New DSP applications on FTS

François Déchelle  Maurizio De Cecco  Enzo Maggi  Norbert Schnell
dechelle@ircam.fr dscecco@ircam.fr maggi@ircam.fr schnell@ircam.fr
IRCAM - 1, place Stravinsky - 75004 Paris

Abstract
This paper describes new real-time applications on FTS. A brief recall of FTS features is given, together with informations on supported platforms. The FTS object libraries are described. A view of the development environment is given. The FTS vector library is described. The future extensions of the DSP objects programming interface are presented, allowing more flexibility in scheduling. The latest developments in DSP are described.

1 Introduction
FTS is the Ircam programming and runtime environment for signal processing oriented real-time applications. Its architecture has been presented in details at ICIMC'95 (Déchelle and De Cecco,1995).

FTS provides a real-time multi-threaded executive and a message-based object system.

The object system provides a specific programming interface for DSP objects. The execution of DSP operations is supported by the FTL vector computation engine [Déchelle and De Cecco,1995].

2 Current architectures
2.1 Currently supported platforms
Current platforms running FTS are :

- Ircam Signal Processing Workstation
- Silicon Graphics workstations, with native audio system
- PCI and Power-PC Macintosh and Davista multi-processors Macintosh, with native audio or add-on audio boards.

2.2 Benchmarks
The following table gives some benchmarks comparing the performances of the different supported platforms. The performance is measured as the maximum sampling rate that can be achieved for a typical patch. All the benchmarks are for one processor. For the ISPW platform, the ISP functions are coded in assembly. For the other platforms, the whole code is C.

<table>
<thead>
<tr>
<th>Platform &amp; Hardware</th>
<th>kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGI Indy (R4000, 300 MHz)</td>
<td>310</td>
</tr>
<tr>
<td>ISPW (800, 40 MHz)</td>
<td>180</td>
</tr>
<tr>
<td>Mac 8500 (Power-PC, 133 MHz)</td>
<td>330</td>
</tr>
</tbody>
</table>

*: estimated

3 FTS object libraries
In order to better organize the objects set, the objects are now partitioned as FTS modules. The FTS module abstraction is a data structure that is part of the FTS kernel [Déchelle and De Cecco,1995] and provides mainly a simple way to install the objects classes.

The object code can be linked to the FTS executable statically (i.e. before execution) or dynamically (i.e. during execution) using a dynamic linker such as dl on a Silicon Graphics platform. The fact that an object is statically or dynamically loaded requires no changes in the code, thus allowing the user to choose at initialization time which sets of objects will be used and to configure FTS according to its needs.

The module organization is the support of both static and dynamic loading of objects. Through a simple provide/require mechanism, the fact that an object class is already installed can be easily tested, and if not installed, this can also trigger the dynamic loading of a class on a platform that supports it.

4 FTS as a DSP programming environment
Several clients can be connected to the FTS server simultaneously, using the ftsd network daemon.
5 The FTS vector library

To achieve complete reusability of FTS DSP objects, a library for vector arithmetic and signal processing has been developed for FTS. This library provides basic vector/vector and vector/scalar arithmetic and logical functions and basic signal processing operations, namely FFT.

The API of the FTS vector library is platform independent. The implementation of each function is platform dependent and can use a platform specific optimized library. A portable C version is provided for each function.

To achieve better performance on pipelined processors, the library provides 2 versions of each function, with and without loop unrolling. The "unrolled" version can be used only if vector size is a multiple of the unrolling factor. For each function, a FTL-callable version is also provided.

6 New DSP programming model

An extension of the DSP scheduling model of FTS is under development, to ease implementation of advanced DSP algorithms, for instance involving FFT. A unified framework will be introduced soon, handling both multi-rate processing, conditional execution and post-optimization. A more flexible and extendible typing system will be added to the DSP computation engine.

7 New DSP objects

7.1 Additive synthesis and processing

The latest synthesis and analysis algorithms developed at Ircam, like the FFT-1 additive synthesis [Rodet et al., 1993], have been implemented as a set of FTS objects.

Robust pitch tracking and score following, including Miller Puckette’s work [Puckette,1995] on voice score following for Philippe Manoury’s De Echo, are now available.

7.2 Physical modeling

Work is in progress to implement under FTS a physical modeling synthesis software. For such synthesis algorithms, a sample by sample computation mode will be introduced.

8 Summary

We describe in this paper the current state and the evolutions of FTS, making it a flexible, modula- lar and evolutive environment for DSP programming and real-time control. We also introduced the latest developed DSP applications.

References


[Dechelle and De Cecco,1995] François Décheule, Maurizio De Cecco, The Ircam Real-Time Platform And Applications. ICMC 95, Banff.


