ANC prototype is described in section 3, while the GPU implementation is explained in section 4. Section 5 presents the experimental results, and section 6 presents the conclusions.

II. Description of the Algorithms

This section focuses on illustrating the FPBFxLMS and FPBMFxLMS algorithms. The block diagram of the multichannel ANC system based on the two algorithms is depicted in Fig. 1. A generic multichannel ANC system with I reference signals, J secondary sources, and K error sensors (I:J:K) has been considered. In this work, samples are processed by blocks of size B. L is the length of the adaptive filters, and M is the length of the FIR filters that model the estimated secondary paths. If L and M are higher than B, we have to split up both the adaptive filters and the estimated secondary paths into F and P partitions, respectively [4]. Thus, the algorithm works simultaneously with all the partitions of size B. Furthermore, the sub-index and super-index of the following notation denote block iteration and the number of the partition, respectively.

With regard to the estimation of the secondary path, there are techniques that are based on the hypothesis that the secondary path model does not have to be accurate and can be represented by a delay (delayed-x LMS) [37]. These techniques are used in applications with variable systems in which rapid reaction is also of utmost interest. Since the application of this paper is set in a listening room with a fixed response, the secondary paths were previously modeled by FIR filters with an accurate estimation.

A. The Frequency-domain Partitioned Block Filtered-x LMS algorithm (FPBFxLMS)

The notation in Table I will be used to describe the algorithms. According to the notation, the adaptive filter output is calculated as follows

$$\mathbf{Y}j_n = \sum_{i=1}^{I}\sum_{f=1}^{F} \mathbf{W}ij_n^f \circ \mathbf{X}i_{n-f+1}, \tag{1}$$

where $\mathbf{X}i_n = \mathrm{FFT}[\mathbf{x}i_{Bn-1} \quad \mathbf{x}i_{Bn}]$, $\mathbf{W}ij_n^f$ is the FFT of size $2B$ of the fth partition of the coefficients of the adaptive filter $\mathbf{w}ij$ at the nth block iteration, and $\circ$ denotes the element-wise

TABLE I: Notation of the description of the algorithms

I	Number of reference signals (reference sensors)	J	Number of secondary sources (actuators)
K	Number of error signals (error sensors)	B	Block size
L	Length of the adaptive filters	F	L/B, number of partitions of the adaptive filters
M	Length of the FIR filters that model the estimated secondary paths	P	M/B, number of partitions of the estimated secondary paths

$x_i(t)$	ith reference signal at time instant t	$\mathbf{x}i_{Bn}$	$[x_i(Bn) \quad x_i(Bn-1) \quad \dots \quad x_i(Bn-B+1)]$
$y_j(t)$	jth actuator signal at time instant t	$\mathbf{y}j_{Bn}$	$[y_j(Bn) \quad y_j(Bn-1) \quad \dots \quad y_j(Bn-B+1)]$
$e_k(t)$	kth microphone signal at time instant t	$\mathbf{e}k_{Bn}$	$[e_k(Bn) \quad e_k(Bn-1) \quad \dots \quad e_k(Bn-B+1)]$

$\mathbf{s}jk$	M-length estimation of the secondary path that links the jth secondary source with the kth error sensor
$\mathbf{S}jk^p$	FFT of size $2B$ of the pth partition of the acoustic path $\mathbf{s}jk$
$\mathbf{w}ij$	Coefficients of the adaptive filter of length L that link the ith reference signal with the jth secondary source
$\mathbf{W}ij_n^f$	FFT of size $2B$ of the fth partition of the coefficients of the adaptive filter $\mathbf{w}ij$ at the nth block iteration

product of two vectors. The valid samples of the adaptive filter output $\mathbf{y}j_{Bn}$ are the last B samples of $\text{IFFT}\{\mathbf{Y}j_n\}$.

The filter coefficients are updated in the frequency domain by calculating the correlations between the reference signals ($\mathbf{X}i_n$) that are filtered through the estimated secondary paths ($\mathbf{S}jk^p$), $\mathbf{V}ijk_n$, and the error signals, $\mathbf{e}k_{Bn}$. To this end, the following operations are performed

$$\mathbf{V}ijk_n = \sum_{p=1}^{P} \mathbf{S}jk^p \circ \mathbf{X}i_{n-p+1}, \tag{2}$$

$$\widetilde{\boldsymbol{\mu}}ijk^f = \mathbf{E}k_n \circ \mathbf{V}ijk^{f*}, \tag{3}$$

where

$$\mathbf{E}k_n = \text{FFT}[\mathbf{0}_B \quad \mathbf{e}k_{Bn}]. \tag{4}$$

The update of the coefficients of each partition of the *ij*th adaptive filters is calculated as follows

$$\mathbf{W}ij^f(n) = \mathbf{W}ij^f(n-1) - \mu \sum_{k=1}^{K} \text{FFT}\{[\boldsymbol{\phi}ijk^f \quad \mathbf{0}_B]\}, \tag{5}$$

where μ is the step-size parameter, and the vector $\boldsymbol{\phi}ijk^f$ corresponds to the first B samples of the 2B-IFFT of the corresponding partition $\widetilde{\boldsymbol{\mu}}ijk^f$

$$\begin{aligned} \text{IFFT}\{\widetilde{\boldsymbol{\mu}}ijk^f\} &= [\boldsymbol{\phi}ijk^f \quad \bar{\boldsymbol{\phi}}ijk^f], \\ f &= n, n-1, \dots, n-F+1. \end{aligned} \tag{6}$$

B. The Frequency-domain Partitioned Block Modified Filtered-x LMS algorithm (FPBMFxLMS)

This section focuses on describing the FPBMFxLMS algorithm. The adaptive filter output is calculated as in Eq. (1).

The signals $\mathbf{Y}j_n$ are used to estimate the undesired signals in the frequency domain, obtaining $\widehat{\mathbf{D}}k_n$. To this end, the following operations are performed

$$\mathbf{YF}k_n = \sum_{j=1}^{J} \sum_{p=1}^{P} \mathbf{Y}j_{n-p+1} \circ \mathbf{S}jk^p, \tag{7}$$

$$\widehat{\mathbf{D}}k_n = \mathbf{E}k_n - \mathbf{YF}k_n. \tag{8}$$

Moreover, the estimated error signals in the frequency domain, $\hat{\mathbf{E}}k_n$, are obtained from

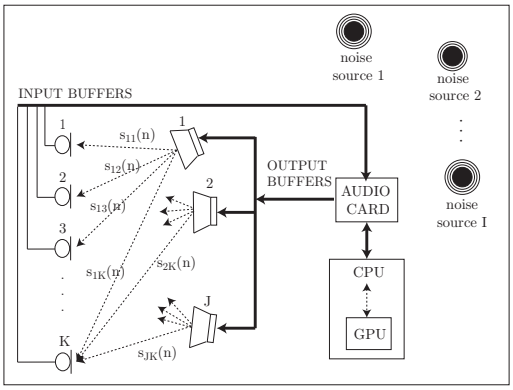


Fig. 2: Scheme of the multichannel ANC system.

$$\mathbf{V}ijk_n = \sum_{p=1}^{P} \mathbf{S}jk^p \circ \mathbf{X}i_{n-p+1}, \tag{9}$$

$$\mathbf{OUT}k_n = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{f=1}^{F} \mathbf{V}ijk_{n-f+1} \circ \mathbf{W}ij_n^f, \tag{10}$$

where

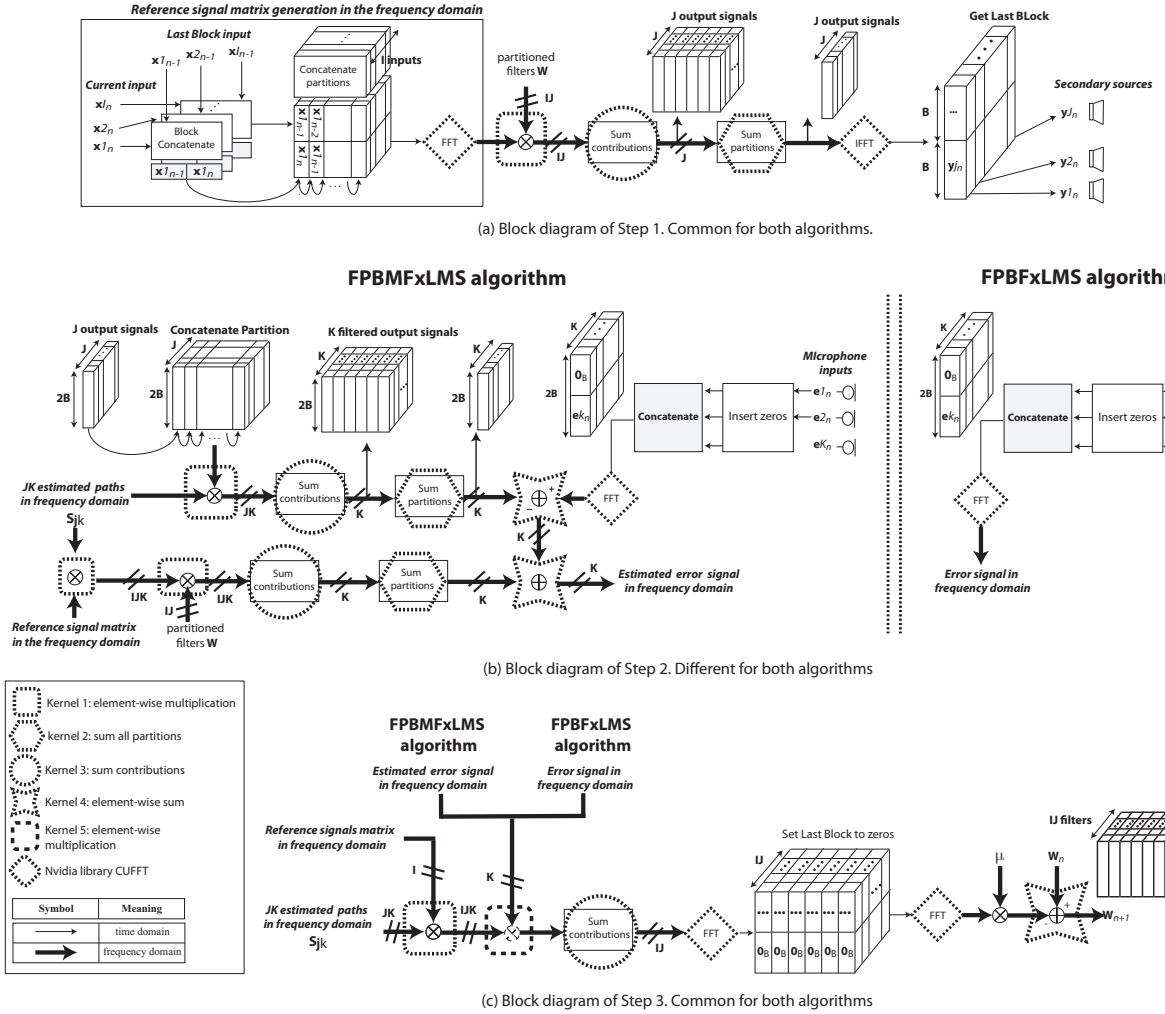
$$\widehat{\mathbf{E}}k_n = \widehat{\mathbf{D}}k_n + \mathbf{OUT}k_n. \tag{11}$$

Finally, the update rule of the frequency-domain filter coefficients is given by Eq. (5), where $\boldsymbol{\phi}ijk^f$, and therefore $\widetilde{\boldsymbol{\mu}}ijk^f$ (Eq. (3) and Eq. (6)) are calculated using the estimated error signal $\widehat{\mathbf{E}}k_n$ instead of the error signal $\mathbf{E}k_n$.

III. Prototype description

The multichannel ANC prototype is depicted in Fig. 2. For the hardware configuration, the GPU used is a GeForce GTX 580 with Fermi architecture. The CPU is an Intel Core i7 (3.07 GHz), and the audio card is a MOTU 24I/O. The MOTU audio uses the ASIO (Audio Stream Input/Output) driver to communicate with the CPU. The ASIO driver provides input/output buffers that are used to collect/send the current microphone and loudspeaker signals. The input buffers are linked to the microphones and the output buffers are linked to the loudspeakers. The operation of the prototype consists of three tasks that are executed in each iteration:

1) Collect the K input-data buffers of size B from the sensors and transfer them through the PCI-Express bus to the GPU.



(a) Block diagram of Step 1. Common for both algorithms.

(b) Block diagram of Step 2. Different for both algorithms

(c) Block diagram of Step 3. Common for both algorithms

Fig. 3: Block diagram of the GPU implementation of the FPBFxLMS and the FPBMFxLMS algorithms.

2) Carry out the corresponding algorithm on the GPU.
3) Save the output audio samples in the J output-data buffers and send them back to the CPU to be reproduced by the loudspeakers.

Two important parameters of the audio card are the sampling rate (f_s) and the block size (B). The block size describes the number of transferred discrete-time samples per iteration and thereby determines the latency of the algorithm. The latency which is defined as B/f_s, is the time spent to fill up the input-data buffers. We will refer to the latency throughout the paper as the buffering time (t_{buff}). The choice of the parameters B and f_s is critical for the performance of the system because there are two conditions that must be satisfied:

- **The real-time condition**. The application can work in real time if the following condition is satisfied: $t_{proc} < t_{buff}$, where t_{proc} is the execution delay measured from the moment the input-data buffer is sent to the GPU until the output-data buffer comes back to the CPU. This includes transfer delays between the CPU and the GPU and the data processing delay of the GPU.
- **The causality condition**. The algorithm needs to satisfy the condition $t_{buff}+\tau_s < \tau_n$ in order to perform properly [38], where τ_s is the maximum delay of the secondary paths that joins the actuators with the error sensors, and τ_n is the minimum delay of the paths that joins the noise source with the error sensors. This condition guarantees the causality of the system.

Causality is not a constraint when excitation is sinusoidal because of the deterministic nature of the signal, but causality has to be considered important in broadband control. In this paper, the multichannel ANC prototype is mounted in a listening room where both real-time and causality conditions are fulfilled by choosing the suitable distances and parameters. If the location of the noise source and the cancellation area do not offer the possibility to fulfill the causality conditions, the algorithm has to work with a lower block size, and consequently with a lower t_{buff}, in order to satisfy the causality condition.

With regards to the choice of the parameters, the audio card offers three sampling rates: 44.1, 44.8, and 96 kHz. We have chosen $f_s = 44.1$ kHz, which is the lowest rate, but it is a fairly high rate for the sounds involved. For the block size, the audio card offers values between $B = 16$ and 2048; however, because of the real-time condition, we have used values between $B = 256$ and 2048.

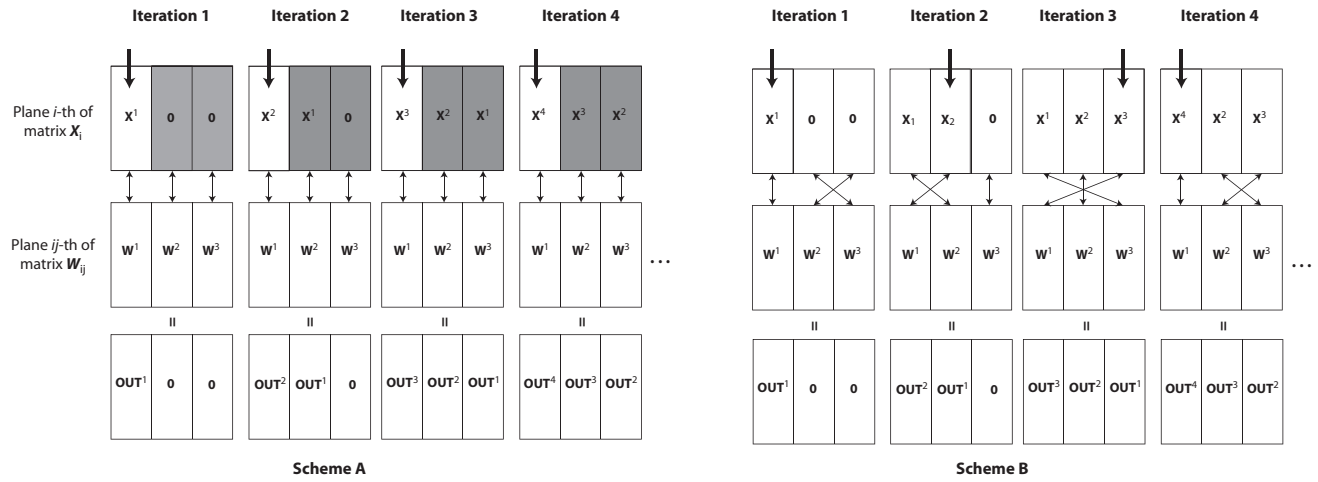


Fig. 4: Two different schemes of an element-wise multiplication for the case F=3.

It should be noted that the signal processing task is carried out by the GPU, while the CPU controls the data transfer between the input/output buffers and the GPU.

IV. GPU IMPLEMENTATION

This section describes the main issues involved in the GPU implementation of the real-time multichannel ANC prototype. The implementation is comprised of three steps: the output signal generation, the error signal calculation, and the update of the adaptive filters. Both algorithms generate the ANC outputs by filtering the reference signal through the adaptive filter, and they both update the adaptive filters using the error signals and the reference signal filtered through the estimated secondary paths. The main difference between the two algorithms is the error signal that is used to update the filters. The FPBFxLMS algorithm uses the signal picked up by the microphones as the error signal, while the FPBMFxLMS algorithm computes an estimate of the error signal. These three steps are implemented as follows:

- **S1** *ANC output generation*. This step is common to both algorithms and aims to calculate the ANC output signals $y_j(t)$. The operations of this step correspond to Eq. (1). The implementation and the CUDA kernels involved in it are shown in Fig. 3(a) and explained in section IV-A.
- **S2** *Error signal calculation*. This step is different for each implementation. While the conventional scheme uses Eq. (4) directly, the modified scheme calculates an estimate of the error signal (see Fig. 3(b)). The corresponding description is shown in Eq. (7)-(11).
- **S3** *Filter updates*. The update of the adaptive filter coefficients involve the implementation of Eq. (3)-(6). The details regarding the different steps and kernels are illustrated by Fig. 3(c) and section IV-A.

All the CUDA Kernels used in this work are explained in detail in the following subsection.

A. CUDA Kernels

Five optimized kernels were developed to achieve the most efficient performance of the algorithm. Moreover, the optimized NVIDIA FFT library (CUFFT) [11] was used to simultaneously carry out multiple one-dimensional FFTs.

- **K1** This kernel performs an element-wise multiplication of two matrices. There is a particular case in which the input matrix $\mathbf{X}_i$ is element-wise multiplied with the adaptive filter matrix $\mathbf{W}_{ij}$. In this case, there are two possible schemes for the multiplication, which are depicted in Fig. 4:
  1) The elements of the fth column are ordered in the same position in both matrices (scheme 'A').
  2) The elements of the fth column are not ordered in the same position in the two matrices (scheme 'B').

  Scheme 'A' shows the direct implementation, where the input-data matrix is ordered so that the elements of the fth column of the ith plane of matrix $\mathbf{X}_i$ are element-wise multiplied by elements of the fth column of the ijth plane of matrix $\mathbf{W}_{ij}$. This scheme has the disadvantage that all columns except one of the ith plane of matrix $\mathbf{X}_i$ are moved at each iteration, so it involves a copy of $2B(F-1)$ elements in GPU memory at each iteration. The copied data is represented in grey, and the current input buffer is marked by an arrow. Scheme 'B' shows an optimized multiplication, where the current input buffer is placed in the corresponding column avoiding GPU memory transactions. Therefore, in order to achieve the same final result in both schemes, we have to redefine the thread memory access of the GPU for multiplying the corresponding elements of the two matrices.
  This kernel launches a three-dimensional grid of three-dimensional blocks of threads. The dimensions of the blocks are (x,y,z)[1] threads, and the dimensions of the grid are (F/x, 2B/y, IJ/z) blocks. The kernel uses each thread to process each sample; thus, each thread will perform a complex multiplication between elements of each matrix.
- **K2** This kernel is launched to simultaneously reduce the F columns of each plane to a single one. The kernel is depicted in Fig. 5(a). It uses a three-dimensional grid

[1] x,y,z are selected according to the restrictions of the CUDA architecture of the device. In our case, this is the Fermi device [11]

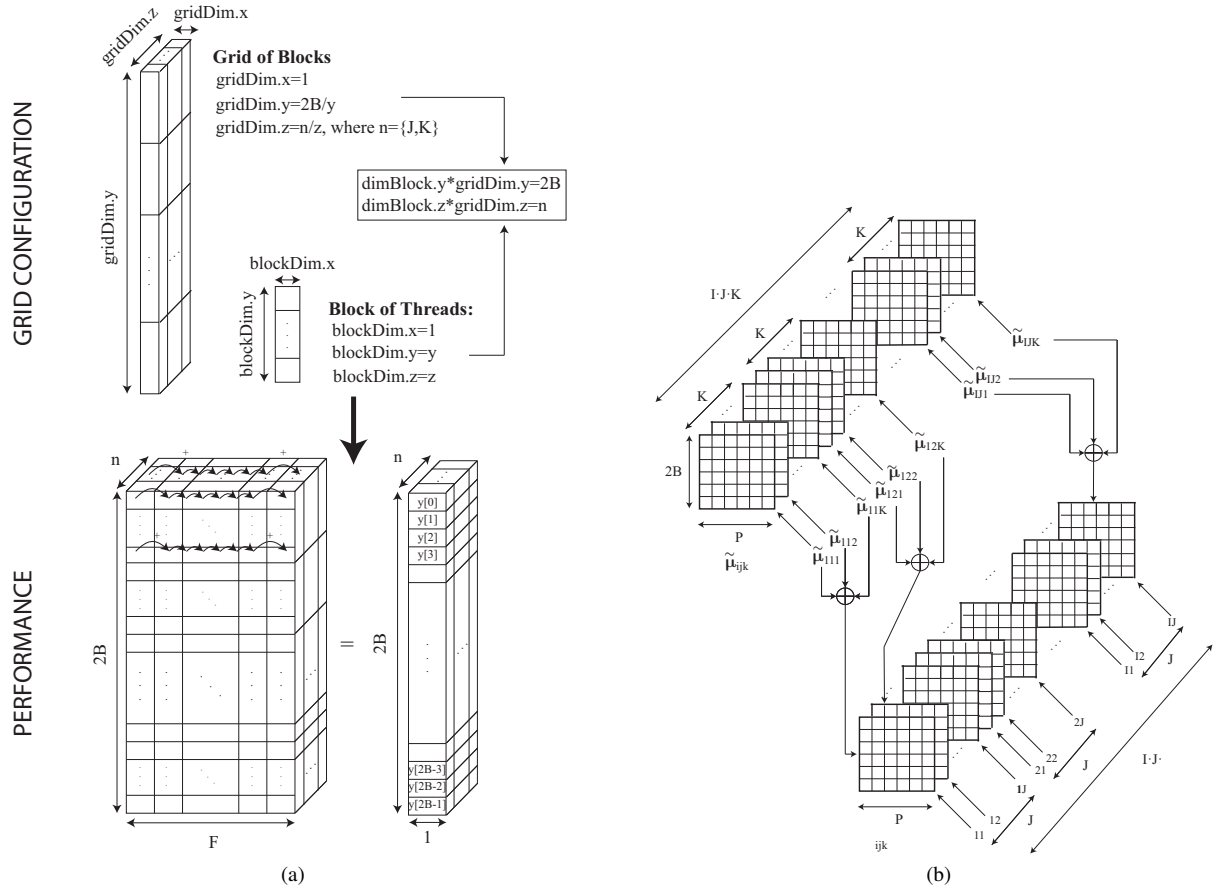


Fig. 5: Representations of (a) the CUDA kernel 2 and (b) the CUDA Kernel 3.

of blocks, where the dimension of the blocks is (1, y, z), and the dimension of the grid is (1, 2B/y, n/z). The kernel launches $2B \cdot n$ threads in total. Each thread carries out F sums. The result is a $2B \times n$ matrix where each element of the $2B$-dimension columns contains the reduction sum of each row. Note that n is the number of planes of $2B \times F$ elements of the matrix involved.

K3 This kernel performs a sum of m planes with the same sub-index. An example of the performance of the kernel is depicted in Fig. 5(b), where the matrix $\widetilde{\boldsymbol{\mu}}$ that has IJK planes results in a matrix with IJ planes. Therefore, in this case, a sum of K planes ($m = K$) with the same ij sub-indexes is performed.
This kernel launches $2B \cdot P$ threads divided into a grid of P blocks, where each block has $2B$ threads. Each thread performs the sum of m elements.

K4 This kernel has the same thread configuration as Kernel 1, but each thread performs a sum instead of a multiplication.

K5 This kernel is launched like Kernel 1. The difference in this case is that the kth plane of the error matrix ($\widehat{\mathbf{E}}k$) used in the dot product is a column vector instead of a matrix. Therefore, each column vector of the ijkth plane of matrix $\mathbf{V}ijk$ is element-wise multiplied by the same column vector $\widehat{\mathbf{E}}k$.

Fig. 6: Photograph of the ANC prototype using a 1:2:2 configuration.

V. Results

Several experiments were performed to validate the ANC systems and to compare the two algorithms. The experiments were carried out using the prototype described in section

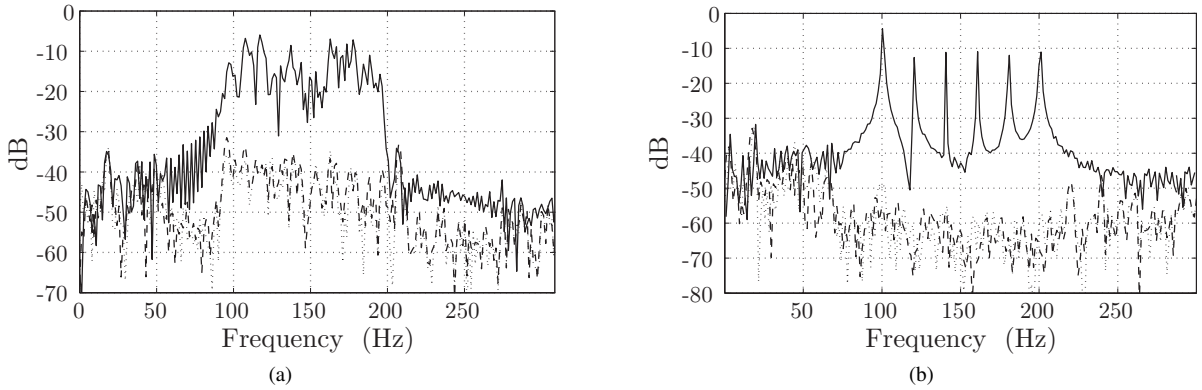


Fig. 7: Power spectral density of the average of the signals measured at the error sensors before (solid line) and after the ANC system operation by using the FPBFxLMS algorithm (dotted line) or the FPBMFxLMS algorithm (dashed line). The disturbance noise used in (a) is a band-limited white noise, and in (b) is a periodic tonal noise.

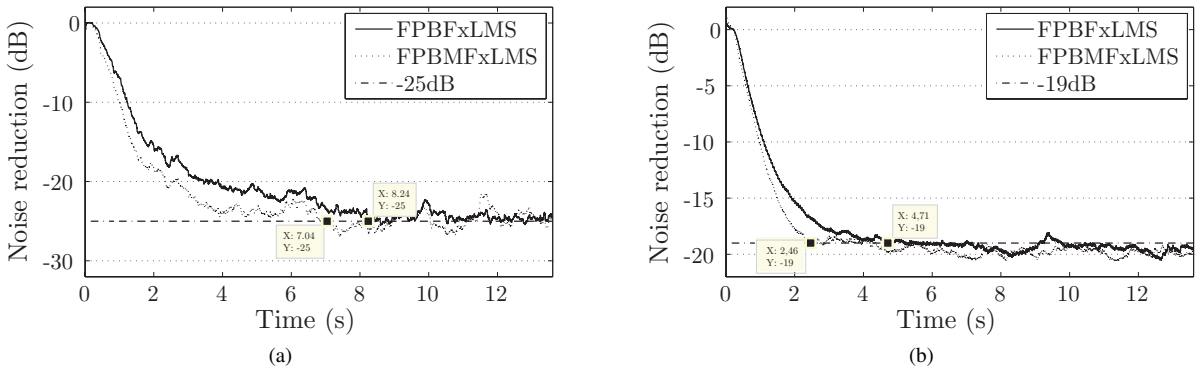


Fig. 8: Learning curves of the FPBMFxLMS and the FPBFxLMS algorithms for the 1:2:2 ANC system and different reference signals. The disturbance noise used in (a) is a band-limited white noise, and in (b) is a periodic tonal noise.

III. Different configurations of the ANC system (I:J:K) were considered. As an example, Fig. 6 shows a picture of the multichannel ANC system with the 1:2:2 configuration. Some other arrangements were also used to test the performance of other configurations, like 1:1:1 or 1:4:4. It should be highlighted that the prototype is capable of dealing with massive systems with high numbers of I, J and K, reaching more than 600 processed channels in the best case.

The performance of the ANC system was evaluated from different points of view. First, section V-A is devoted to validating the ANC performance. For this purpose, the attenuation of the reference signal achieved by the ANC system at the error sensors was measured. A convergence performance comparison between the FPBMFxLMS and the FPBFxLMS for different types of reference signals is shown in section V-B. The computational complexity of the GPU implementation of both algorithms is detailed in section V-C, and, finally, some computing results of the FPBMFxLMS algorithm are analyzed in section V-D.

With regard to the computing results, a previous analysis of the distribution of threads in a block and blocks in a grid is necessary for each specific case in order to achieve good performance [27]. This previous analysis consists of testing the processing delay of the algorithm by changing both the dimensions of the blocks of threads and the grid of blocks to find the fastest configuration for each different case.

A. Residual Noise level

The 1:2:2 configuration of the ANC system was considered in this set of experiments. The following parameters were chosen: $B = 2048$, $L = M = 4096$, and two different types of reference signal. The reference signals were:

1) band-limited white noise.
2) a periodic noise that emulates an engine signal composed of six harmonics between 100Hz and 200Hz with an effective fundamental frequency of 20 Hz.

Fig. 7 shows the power spectral density of the average signals measured at both error sensors by using the described algorithms. A similar performance is observed in both algorithms with noise reductions that depend on the type of reference signal. If we consider the band pass filtered white noise, an attenuation between 20-30 dB is achieved depending on the frequency (see Fig. 7(a)). Fig. 7(b) illustrates the results for periodic noise. It can be easily observed that a reduction of around 45 dB is achieved at each harmonic.

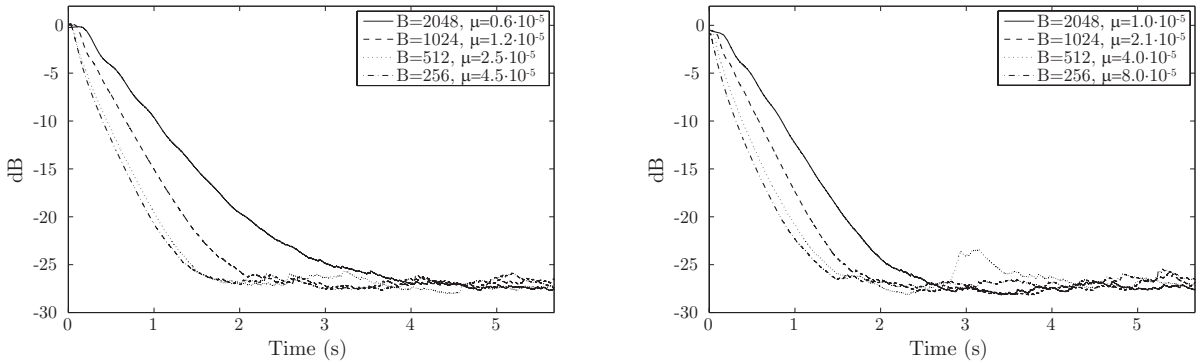


Fig. 9: Learning curves of the FPBMFxLMS and FPBFxLMS algorithms for different B size and a single tone reference.

B. Convergence performance

This section compares the convergence performance of the FPBMFxLMS and the FPBFxLMS algorithms. The same set of parameters used in the previous section was chosen. The learning curves of both algorithms were obtained using the following equation:

$$A[n] = 10\log_{10}\left(\frac{P_e[n]}{P_d[n]}\right), \tag{12}$$

with

$$\begin{aligned} P_d[n] &= \alpha P_d[n-1] + (1-\alpha)p_d[n], \\ P_e[n] &= \alpha P_e[n-1] + (1-\alpha)p_e[n]. \end{aligned} \tag{13}$$

and

$$\begin{aligned} p_d[n] &= \sum_{k=1}^{K} d_k^2[n], \\ p_e[n] &= \sum_{k=1}^{K} e_k^2[n]. \end{aligned} \tag{14}$$

where K is the number of error sensors.

Fig. 8 illustrates the performance of the algorithms. The highest step size μ that ensures the stability for each case was chosen. Using filtered white noise as the disturbance signal, the step-size parameter was set to $\mu_c = 1.5 \cdot 10^{-5}$ and $\mu_m = 2.8 \cdot 10^{-5}$ for the conventional filtered-x scheme and the modified filtered-x scheme, respectively. The step-size parameter was set to $\mu_c = 3.5 \cdot 10^{-7}$ and $\mu_m = 7.3 \cdot 10^{-7}$ for the periodic noise, respectively.

As Fig. 8 shows, the FPBMFxLMS provides faster convergence speed than the FPBFxLMS but similar steady-state behavior. Specifically, for the case of the filtered white noise, the modified scheme converges 1.2 seconds before the conventional scheme, which corresponds to a 15% reduction in the convergence time. A larger difference is observed when periodic noise is considered. The FPBMFxLMS converges almost 2.25 seconds before FPBMFxLMS, which corresponds to a 47% reduction in the convergence time. Therefore, it can be concluded that the FPBMFxLMS significantly outperforms the FPBFxLMS in terms of convergence speed. Also, note that both algorithms converge faster for the periodic noise signal than for the filtered random noise signal.

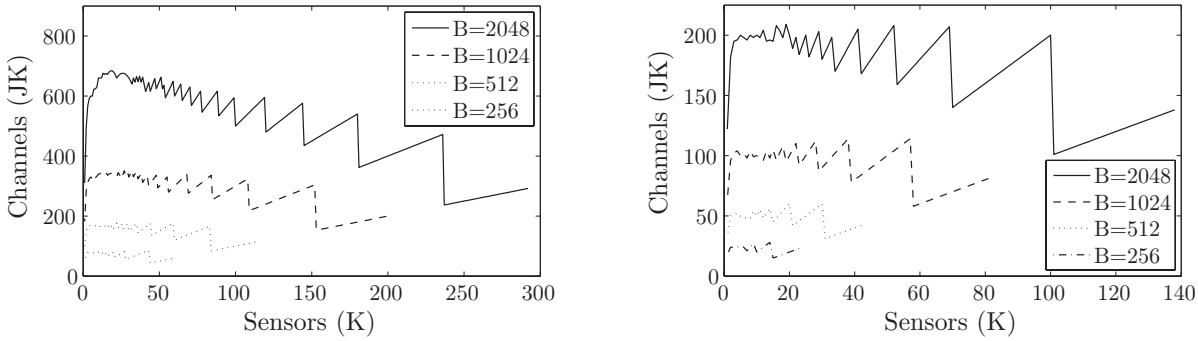
TABLE II: Processing time and total number of multiplications, additions, and FFTs per iteration of the GPU implementation of (1) the FPBFxLMS algorithm and (2) the FPBMFxLMS algorithm for $L = M$ and different ANC configurations.

		I:J:K configuration		
		1:1:1	1:2:2	1:4:4
(1)	Multiplications	$8L$	$24L$	$80L$
	Additions	$3L$	$10L$	$36L$
	FFTs	5	9	17
	Time (ms)	0.55	0.78	1.58
(2)	Multiplications	$12L$	$40L$	$144L$
	Additions	$9.5L$	$33L$	$124L$
	FFTs	5	9	17
	Time (ms)	0.68	0.96	2.05
	M_m/M	1.5	1.6	1.8
	A_m/A	3.16	3.3	3.45
	t_m/t	1.24	1.23	1.30

Another important property of the adaptive algorithms is the stability limit. In the literature, there are some contributions made to study the convergence behavior of the Block filtered-x LMS algorithm (BFxLMS). The maximum μ parameter that leads to the fastest convergence rate was derived in [39] and is

$$0 \leq \mu < \frac{1}{B\lambda_{max}} \tag{15}$$

where λ_{max} is the maximum eigenvalue of the filtered input signal autocorrelation matrix $\mathbf{R}_{vv}$ defined as $\mathbf{R}_{vv} = E[\mathbf{V}\mathbf{V}^T]$. Therefore, the convergence performance of the algorithm depends on the statistics of the input signal, the acoustic paths, and the block length B. For the same reference signal, the step-size parameter μ depends on B, so the maximum μ value increases by reducing the size of B, and, consequently, the convergence speed is improved by reducing B. However, the size of B is also limited by the real-time condition



(a) I=1 (b) I=4

Fig. 10: Maximum number of channels (JK) for each K value performing in real-time when (a) I=1 and (b) I=4 for different sizes of B and L=M=4096.

$t_{proc} < B/f_s$; therefore, there is a minimum value of B for each configuration that assures the real-time condition and maximum convergence speed.

Fig. 9 illustrates the convergence behavior of both algorithms using a single tone of 200 Hz as the reference signal and varying the size of B between 256 and 2048. As expected, it shows that the algorithms converge faster with a smaller block size, B. As these results show, the maximum μ is more or less doubled when B is halved. This fact can be explained from Eq. (15), where, for the same reference signal, the maximum μ is doubled by reducing the size of B by half. The difference in convergence time between $B = 2048$ and $B = 256$ is around 2.5 seconds in the conventional scheme and 1 second in the modified scheme (using $f_s = 44,100$ Hz). Finally, Fig. 9 shows that in order to achieve a certain convergence speed, the modified scheme can use a larger block size than the conventional scheme. This means the adaptive controller has more time for processing without violating the real-time condition, and, therefore, more channels can be handled while maintaining a given convergence speed.

C. Computational complexity

Table II compares the computing time and the computational complexity in terms of multiplications, additions, and FFTs per iteration of the GPU implementation of the two algorithms (FPBMFxLMS and FPBFxLMS) for different configurations. Since we use a value of $M = L$, the computational complexity only depends on L.

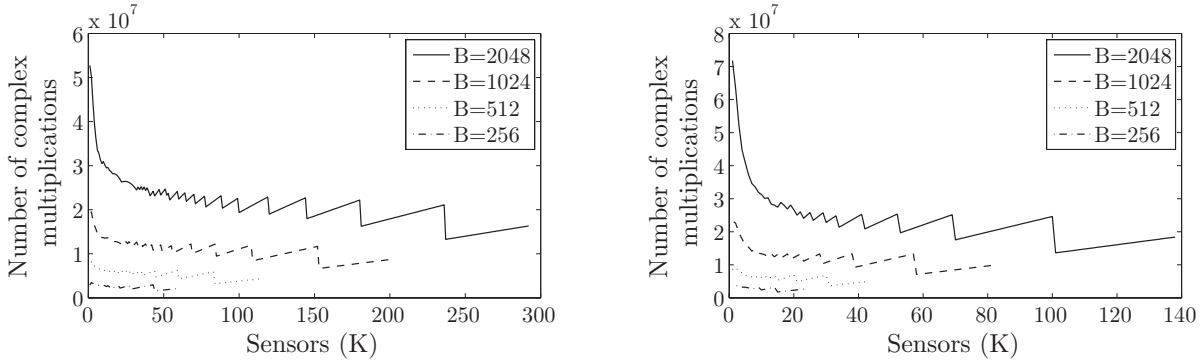
First, table II shows that the computational complexity for both algorithms increases significatively with the number of channels. In the modified scheme, when the ANC configuration changes from 1:1:1 to 1:4:4 (16 secondary pahts) the number of multiplications increases by a factor of 12, the additions by a factor of 13, and the FFTs by a factor of 3.45 while the time delay increases only by a factor of 3. Taking into account that the complexity increases with the number of channels, we can conclude that the computational complexity is a bottleneck of massive multichannel ANC systems. Therefore, if the operations of each channel are properly parallelized, an implementation over GPU could be a viable and meaningful solution.

Table II also shows that the modified scheme exhibits higher computational complexity due to the estimation of the error signals ($\hat{\mathbf{e}}k_n$) (see Fig. 3). Moreover, the number of multiplications and additions also significantly increases but not the number of FFTs. Furthermore, if we define the ratio M_m/M as the number of multiplications of the FPBMFxLMS algorithm divided by the number of multiplications of the FPBFxLMS, and the same for additions (A_m/A) and time delay (t_m/t), it is shown that the ratios of both multiplications and additions are larger than the ratio of delays. This result further confirms that the GPU implementation is a good solution for multichannel ANC systems. Specifically, the increase of computational complexity of the modified scheme can be overcome by using a GPU implementation.

D. Prototype computing performance

It is well known that the zone of high attenuation achieved by an ANC system can be extended by adding more sensors and loudspeakers. However, as noted in the previous section, the computational cost can become extremely large.

In this section, we will study the computational constraints of the multichannel ANC prototype using the FPBMFxLMS algorithm. For this purpose, Fig. 10 shows the maximum number of channels that the ANC system can handle without violating the real-time condition for L=M=4096, different B values, and in two cases (Fig. 10(a) one reference signal, and Fig. 10(b) four reference signals). This maximum number of channels is calculated by fixing the number of error sensors and finding the maximum number of actuators (J) that the system can handle without violating the real-time condition. When the maximum J value for each value of K is found, the maximum number of physical channels that are processed for each value of K is $J \cdot K$. For example, in Fig. 10(a), when $K = 180$, the maximum number of actuators that can be used without violating the real-time condition is $J = 3$; therefore, the maximum number of processed channels is 540. However, for $K = 181$, the maximum number of actuators is $J = 2$ because the real-time condition is violated with $J = 3$. Thus, the maximum number of processed channels is 362 ($J = 2$) when $K = 181$. For this reason, the shape of the curves jumps with the increase of error sensors.



(a) I=1 (b) I=4

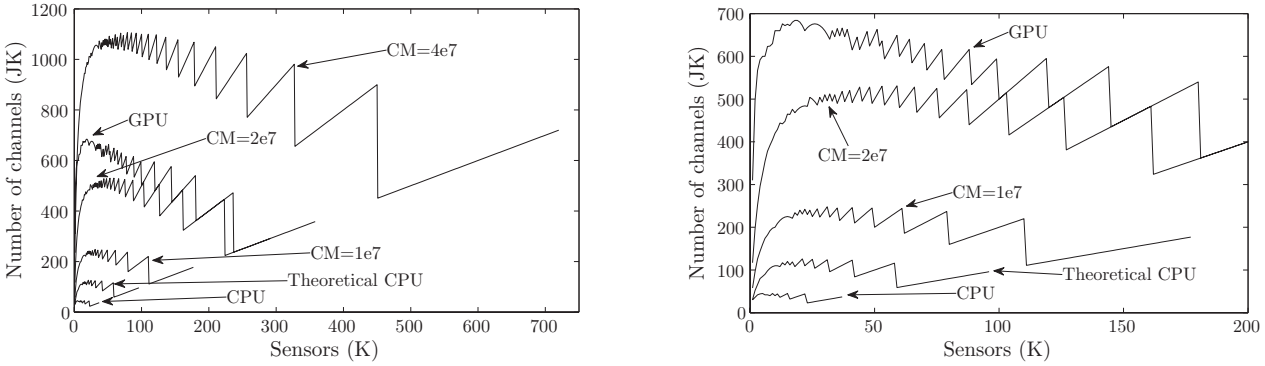
Fig. 11: Number of complex multiplications (CM) performed for the maximum JK configuration when (a) I=1 and (b) I=4 for different sizes of B and L=M=4096.

The following considerations are highlighted in the simulation results depicted in Fig. 10:

- The maximum number of actuators for each value of K is calculated by taking into account that it has to satisfy the real-time condition: $t_{proc} < B/f_s$. Therefore, if B increases, there is more time for processing and thus more channels can be handled.
- Two systems with different I:J:K configurations could have the same number of physical channels but different computational costs. For example, both the 1:1:2 and 1:2:1 configurations have 2 physical channels, but the second configuration has a higher computational cost because it has two adaptive filters instead of one. An increase in the number of adaptive filters involves many more operations than an increase in the number of error sensor signals, with IJ being the number of adaptive filters. Therefore, when K is low and J is high, the maximum number of channels is limited by the delay of processing the adaptive filters (see Fig. 10 when K is low).
- When K increases, J has to decrease in order to satisfy the real-time condition, and, consequently, the number of adaptive filters decreases and the curves of the number of processed channels grow quickly reaching the maximums. The maximum of the curves is reached when neither K nor J is much bigger than the other.
- On the other hand, when J is small and K is large, even though the number of adaptive filters is low, the system is limited by the error signal handling (see Fig. 3). Moreover, for low values of B or configurations with more than one reference signal, the decrease in processed channels with the increase of K is not so significant. This is because the system is able to process fewer sensors in real time than when $I = 1$ and $B = 2048$; therefore, since K is moderate, the decrease in the number of processed channels is also moderate.
- Two systems with the same JK configuration but a different number of reference signals have the same number of processed physical channels but different computational costs. For instance, if four reference signals instead of one are handled, the maximum number of processed channels are reduced because each channel is used four times instead of one (one time for each reference signal). On the other hand, due to the parallelization of the operations of each reference signal, even though the channels are used four times, the number of processed channels is not decreased by four.

Once the maximum number of processed channels has been analyzed for each value of K, the number of complex multiplications (CM) involved in both the products and the FFTs for the JK configurations derived in Fig. 10 is depicted in Fig. 11. It can be observed that the number of CM performed is not constant for the different configurations. This is because the GPU implementation is affected by the JK configuration. The most remarkable aspect of Fig. 11 is that the JK configurations with more CM are those with low values of K and high values of J. As explained above, the number of adaptive filters depends on both the I and J variables; therefore, if J grows, more CM are performed because there are more adaptive filters to cope with. Consequently, this is accentuated when $I = 4$.

The maximum number of channels that can be handled in real time by the GPU is shown in Fig. 10. In order to compare the computational capabilities of the GPU with other hardware platforms, an ANC system based on a CPU i7 was also implemented using one core and a sequential execution. The maximum number of channels allowed by the CPU implementation was theoretically and practically studied. Moreover, the GPU and CPU evaluation results were also compared with a theoretical processing machine that was limited to perform $1 \cdot 10^7$, $2 \cdot 10^7$, or $4 \cdot 10^7$ CM per buffering time. The results of this comparison are illustrated in Fig. 12 for the case when $I = 1$ and $B = 2048$. In this case, the buffering time is $B/f_s = 2,048/44,100 = 0.0464$ seconds. For example, the curve labeled as 'CM=$1e7$' in Fig. 12 represents the maximum number of channels for each value of K that a given machine would process if this machine is able to carry out $1 \cdot 10^7$ CM every 46.4 milliseconds. Finally, the theoretical maximum performance of our CPU was found by calculating the maximum number of CM that this



(a) I=1, B=2048

(b) Detail for the first 200 values of K

Fig. 12: A comparison of the maximum number of processed channels for the GPU implementation, the CPU implementation, and a theoretical processing machine limited to perform $1 \cdot 10^7$, $2 \cdot 10^7$, or $4 \cdot 10^7$ complex multiplications (CM) per block of samples for $B = 2048$ and $I = 1$.

CPU could handle in a real-time execution. A CM involves 4 floating point multiplications and 2 floating point additions. The floating point multiplications are performed by our CPU in 5 clock cycles, whereas the floating point additions are performed in 3 clock cycles (see annex 3 of [40]). Therefore, a CM involves 26 clock cycles. The CPU operates at 3.07GHz, which means that our CPU could make $3.07 \cdot 10^9/26$ CM per second and $(3.07 \cdot 10^9/26) \cdot 0.0464$ CM per buffering time. This is represented in the figure with the curve labeled as 'theoretical CPU'. Since our CPU also performs memory transactions and flow control instructions, the practical CPU implementation handles fewer channels than the 'theoretical CPU'. Furthermore, the theoretical processing machines that process $1 \cdot 10^7$, $2 \cdot 10^7$, or $4 \cdot 10^7$ CM per buffering time would also have to perform memory transactions and flow control instructions in addition to the complex multiplications. Therefore, in practice, these three curves would be lower.

Fig. 12 illustrates that the GPU implementation outperforms the CPU implementation and shows the number of CM that a processing machine would have to carry out each buffering time to outperform the GPU implementation. Moreover, the maximum benefit of the GPU is obtained when low values of K and high values of J are used. Therefore, the GPU reaches $4 \cdot 10^7$ CM per buffering time. This can be explained by the fact that an increase in the value of K involves an increase in the time required to handle the error signals in the frequency domain.

VI. Conclusions

This work has analyzed the suitability of GPUs for the real-time implementation of multichannel adaptive systems (specifically for ANC systems based on the FxLMS algorithm). We have compared two different schemes of the FxLMS algorithm, one that is based on the conventional filtered-x scheme (FPBFxLMS) and another that is based on the modified filtered-x scheme (FPBMFxLMS). To fit the hardware/GPU requirements, the algorithms have been implemented in the frequency domain, working with blocks of data and partitioning the adaptive filters. As a result, a prototype of a multichannel sound-control application has been successfully implemented on a GPU using CUDA language and exploiting the benefits of the parallelization of the multiple channels involved. In order to obtain the most efficient implementation, we have used the NVIDIA CUDA avoiding memory copies in the GPU and analyzing certain CUDA aspects such as the number of threads per block and the distribution of threads within the blocks.

This work also shows that the FPBMFxLMS algorithm converges faster than the FPBFxLMS algorithm, but that the computational complexity of the FPBFxLMS increases significantly, especially for massive multichannel systems. Nevertheless, by taking advantage of the parallelization capabilities of the GPU, this increase in computational cost did not lead to a great increase in the processing delay. Therefore, the use of a GPU platform can help to overcome the disadvantage of the modified scheme in a real-time ANC system. As a conclusion, the same convergence behavior of the conventional scheme can be obtained with the modified scheme by using a larger block size. This provides more time for processing and, therefore, also provide the possibility to extend the zone of high attenuation by adding more microphones and loudspeakers.

Finally, we also studied the computational limits of the ANC system in order to obtain a massive multichannel system that provides a large area of high attenuation by using more sensors. This work demonstrates that the GPU is a meaningful and versatile solution for massive multichannel ANC systems, which can provide, slightly more than 600 processed channels in real time. Moreover, it is important to note that more channels could be processed by using a different audio card with a lower frequency sampling, by decimating, or by using newer GPUs with more computational capacity.

References

[1] P. A. Nelson and S. J. Elliott, *Active control of sound*. Imperial College Press, New York, 1992.

[2] S. M. Kuo and D. R. Morgan, "Active noise control: a tutorial review," *Proceedings of the IEEE*, vol. 87, pp. 943–973, 1999.

[3] S. J. Elliott and P. A. Nelson, "Active noise control," *IEEE Signal Processing Magazine*, vol. 10, no. 4, pp. 12–35, October 1994.

[4] S. Haykin, *Adaptive filter theory*. Prentice Hall, 1986.

[5] S. J. Elliott, I. M. Stothers, and P. A. Nelson, "A multiple error LMS algorithm and its application to the active control of sound and vibration," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, no. 10, pp. 1423–1434, October 1987.

[6] B. Widrow and S. D. Stearns, *Adaptive signal processing*. Prentice-Hall, Englewood Cliffs, New York, 1985.

[7] S. Elliott, I. M. Stothers, P. A. Nelson, A. M. McDonald, and D. C. Quinn, "The active control of engine noise inside cars," in *Proceedings of the Inter-Noise conference*, New york, 1998, pp. 987–990.

[8] A. Gonzalez, M. Ferrer, M. D. Diego, G. Pinero, and J. J. Garcia-Bonito., "Sound quality of low-frequency and car engine noises after active noise control," *Journal of Sound and Vibration,*, vol. 265, pp. 663–679, 2003.

[9] S. Elliott, P. Nelson, I. Stothers, and C. Boucher, "In flight experiments on the active control of propeller-induced cabin noise," *Journal of Sound and Vibration,*, vol. 140, pp. 219–238, 1990.

[10] S. M. Kuo and D. R. Morgan, *Active noise control, algorithms and DSP implementations*. John Wiley & Sons, Inc., New York, 1996.

[11] "NVIDIA programming guide," *online at: http://developer.download.nvidia.com/*.

[12] M. D. McCool, "Signal processing and general-purpose computing and GPUs," *IEEE Signal Processing Magazine*, vol. 24, pp. 109–114, 2007.

[13] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proceedings of the IEEE*, vol. 96, pp. 879–899, May 2008.

[14] L. Savioja, V. Valimki, and J. O. Smith, "Audio signal processing using graphics processing units," *Journal Audio Eng. Soc*, vol. 59, pp. 3–19, 2011.

[15] N. Tsingos and W. Jiang, "Using programmable graphics hardware for acoustics and audio rendering," *Journal Audio Eng. Soc*, vol. 59, pp. 628–648, 2011.

[16] C. J. Webb and S. Bilbo, "Computing room acoustics with CUDA-3D FDTD schemes with boundary losses and viscosity," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, 2011, pp. 317–320.

[17] J. Vaněk, J. Trmal, J. V. Psutka, and J. Psutka, "Optimized acoustic likelihoods computation for NVIDIA and ATI/AMD graphics processors," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1818–1828, 2012.

[18] P. Cardinal, P. Dumcouhel, and G. Boulianne, "Large vocabulary speech recognition on parallel architectures," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 21, no. 11, pp. 2290–2300, 2013.

[19] R. Mazur, J. O. Jungmann, and A. Mertins, "On CUDA implementation of a multichannel room impulse response reshaping algorithm based on p-norm optimization," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, vol. 24, 2011, pp. 305–308.

[20] J. Lorente, G. Piñero, A. M. Vidal, J. A. Belloch, and A. Gonzalez, "Parallel implementations of beamforming design and filtering for microphone array applications," in *Proceedings of the 19th European Signal Processing Conference*, Barcelona, 2011, pp. 501–505.

[21] S. Zhao, S. Ahmed, Y. Liang, K. Rupnow, D. Chen, and D. L. Jones, "A real-time 3D sound localization system with miniature microphone array for virtual reality," in *Proceedings of the IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Singapore, July 2012, pp. 1853 – 1857.

[22] D. Theodoropoulos, G. Kuzmanov, and G. Gaydadjiev, "Multi-core platforms for beamforming and wave field synthesis," *IEEE Transactions on Multimedia*, vol. 99, pp. 235–245, December 2010.

[23] J. A. Belloch, A. Gonzalez, F. J. Martinez-Zaldivar, and A. M. Vidal, "Real-time massive convolution for audio applications on GPU," *Journal of Supercomputing*, vol. 58, pp. 449–457, 2011.

[24] J. A. Belloch, M. Ferrer, A. Gonzalez, F. J. Martinez-Zaldivar, and A. M. Vidal, "Headphone-based spatial sound with a GPU accelerator," in *Proceedings of the International Conference on Computational Science*, Omaha, 2011, pp. 116–125.

[25] M. Schneider, F. Schuh, and W. Kellerman, "The generalized frequency-domain adaptive filtering algorithm implemented on a GPU for large-scale multichannel acoustic echo canceller," in *Proceedings of the TG Conference on Speech Communication*, 2012, pp. 1–4.

[26] J. Lorente, A. Gonzalez, M. Ferrer, J. A. Belloch, M. de Diego, G. Piñero, and A. Vidal, "Active noise control using graphics processing units," in *Proceedings of the 19th International Congress on Sound and Vibration*, 2012, pp. 138–146.

[27] J. Lorente, J. A. Belloch, M. Ferrer, A. Gonzalez, J. A. Belloch, M. de Diego, G. Piñero, and A. Vidal, "Multichannel active noise control system using a GPU accelerator," in *Proceedings of the Inter-Noise conference*, 2012, pp. 7768–7780.

[28] B. Farhang-Boroujeny, *Adaptive Filters Theory and Applications*. John Wiley & Sons, Inc., New York, 1998.

[29] P. Sommen, "On the convergence properties of a partitioned block frequency domain adaptive filter," in *Proceedings of the European Signal Processing Conference*, Barcelona, 1990, pp. 201–204.

[30] X. Qiu and C. H. Hansen, "Multidelay adaptive filters for active noise control," in *Proceedings of the 14th International Congress on Sound and Vibration (ICSV)*, Cairns, Australia, 2007, pp. 724–732.

[31] J. Lorente, M. Ferrer, M. de Diego, J. A. Belloch, and A. Gonzalez, "GPU implementation of a frequency-domain modified filtered-x LMS algorithm for multichannel local active noise control," in *Proceedings of the 52nd Audio Engineering Society International Conference*, Sep 2013.

[32] Q. Shen and A. S. Spanias, "Time and frequency domain x block LMS algorithms for single channel active noise control," in *Proceedings of the 2nd International Congress of Recent Developments in Air-and Structure-Borne Sound Vibration*, 1992, pp. 353–360.

[33] S. C. Douglas, "Fast implementations of the filtered-x LMS and LMS algorithms for multichannel active noise control," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 4, pp. 454–465, July 1999.

[34] J. Páez Borrallo and M. Garcia Otero, "On the implementation of a partitioned block frequency domain adaptive filter (PBFDAF) for long acoustic echo cancellation," *Signal Processing*, vol. 27, no. 3, pp. 301–315, 1992.

[35] P. Sommen, "Partitioned frequency domain adaptive filters," in *Proceedings of the 23rd Asilomar Conference on Signals, Systems and Computers*, vol. 2, 1989, pp. 677–681.

[36] E. Bjarnason, "Active noise cancellation using a modified form of the filtered-x LMS algorithm," in *Proceedings of the 6th European Signal Processing Conference*, vol. 2, 1992, pp. 1053–1056.

[37] M. Pawełczyk, "Feedforward algorithms with simplified plant model for active noise control," *Journal of sound and vibration*, vol. 225, pp. 77–95, 2003.

[38] R. Burdisso, J. Vipperman, and C. Fuller, "Causality analysis of feedforward-controlled systems with broadband inputs," *The Journal of the Acoustical Society of America*, vol. 94, no. 1, pp. 234–242, 1993.

[39] D. P. Das, G. Panda, and S. M. Kuo, "New block filtered-x lms algorithms for active noise control systems," *IET Signal Processing*, vol. 1, no. 2, pp. 73–81, 2007.

[40] "Intel 64 and ia-32 architectures optimization reference manual," *online at: http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf*.

Jorge Lorente Jorge Lorente was born in Algemesí, Spain in 1985. He received a degree in Technical Engineering in Telecommunications in 2007 and a Master of Science degree in Telecommunication Technologies in 2010 from the Universitat Politècnica de València, Spain. He is a PhD grant holder from the Universitat Politècnica de València under the FPU program and is pursuing his PhD degree in Electrical Engineering at the Institute of Telecommunications and Multimedia Applications (iTEAM). His research focuses on audio signal processing and the parallelization of audio signal processing algorithms using a GPU accelerator. He is currently focused on the implementation of real-time multichannel sound application based on adaptive filtering.

Miguel Ferrer Miguel Ferrer Contreras graduated in Telecommunications Engineering in the year 2000 at the Universidad Politècnica de Valencia (UPV), Spain. He has been collaborating with the Audio and Communications Signal Processing Group (GTAC) since 1999, having completed a six-months research stay in the Instituto de Investigaciń Aplicada al Automv́il, Tarragona, Spain (The Automobile Applied Research Institute). Subsequently, he was awarded several grants from both the Communications Department of the UPV and the Research, Development, and Innovation Vice-Chancellery of the same university, enabling him to start his Doctoral studies and to collaborate in different research projects within the GTAC. During this period, he has authored or co-authored over twenty papers related to signal processing in renowned journals and conferences. Since 2005 he has been working as an assistant lecturer in the Communications Department of the Polytechnic University of Valencia (UPV). His research activity is focused on the study of adaptive algorithms and their application to audio digital processing and active noise control, the subject of his doctoral thesis.

Maria de Diego Maria de Diego was born in Valencia, Spain, in 1970. She received the Telecommunication Engineering degree from the Universidad Politecnica de Valencia (UPV) in 1994, and the Ph.D degree from the same University in 2003. Her dissertation was on active noise conformation of enclosed acoustic fields. She is currently working as Associate Professor in digital signal processing and communications. Dr. de Diego has been involved in different research projects including active noise control, fast adaptive filtering algorithms, sound quality evaluation, and 3-D sound reproduction, in the Institute of Telecommunications and Multimedia Applications (iTEAM) of Valencia. She has published more than 40 papers in journals and conferences about signal processing and applied acoustics. Her current research interests include wireless sensor acoustic networks, multichannel signal processing, distributed adaptive filtering and sound quality improvement.

Alberto González Alberto González Salvador works as Professor at the Universitat Politècnica de València, Spain. He attended the Universitat Politècnica de Catalunya, Spain, where he graduated in 1991 in Telecommunications Engineering with the highest honours. In 1997 he was awarded a Doctorate (PhD), magna cum laude, from the Universitat Politècnica de València. In 1995 he worked as a visiting researcher at the Institute of Sound and Vibration Research (ISVR) at the University of Southampton, United Kingdom. Currently, he is the head of the research group in Audio and Multimedia Digital Signal Processing. Alberto González has published over 100 papers in international technical journals and renowned conferences in the fields of signal processing and applied acoustics. In addition to this, he has led twelve research projects and has collaborated in twenty more. He is a member of the Senate of the Universitat Politècnica de València and has served as Dean of the School of Telecommunications Engineering since June 2012, having previously been head of the Communications Department and assistant director of research since 1997. His current research interests include multichannel signal processing for communications and three-dimensional sound reproduction.