DYNAMIC RANGE REDUCTION OF AUDIO SIGNALS USING MULTIPLE ALLPASS FILTERS ON A GPU ACCELERATOR

Jose A. Belloch[⋆†] *Julian Parker*[⋆] *Lauri Savioja*[†] *Alberto Gonzalez*[⋆†] *Vesa Välimäki*[⋆]

[⋆†] iTeAm, Universitat Politècnica de València, Valencia, Spain
[⋆] Dept. of Signal Processing and Acoustics, Aalto University, Espoo, Finland
[†] Dept. of Media Technology, Aalto University, Espoo, Finland

ABSTRACT

Maximising loudness of audio signals by restricting their dynamic range has become an important issue in audio signal processing. Previous works indicate that an allpass filter chain can reduce the peak amplitude of an audio signal, without introducing the distortion associated with traditional non-linear techniques. Because of large search space and the consequential demand of the computational needs, the previous work selected randomly the delay-line lengths and fixed the filter coefficient values. In this work, we run on a GPU accelerator multiple allpass filter chains in parallel that cover all relevant delay-line lengths and perform a wide search on possible coefficient values in order to get closer to the optimal choice. Our most exhaustive method, which tests about 29 million parameter combinations, reduced the amplitude of test signals by 23% to 31%, whereas the previous work could only achieve a reduction of 23% at best.

Index Terms— Audio systems, digital filters, parallel architectures, parallel processing

1. INTRODUCTION

Dynamic range reduction in audio signals is important. This process consists of restricting the dynamic range of an audio signal to a smaller space [1], hence allowing maximization of loudness. Useful dynamic range reduction is achieved if the peak amplitude of a signal decreases with respect to its RMS amplitude.

Up to now, dynamic range reduction was achieved by using non-linear techniques [2,3]. Allpass filters were presented previously in [4] as a method for reducing the peak to RMS amplitude ratio, since they present a flat frequency response, but a non-linear phase response. By modifying the phase of the signal, maximum peak level can be reduced without introducing new frequency content. Listening tests suggest that if the impulse response length of the allpass filters is below 4 ms, the change is inaudible [5].

The previous work [4] analyzes in detail the behavior of a first-order allpass filter. This analysis concludes that peak amplitude of the impulse response is minimised when the allpass coefficient is equal to $\pm\Phi$, where Φ is the inverse of the *golden ratio*. To three decimal places, the value of the *golden ratio* is 1.618 and its inverse Φ is 0.618. This property was also referenced in [6], without proof.

In addition, previous work [4] suggests that the unit delay in the first-order allpass is replaced with a longer delay-line length whose maximum value d_{max} is set to 30 using a sample frequency of 44.1 kHz. This maximum is chosen in order to restrict the length of the impulse response to the range in which the spreading of the signal is inaudible. Similar allpass filters with a long delay line have previously been used for artificial reverberation [7, 8] and for spectral delay in audio processing [9].

The authors in [4] propose the structure shown in Fig. 1. This structure is composed of M filters in parallel, where each filter is made of a cascade of three allpass filters with three different embedded delay-line lengths $\{d_1, d_2, d_3\}$, and three coefficients $\{a, b, c\}$ whose values are set to $a = -\Phi$, $b = +\Phi$, and $c = -\Phi$. Their test consists of processing an input signal through 100 filters in parallel (i.e. in their tests, $M = 100$). Each one of the filters has its delay-line lengths $\{d_1, d_2, d_3\}$ determined randomly, between 1 and the d_{max} of 30. The output of every filter is examined, and the one that produces the lowest peak amplitude is selected as the one that offers best linear dynamic range reduction. Note that the structure also contains a path that bypasses the processing, since in rare cases the lowest peak amplitude produced by the allpass filters could be higher than the input. In that case, the input itself is selected as the output.

Their results show that even with such a small random selection of delay line lengths, the dynamic range is generally reduced [4]. However, the theoretical maximum dynamic range reduction is not achieved. Targeting more delay-line length combinations in order to maximise the reduction requires the use of more computational resources. This resource problem can be vastly reduced by noticing that the process is

This work was conducted in fall 2013 when J. A. Belloch was visiting the Aalto University Department of Signal Processing and Acoustics. Thanks to the EEBB-I-13-06059, TEC2012-38142-C04-01, GVA/2013/134 and to the PROMETEO/2009/013 projects for funding.

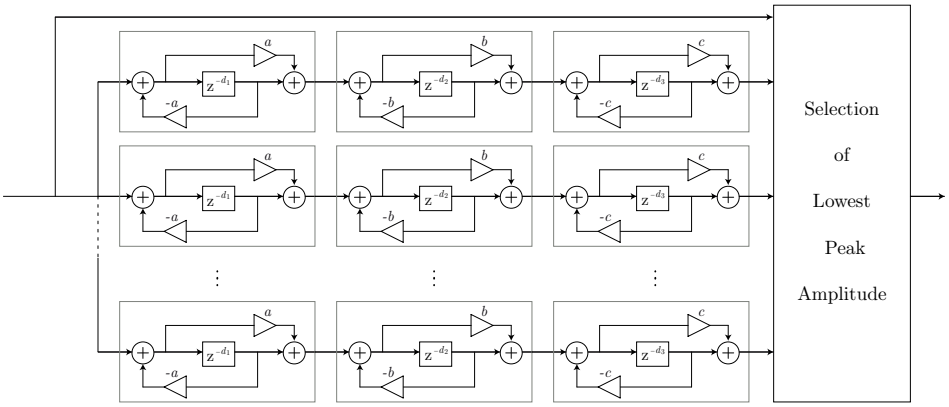

Fig. 1. Block diagram of the M parallel allpass filter chains.

intrinsically highly parallel, and hence suitable for computation with a Graphics Processing Unit (GPU). A GPU can be considered to be a Single Instruction Multiple Data machine (SIMD), i.e., a computer in which a single set of instructions is executed on a large number of data sets simultaneously. The analogy with the proposed parallel allpass structure should be clear.

This paper presents a parallel GPU-based implementation of the structure that aims to seek the maximum dynamic range reduction of a signal. To this end, not only are all combinations of delay-line lengths examined, but also different values of coefficients, in order to validate or reject the use of the *golden ratio* as a unique coefficient. We also assess the computational performance achieved by the GPU when these are compared with the ones obtained by a powerful multi-core computer. The next section gives an overview of some characteristics of GPUs and its program environment, and its use in audio signal processing. In Section 3, the test setup is presented, along with its GPU-based implementation. Section 4 is devoted to analyzing the results, which are also compared with the ones shown in previous work [4]. Finally, some concluding remarks are given in Section 5.

2. GPUS IN AUDIO SIGNAL PROCESSING

Compute Unified Device Architecture (CUDA) is a software programming model that allows the use of GPUs for applications beyond graphics rendering. GPUs have the potential of highly parallel data processing. The recent Nvidia GPU Kepler architecture [10] is composed of multiple Stream Multiprocessor (SMX), where each SMX consists of 192 pipelined cores per SMX. A GPU device has a large amount of off-chip device memory (*global-memory*) and a limited amount of fast on-chip memory (*shared-memory*). The code to be executed on the GPU by multiple elementary processes, called *threads*, is written as a *kernel* function. A CUDA grid configuration, which defines the number of threads and how they are distributed and grouped, must be built into the main code. We can define a CUDA grid to be a mesh of blocks, each of them has a mesh of threads. At runtime, the blocks are distributed among SMXs by the scheduler. If the number of blocks exceeds the resources of the GPU, these blocks wait until other blocks finish their computation in order to be later hosted. Thus, the number of launched threads can exceed the number of physical cores. It is important to point out that each thread can have up to 255 registers, and that only the threads that belong to the same block can share data through *shared-memory*.

GPU computing has already been applied to different problems in acoustics and audio processing [11, 12]. A full 3-D model of drums in a large room was described in [13]. GPU-based multichannel local active noise control implementations can be found in [14]. There are also multichannel spatial audio applications in the literature that use the GPU as a co-processor for carrying out all the audio signal processing, such as applications based on wave field synthesis (WFS) [15], and headphone-based binaural audio [16].

3. TEST SETUP

Taking into account that the maximum delay-line length d_{max} of the allpass filter is chosen to be 30, it would be easy to think that we are dealing with 27,000 possibilities. However, as an allpass filter cascade is a linear system and can be re-ordered freely, many of the possibilities are redundant. All delay-line lengths combinations are obtained if we apply *Multiset theory* [17]. This theory indicates that given n elements, the number of multisets of cardinality k is:

$$\frac{(n+k-1)!}{k!(n-1)!}. \tag{1}$$

In our case, n=30 and k=3. Thus, combining all possible delay-line lengths implies 4,960 combinations of interest. Additionally, there are also the three different coefficients

$\{a, b, c\}$ that previously were fixed to Φ. However, as was mentioned in Sec.1, it was not proved that Φ offers maximum reduction for general signals, only for impulses. Therefore, it may be advantageous to explore a larger coefficient space. If we vary the coefficients between 0.3 and 0.7 in steps of 0.05, we have nine possibilities for each coefficient. As we have three coefficients, we have 729 possibilities more plus one. We must also add the combination of the previous work (all coefficients are equal to Φ). It is clear that a single allpass filter can have a coefficient of $\pm a$ and maintain a stable impulse response. Therefore, we must also consider the different combinations of the sign in the coefficients: $\pm a$ and $\mp a$, $\pm b$ and $\mp b$, and $\pm c$ and $\mp c$. This implies 8 sign-based combinations for each $\{a, b, c\}$ coefficient combination. In total, to tackle all the described combinations, we must compute 28,966,400 allpass filter chains. The use of the GPUs can reduce the required processing time to evaluate all these combinations.

3.1. GPU-based Implementation

The hardware we use is a Nvidia Tesla K20c that is based on the Kepler architecture and is composed of 13 SMXs. The implementation we propose aims to launch as many threads as combinations we have. To this end, we can divide the combinations in two: coefficients combinations and delay-line lengths combinations. These combinations are stored in two matrices at the GPU *global-memory*. The CUDA grid we launch is two-dimensional and is composed of blocks of 256 threads. In this case, the identification of a thread is given by two variables `Col` and `Row`.

Figure 2 shows how thread (`Col`, `Row`) performs the allpass filter chain with coefficient combination `Row` and delay-line lengths combination `Col`. Each thread has three vectors of size d_{max} whose role is to simulate the delay lines. These $3d_{max}$ elements per thread are stored at the GPU registers. All the samples of the input are processed by the thread, which only stores in the *shared-memory* the maximum absolute value of the signal. Afterwards, a synchronization barrier is set in order to wait for all the threads to finish. After that, the reduction algorithm described by Harris [10] is implemented. It consists in looking at the minimum of all stored values and identifying its combination `Col` and `Row`.

4. RESULTS

The described implementation was applied to five isolated clean musical sounds, in the same way as in the previous work [4]. The sounds consist of single drum hits recorded from a Roland TR-808, a single piano tone played at C3 and a single synthesised mallet-like sound. All input signals have a sample rate of 44.1 kHz, and are normalized so that their peak amplitude is 1.

Previous work evaluated only 100 random possibilities. The first test we have carried out consisted in evaluating all

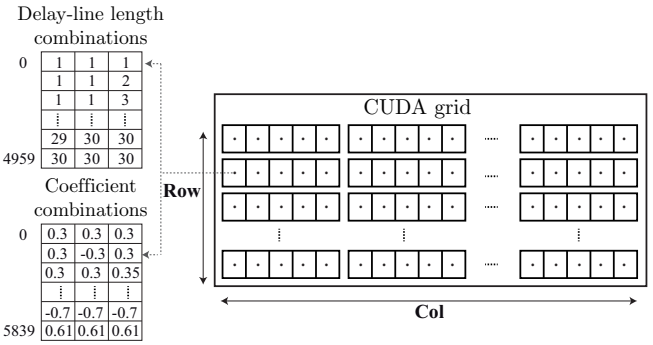

Fig. 2. Two-dimensional CUDA grid configuration. One thread performs an allpass filter using a delay line combination with a coefficient combination. **Col** defines the delay-line lengths and the **Row** determines the lookup in the coefficient table.

Sound	d_1	d_2	d_3	a	b	c	**Peak(out)**
Bass PW	24	22	28	-Φ	Φ	-Φ	0.94
Bass 1st	29	29	29	-Φ	Φ	-Φ	0.89
Bass 2nd	13	28	30	-Φ	-Φ	-Φ	0.76
Bass 3rd	19	23	27	-0.4	-0.65	-0.7	**0.74**
Snare PW	21	14	26	-Φ	Φ	-Φ	0.77
Snare 1st	23	28	29	-Φ	Φ	-Φ	0.72
Snare 2nd	17	20	23	-Φ	-Φ	-Φ	0.71
Snare 3rd	20	21	30	-0.70	-0.65	0.60	**0.69**
Hi-hat PW	1	19	11	-Φ	Φ	-Φ	0.85
Hi-hat 1st	2	20	20	-Φ	Φ	-Φ	0.82
Hi-hat 2nd	1	8	18	Φ	-Φ	Φ	0.80
Hi-hat 3rd	11	24	26	0.55	-0.40	0.55	**0.75**
Piano PW	20	28	5	-Φ	Φ	-Φ	0.86
Piano 1st	16	16	26	-Φ	Φ	-Φ	0.85
Piano 2nd	14	28	30	Φ	-Φ	Φ	0.80
Piano 3rd	20	25	30	0.40	-0.70	0.55	**0.77**
Mallet PW	11	14	29	-Φ	Φ	-Φ	0.87
Mallet 1st	11	16	28	-Φ	Φ	-Φ	0.79
Mallet 2nd	3	30	30	-Φ	-Φ	-Φ	0.76
Mallet 3rd	5	30	30	-0.45	-0.70	-0.70	**0.73**

Table 1. Delay lines lengths, coefficient values and maximum peak value that offer maximum dynamic range reduction obtained after the three tests for the five sounds. Maximum reduction is bolded. PW, 1st, 2nd, and 3rd correspond to results from previous work, first test, second test, and third test, respectively.

delay lines length combinations using the same coefficients that were used previously ($a = -\Phi$, $b = +\Phi$, $c = -\Phi$), which means 4960 combinations. The second test introduces three different variations in the sign of the coefficients but maintains the same value, i.e 14880 combinations. Finally, third test launches the 28,966,400 combinations. Table 1

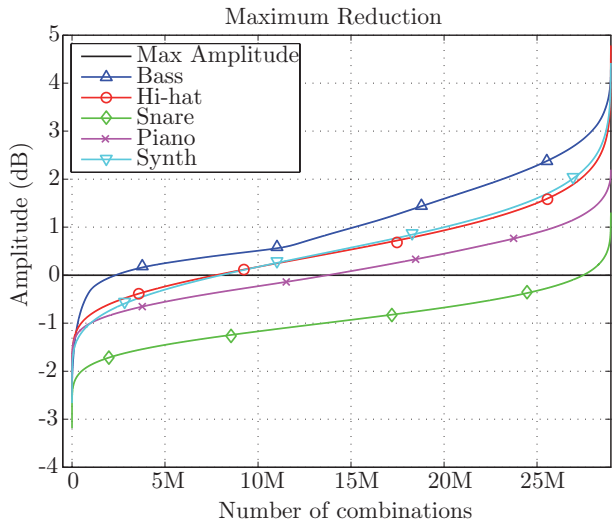

Fig. 3. Maximum peak value obtained for the 28,966,400 combinations for all the signals.

shows the results of the three tests and includes: the maximum peak value of all the signals with its corresponding delay-line lengths, and $\{a, b, c\}$ coefficients for each test. It is noticeable how the maximum reduction in every signal is improving as more combinations are attempted. One important result to point out is that maximum reduction is not necessarily achieved by using Φ as a coefficient.

Figure 3 presents in an increasing way the maximum peak value obtained for the 28,966,400 combinations for all the signals. On the left are the best results providing compressions whereas most of the random combinations would increase the dynamic range. Although it depends on the signal, few combinations achieve to reduce the dynamic range meaningfully. Thus, in case of using allpass filters for reducing dynamic range, many combinations must be tackled. Figure 4 compares the waveforms of the input signal, the signal obtained after processing from previous work, and the signal obtained after processing the third test: 28,966,400 combinations. It is noticed that the improvement is remarkable.

4.1. Computational Performance

A real-time scenario could be given in studio situation where every signal frame would have to be processed in a time of around 0.5 s, which implies the need to process 22050 samples (sample frequency f_s=44100 Hz) in less than 0.5 s. In order to assess the computational performance achieved by the GPU implementation, we have also performed the three tests in a powerful multi-core computer that has one SMPs (*Symmetric Multi-Processing*) Intel Xeon CPU X5680 at 3.33 GHz, which is a hexacore. Thus, our multicore computer is composed of six cores. We have tested all the combinations in a sequential way (one core carries out all combinations), and

Test	1st	2nd	3rd
Combinations	4960	14800	28,966,400
One core - CPU Time	1.35 s	4.08 s	7914 s
Six cores - CPU Time	0.23 s	0.71 s	1374 s
GPU Time	**0.07** s	**0.32** s	**760** s

Table 2. Processing time employed by the CPU and GPU to process the three described tests.

in a parallel way (distributing among the six cores all combinations) by using the programming framework *openMP* [18]. Table 2 shows the required time in processing 22050 samples for the three tests. As can be appreciated, only the third test can not be performed in real time by the GPU implementation. The GPU-based implementation outperforms the CPU-based multicore implementation in 2-4 times, which could be considered a limited difference. This could occur because the GPU-based implementation is penalized for the massive use of the registers, as commented in Section 3.1.

5. CONCLUSION

The results show that the use of the inverse of the *golden ratio* as a coefficient in the allpass filter chains does not necessarily give the absolute maximum reduction. Moreover, we have verified that the more combinations are tackled, the better reduction can be achieved, but a meaningful reduction is achieved by few combinations. To compute a large number of combinations, we have used the computational capacity of the GPUs, which has allowed us to launch 28,966,400 combinations in 760 s. This time is two times faster than performing the same number of combinations in a six-core powerful computer.

REFERENCES

[1] U. Zölzer, "DAFX - digital audio effects (second edition)," Chichester, U.K.: Wiley, Edited by Udo Zölzer, 2011.

[2] E. Vickers, "The loudness war: Do louder, hypercompressed recordings sell better?," *J. Audio Eng. Soc*, vol. 59, no. 5, pp. 346–351, 2011.

[3] G. Giannoulis, M. Massberg, and J.D. Reiss, "Digital dynamic range compressor design: A tutorial and analysis," *J. Audio Eng. Soc*, vol. 60, no. 6, pp. 399–408, 2012.

[4] J. Parker and V. Välimäki, "Linear dynamic range reduction of musical audio using an allpass filter chain," *IEEE Signal Processing Letters*, vol. 20, no. 7, pp. 669–672, 2013.

[5] J.M. Kates and K.H. Arehart, "Multichannel dynamic-range compression using digital frequency warping,"

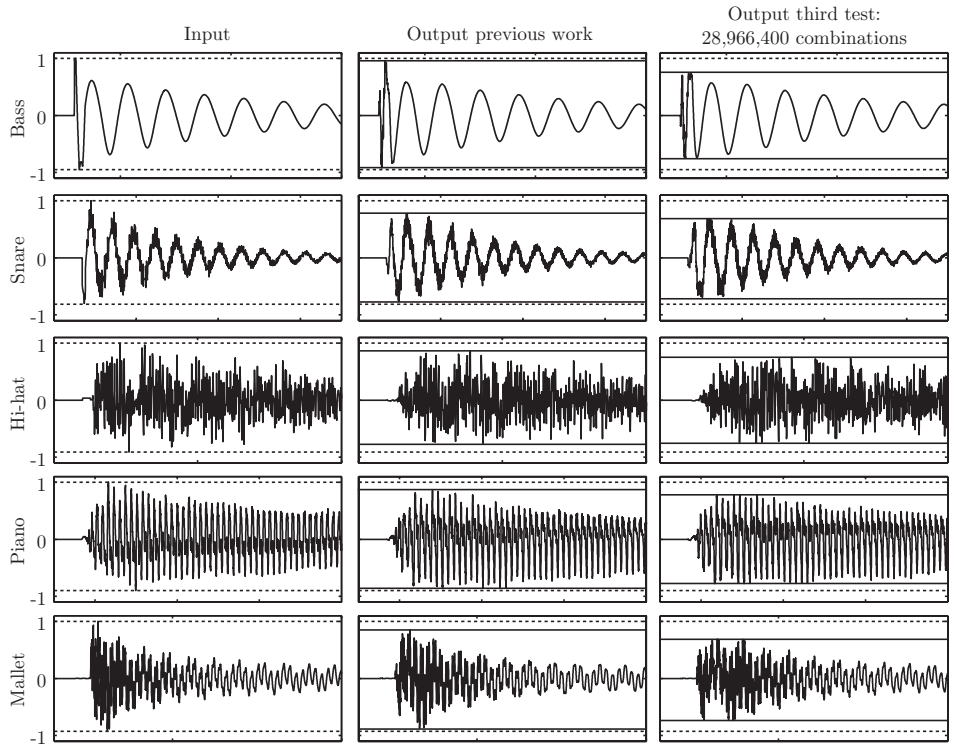

Fig. 4. Waveforms of the five isolated musical sound, before and after being processed (previous work [4] and third test: 28,966,400 combinations). The horizontal dashed lines show the positive and negative peaks of the original waveform whilst the solid horizontal lines show the positive and negative peaks after processing.

EURASIP J. on Applied Signal Process., vol. 18, pp. 3003–3014, 2005.

[6] D. Griesinger, "Impulse response measurements using all-pass deconvolution," in *Proceedings of the 11th AES Conference*, Portland, May 1992.

[7] M.R. Schroeder and B.F. Logan, "Colorless artificial reverberation," *J. Audio Eng. Soc*, vol. 9, no. 3, pp. 192–197, 1961.

[8] V. Välimäki, J.D. Parker, L. Savioja, J.O. Smith, and J.S. Abel, "Fifty years of artificial reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[9] V. Välimäki, J.S. Abel, and J.O. Smith, "Spectral delay filters," *J. Audio Eng. Soc*, vol. 57, no. 7-8, pp. 521–531, 2009.

[10] "Nvidia Programming Guide," *http://developer.download.nvidia.com/*.

[11] L. Savioja, V. Välimäki, and J. O. Smith, "Audio signal processing using Graphics Processing Units," *J. Audio Eng. Soc*, vol. 59, no. 1-2, pp. 3–19, 2011.

[12] N. Tsingos, W. Jiang, and I. Williams, "Using programmable graphics hardware for acoustics and audio rendering," *J. Audio Eng. Soc*, vol. 59, no. 9, pp. 628–646, 2011.

[13] S. Bilbao and C. J. Webb, "Physical modeling of timpani drums in 3D on GPGPUs," *J. Audio Eng. Soc*, vol. 61, no. 10, pp. 737–748, 2013.

[14] M. Schneider, F. Schuh, and W. Kellermann, "The generalized frequency-domain adaptive filtering algorithm implemented on a GPU for large-scale multichannel acoustic echo cancellation," in *Proc. of Speech Communication; 10. ITG Symposium*, Braunschweig, Germany, September 2012.

[15] D. Theodoropoulos, G. Kuzmanov, and G. Gaydadjiev, "Multi-core platforms for beamforming and wave field synthesis," *IEEE Transactions on Multimedia*, vol. 3, no. 2, pp. 235–245, April 2011.

[16] J. A. Belloch, M. Ferrer, A. Gonzalez, F.J. Martinez-Zaldivar, and A. M. Vidal, "Headphone-based virtual spatialization of sound with a GPU accelerator," *J. Audio Eng. Soc*, vol. 61, no. 7/8, pp. 546–561, 2013.

[17] W.D. Blizard, "Multiset theory," *J. Formal Logic Notre Dame*, vol. 30, no. 1, pp. 36–66, 1989.

[18] "OpenMP.," *http://www.openmp.org*.