SPECTRAL DATA MANAGEMENT TOOLS FOR ADDITIVE SYNTHESIS

Graziano Bertini, Massimo Magrini, Leonello Tarabella,

I.E.I. (Istituto per l'Elaborazione dell'Informazione)&CNUCE – C.N.R.
bertini@iei.pi.cnr.it

ABSTRACT

This paper describes a set of procedures, named Toolbox, for spectral data reduction to be used in additive synthesis system based on sinusoidal model. The Toolbox has been developed in MATLAB with the specific aim to support the realization of a very large scale additive synthesis system (AddSynth) based on special purpose VLSI chips. Test signals and musical tones played by real acoustic instruments, in different conditions of execution: intensity, pitch etc, are considered. The Toolbox mainly performs spectral envelope data extraction, data reduction, spectral interpolation and produces proper data block set for both the AddSynth machine and the related software simulator.

1. INTRODUCTION

In the framework of the CNR National Project MADESS II (Subproject "VLSI Systems", Research Line "Multimedia Systems"), a design of dedicated VLSI's allowing the implementation of some special algorithms for audio synthesis is under development [1]. IRIS srl, acting as industrial partner, coordinates the whole work; his proposal is related to the design of some ASICs (inclusive of a floating point unit core) for a physical model implementation. IEI/CNR and DEIT (Dept. of Information Eng., Univ. of Pisa) as partner units, propose the prototyping of a ASIC (named Synth), a dedicated chip for the integration of a very large array of sinusoidal oscillators, able to produce around one thousand sinus [2,3]. The AddSynth machine consists of several chips arranged as a coprocessor in conjunction with the IRIS's section and allows cost effective facilities for synthesis and processing in the frequency domain.

It is well known that the additive synthesis implementation drawback is the computational cost for the oscillator parameters control: at least three parameters must be updated at micro-level for each oscillator. Since the update transfer is usually high, attention has been focused on the reduction of the amount of data from the host processor (or other input devices) toward the synthesis system. For that purpose we mainly considered amplitude data reduction and spectral data interpolation techniques [4,5,6].

2. ADDSYNTH SYSTEM ARCHITECTURE

The AddSynth machine architecture consists of a controller and a set of Synth chips, each one including an array of second order IIR resonant filters; a pipeline arrangement of up to 7 Synth chips allows to produce up to 8442 sinus (see fig. 1).

The internal arithmetic is 24 bit and the samples of the 8 digital audio out channels are represented by 20 bit at the frequency of 44.100 Hz.

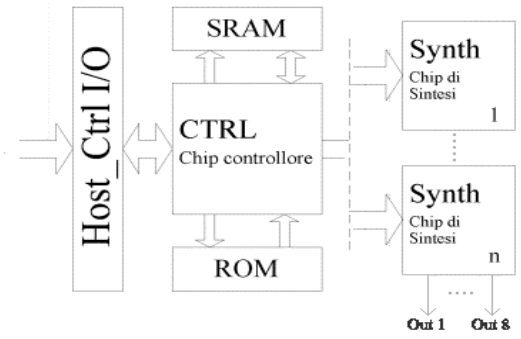

Figure 1. HW architecture of AddSynth system.

Usually, the host computer which generates musical events cannot guarantee to supply the huge data flow requested for feeding the oscillators parameters. For that, we included in the system a controller which processes incoming special command blocks and sends proper data to the synthesis chips.

The Synth chip architecture is shown in fig 2.

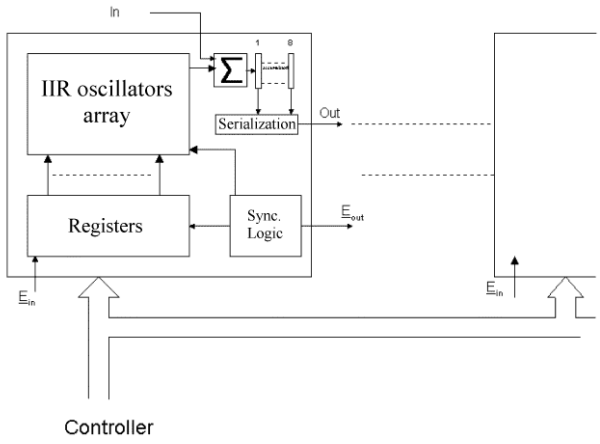

Figure 2. *Synth chips architecture*

Two main tasks are performed the first one consisting on updating the parameters in the input registers and the second one computing the samples of the specified oscillators.

First process

- The controller CTRL enables the first Synth-chip to receive data every period of length T (a 128 samples frame)
- CTRL starts writing data on the bus
- First Synth-chip receives and stores data on input registers
- First Synth-chip enables the second chip (and so on..).

This process continues until the last chip sends an E-out signal to the controller.

Second process

- Oscillators of each chip compute samples
- Results are stored on the set of 8 accumulators and sent to the next chip
- The last chip sends data to the output block.

Fig.3 shows the operators and connections schema of a single oscillator cell based on a typical IIR resonant filter.

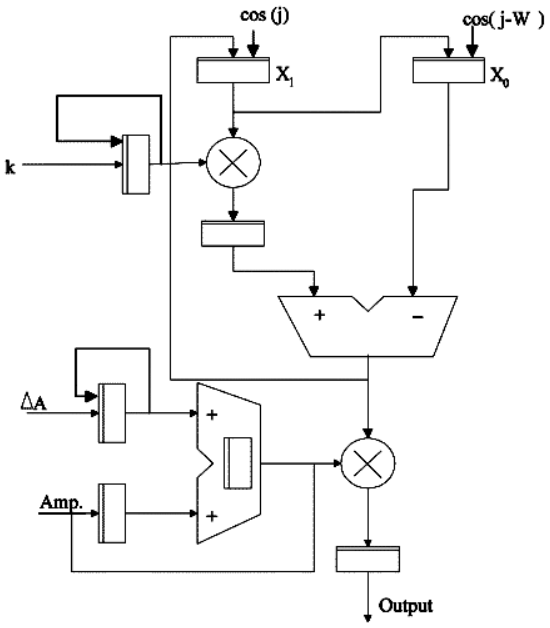

Figure 3 *Block diagram of a single IIR based oscillator with the amplitude linear interpolator.*

Each Synth-chip receives a data block consisting of values for the oscillators every T period of 128/44100=2.9 ms. This value has been chosen for getting sound without audible discontinuities (as reported in literature, granularity of the controls must be shorter than 3 ms for smooth variations of the sound parameters) and to be within a reasonable data flow bandwidth. Each block contains data for the frequency, the amplitude and the phase for each oscillator, that is : *k, Amp, ΔA, cos(j), cos(j-W),* of Figure 3.

The frequency of the oscillation f is determined by the k parameter following the expression $k=2\cos(2\pi T_c * f)$. The amplitude smoothing within each frame is provided automatically by *ΔA* factor into 128 points of interpolation. Other parameters necessary to the IIR cell, such as the initial constants and the other variables of the synthesis algorithm (Xi), are updated at each frame to maintain the phase consistency. Details on the IIR resonant filter implementation can be found in [2, 3].

A software simulator, developed by DEIT - University of Pisa, implements the tasks of a number (up to 7) of Synth chips. With an appropriate input file the simulator generates an output file of samples which permits to control the performances of the system and a quality evaluation of the generated sound. The simulator has been successfully tested on a set of reference signals.

The architecture of the software running on the host (at the moment under development) is shown in fig. 4. The first demo board with a single synth chip will fit in a standard PCI slot and it should be available at the beginning of next year.

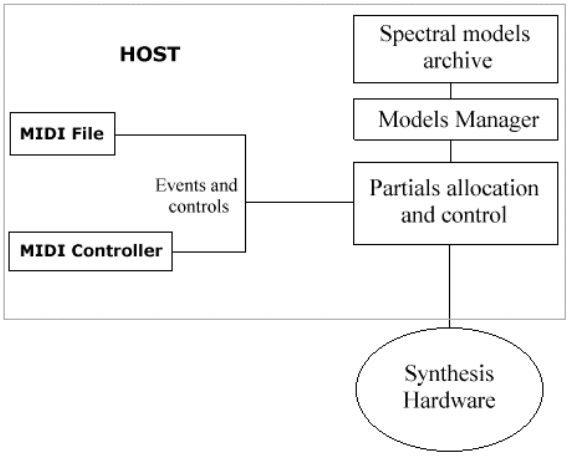

Figure 4. *Processes running on the host*

3. SIGNAL ANALYSIS AND SPECTRAL DATA MANAGEMENT

In order to update the huge amount of parameter necessary for the synthesis, the use of spectral data reduction methods is required. For this purpose we developed specific procedures taking into account well-know linear piecewise approximation techniques [3] which strongly reduce the amount of data regarding the evolution (envelope) of each partial amplitude resulting from the analysis of signals of real instruments. Some choices have been made taking into account the hardware specification of our system.

The signals generated using these techniques are then compared with the original ones in order to evaluate the required and expected quality of the sounds generated by the AddSynth system. The main steps are here summarized as follows:

a) *Signal Analysis by STFT*

b) *Linear approximation -spectral data reduction-*

c) *Data block setting for the of the AddSynth machine (or SW based simulator.*

As regard to step a), i.e. the analysis of the time-varying signals, we faced the typical processing problems for stating the fundamental frequency, the length and the type of the window used in the FFT, the overlapping degree, etc…

In [7] Serra and Smith proposed a spectral model based on a deterministic plus stochastic decomposition. That is, any audio signal can be decomposed into the sum of two signals:

$$v(t) = d(t) + s(t)$$

where *d(t)* is the deterministic part and *s(t)* is the stochastic part. The deterministic part is a sum of sinusoidal oscillations, *slowly* time-varying, while *s(t)* is the noisy-part, e.g. the attack phase of a plucked string sound. In the analysis we considered only the first part even if we tried to add the second part with a fast pseudo-random modulation of partials frequency (see later).

Therefore, it is important to remark that the sinus components of the deterministic parts must change slowly in their characteristics (pitch and amplitude).

We do not take in account the phase at the origins, according to several well known psycho-acoustic experiments, and we set it randomly in the synthesis phase.

During the experimentation, we also limited our tests to sounds with fixed pitch partials. So, we focused our attention in modeling the partials envelope. Goal at point (b) (spectral data reduction) has been faced with the linear piecewise approximation by balancing the error threshold between the original and the reduced amplitude envelope and the number of breakpoints.

We used the same number of breakpoints for all partials. Although this may seems a restriction, we found the following interesting features:

- *memory space saving*
- *faster execution*
- *same quality as independent breakpoints, on most of sounds.*

The processing is based on the *error* computed by the following equation:

$$\text{Error} = \frac{1}{N_{frames}} \times \sum_{n=1}^{Nframes} \sqrt{\frac{\sum_{k=1}^{N_{har}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{har}} (a_k(n))^2}}$$

where:

$a'_k(n)$ approx. amplitude of k harmonic at tn,

$a_k(n)$, original amplitude of k harmonic at tn

N_{har} number of used harmonics,

N_{frames} number of used windows in the analysis.

However, this function does not guarantee a good response on sounds with an attack phase with a richness of components. In these cases we used the next function, which gives more weight to the attack phase.

Here W1 and W2 are respectively the two weights of the attack phase and the remaining part of the sound.

$$\frac{W_1}{FrameAttac-1} \times \sum_{n=1}^{FrameAttac-1} \sqrt{\frac{\sum_{k=1}^{N_{har}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{har}} (a_k(n))^2}} + \frac{W_2}{N_{frame} - FrameAttac + 1} \times \sum_{n=FrameAttac}^{N_{frame}} \sqrt{\frac{\sum_{k=1}^{N_{har}} (a_k(n) - a'_k(n))^2}{\sum_{k=1}^{N_{har}} (a_k(n))^2}}$$

Setting W1=W2=0.5 means to give the same weights to the two parts. The management of the linear piecewise approximation of the envelopes is performed automatically by the controller section, preventing the continuous updating of the parameters

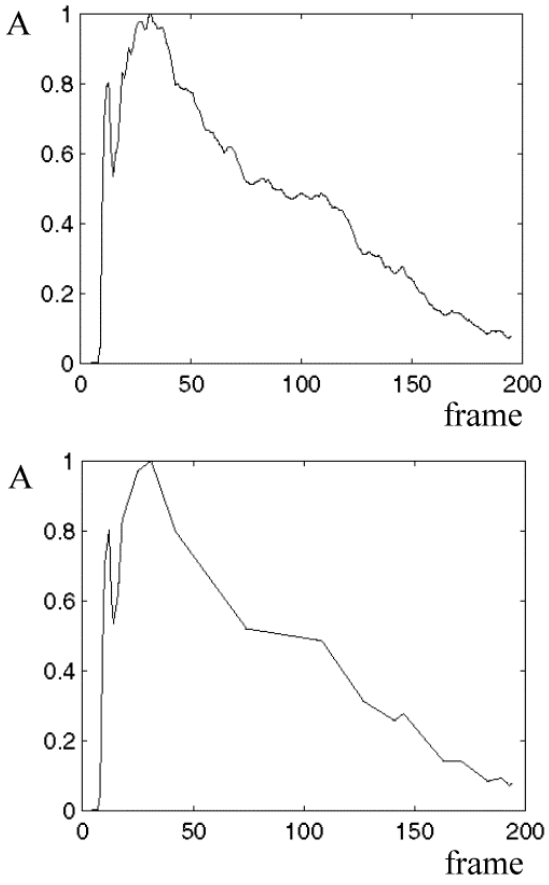

Figure 5. *Original and reduced amplitude envelope of a partial*

(amplitude and frequency) of the partials by the host, which will update the data only if necessary (for example in case of changing of the slope).

A clustering technique, which permits to group and organize the data block set, is used to manage the amount of the partials.

The Toolbox so allows to analyze an audio file, to reduce its instantaneous variations of the amplitude envelopes of the

partials and, finally, to generate a commands sequence to be sent the synthesis system simulator.

In this preliminary phase we used the MATLAB environment: MATLAB's Signal Processing Toolbox to easily compute FFT and other typical DSP operations. It allows easily drawing of spectral data too, very useful for comparing the results of the reduction algorithm as shown in fig.6 and 7.

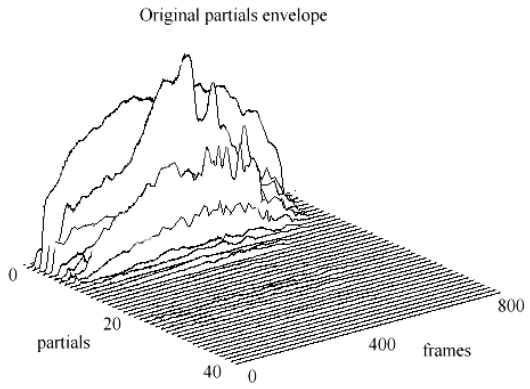

Figure 5. *Original partials envelope: oboe-C*

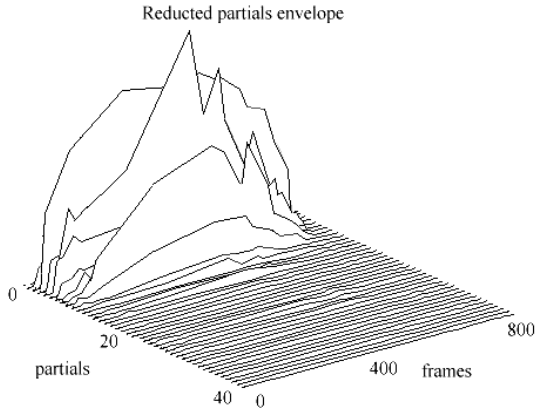

Figure 6. *Reduced partials envelope: oboe-C*

4. SPECTRAL DATA PROCESSING

A data structure of timbre model which summarizes the dynamic spectra characteristics of a musical signal has been proposed.
Some trials for transforming the parameter values with the aim to get more realistic simulation of real sound, in particular by adding noise to the attack section in the ADSR envelope, are under development.

In fact noisy sounds are very difficult to synthesize using additive synthesis without using a huge number of partials. We tried to add a noisy attack to plucked sounds by modulating the frequency of the partials at each frames in the attack phase.

As we can see in fig.7 we obtained a noise band centered in the partials frequencies very similar to some acoustic instruments. Besides, we are now studying the variations of the partial amplitude versus overall tone amplitude to investigate the problem of spectral interpolation (issue related to tone played

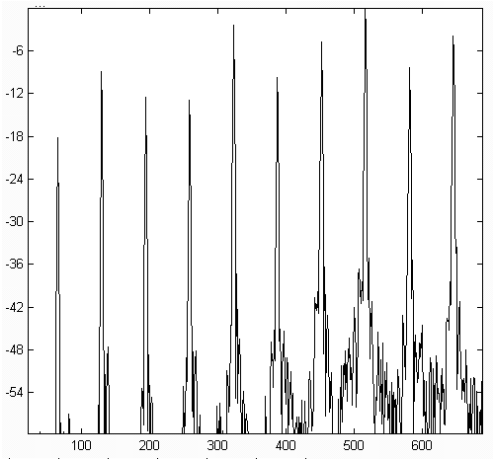

Figure 7. ***Noise simulation***

with different dynamic, pitch). As an example, we computed the ratio between the partials amplitude of a sound (e.g. at a level *piano)* and the correspondent amplitude of the same sound at a different level of expression (e.g. *forte*).

By interpolating these two extreme points we can easily get a variety of intermediate sounds with a rather good quality with a simple technique if compared to other morphing techniques [8]. In this way we can use only one spectral model and one ratio-vector instead of a lot of models at different levels of amplitude.
For the partials amplitude we used the amplitude average at each frame computed over the whole sound (fig.8).

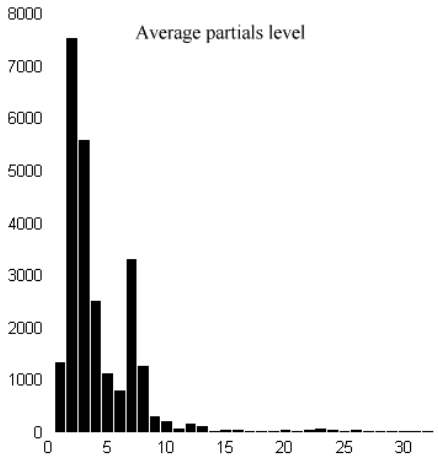

Figure 8. *Average partials levels*

Our system can generate partials with very close frequencies, what cannot be easily reached using Inverse Fast Fourier Transform methods [9,10]. We used this facilities in some

experiments of a slight de-tuning of the partials and obtained a sort of *spectral chorus* which dramatically enriches the generated sound.

Using this system will be very easy to implement many of the standard audio effects working in the frequency domain (filtering, phase shifting, flanging, vocoding etc.) and to simulate some different synthesis technique, too.

5. CONCLUSIONS

The computing power of last generations of personal computers and DSP processors renewed the interest on the additive synthesis technique not exploitable in real time during the 70's and 80's when researchers were forced to invent alternative (and in some case fortunate) methods such as FM, non-linear distortion and others.

A software toolbox for creating, recording and the management of spectral data blocks which allows the control of a special additive synthesis system, is under development. The toolbox makes it possible an easy and intuitive mapping between controls and parameters of the additive synthesis.

A special coding and clustering techniques of parameter values allows to reduce the data flow between the host computer and the synthesis chips, making possible to manage a very high number of partials.

6. ACKNOLEDGMENTS

The authors would like to thank the musicians Riccardo Lenzi, Stefania Tedesco, Alfonso Belfiore and the studio Vecchio Mulino Produzioni technicians for supporting the instruments sound recordings.

7. REFERENCES

[1] IRIS, CNR-IEI, Univ. dell'Aquila, Univ. di Pisa: "Design of VLSI ASICs and development of innovative algorithms for advanced application of audio signals generation by physical modeling and large scale additive synthesis" Middle Term Report of PF-CNR MADESSII (Materials and Devices for Solid State Electronics II, Sub-Project 3.2), pp. 181-184, Marzo 2000.

[2] F. De Bernardinis, R.Roncella, R. Saletti, P. Terreni. & G.Bertini "A new VLSI Implementation of Additive Synthesis" CMJ 22(3) pp. 49-61, MIT, Fall 1998.

[3] F. De Bernardinis, R.Roncella, R. Saletti, P. Terreni. & G.Bertini "An efficient VLSI Architecture for real-time Additive Synthesis of Musical Sounds" IEEE Trans. On VLSI Systems 7(1), 105-110, 1999.

[4] G. R. Charbonneau. "Timbre and the Perceptual Effects of Three Types of Data Reduction". CMJ, MIT 5(2), 10-19, 1981.

[5] A. Horner and J. Beauchamp, "A Piecewise Linear Approximation of Additive Synthesis Envelopes: A Comparison of Various Methods" CMJ, 20(3).

[6] M. H. Serra, D. Dubine, R. Dannenberg " Analysis and Synthesis of Tone by Spectral Interpolation" AES Journal, 38(3), 111-128, 1990.

[7] X. Serra and J.O. Smith, "Spectral Modeling Synthesis: A sound Analysis/Synthesis System based on a Deterministic plus Stochastic Decomposition" CMJ, MIT 14(4) pp 12-24 (1990).

[8] G. Cospito, R. de Tintis, "Morph: Timbre Hybridization Tools Based on Frequency Domain Processing". Proc. Workshop on Digital Audio Effects (DAFx-98) Barcelona, Spain, 1998.

[9] M. Barutti and G. Bertini. "An Implementation of the Additive Synthesis Based on FFT^{-1}". Atti X Colloquio di Informatica Musicale, pp 127-133, Milano 1993.

[10] X. Rodet, Ph. Depalle, "A New Additive Synthesis Method Using Inverse Fourier Transform and Spectral Envelope" Proc of ICMC, San Josè, CA. USA, pp. 410-411, 1992.