MARS - DSP environment and applications

F. Armanini, L. Bizzarri, E. Favezza, A. Polidori

IRIS S.r.l.
Ponte Le Selva 151 - 03018 Pianello (FR) ITALY
Fax: (775) 533343 - Tel: (775) 533441 - E-mail: nicola.learna@eurokoms.ie

ABSTRACT

MARS is a system developed by IRIS composed of a sound generation board (SM1000) and an integrated package (EDIT20) running on a host computer. This paper will describe the two environments of the EDIT20 package devoted to DSP applications: ALGO and ORCH. The basic level ALGO, a real-time interactive and graphic environment, allows the user to create algorithms by connecting primitive blocks. A set of debugging tools is also available to test the algorithm during its development. The higher level ORCH is an interactive graphic environment that defines an orchestra as a set of different algorithms. Each algorithm can be duplicated in several clones and linked to the inputs and outputs of the SM1000 board.

1. OVERVIEW

MARS (Musical Audio Research Station) is a tool for the interactive design and implementation of digital audio algorithms particularly oriented to musical applications. It also allows musicians to create their own performance environment by mixing and configuring many different algorithms. MARS is based on the SM1000 board which consists of a X20 DSP (Cavaliere, et al. 1992) chip developed by IRIS controlled by a Motorola 68002, and of a set of libraries (Audrenocc, et al. 1992) devoted to communications and real-time handling.

EDIT20 is the standard environment for programming and controlling MARS. It derives from a series of tools devoted to sound generation and processing, originated with MUSIC5 (Mathews 1969) and evolving to the current class of interactive graphic editors (Sicotti 1991; Lindemann, et al. 1991). This paper describes the ALGO editor which defines the algorithm, and the ORCH editor which duplicates and combines several algorithms in a musical context. For advanced DSP applications it is possible to define the algorithms with the assembler ASM20 (Favezza and Prestipiciomo 1990), combine them with the ORCH editor, or directly control the SM1000 board with user routines which use the communication libraries.

![Figure 1. DSP environment in MARS workstation](image)

2. ALGO EDITOR

The ALGO editor allows the user to create and debug DSP algorithms in an interactive, graphical and real-time context. For the ALGO editor an algorithm consists of a collection of:

- interconnected atomic objects called modules,
- configurable variables called entry points, which define the interface toward the TONE editor;
- constants called local points because they are not exportable to other editors;
- audio inputs/outputs called audio bus connections, to define the interface with the ORCH editor.

A module consists of a body, which is a predefined and optimized microprogram segment, and a collection of input/output pads, which correspond to memory locations of the X20. These pads may be used for connection to other modules, by sharing memory resources, for entry points, for local points and for audio bus connections. The semantically incorrect connections are not allowed and raise error messages and warnings.
2.1 Module classes
The system provides a module library (Mathews 1969) divided in several logical classes grouped in families (see Fig. 2). The basic family modules perform low-level operations such as adding, reading an external wave table, etc. A generic DSP algorithm can be realized using these elements. The generators family modules, such as oscillators, etc., are generally used to build sound synthesis algorithms. The transformers family modules are useful to create complex sound processing structures using objects such as chorus, echo, etc.

The reader may note that the basic family modules can be assembled to obtain all the complex elements of the two last families (De Poli 1983). Nonetheless, the complex modules generate more optimized algorithms and represent objects more familiar to musicians. The LFO (Low Frequency Oscillator) and the Envelope are special modules, because they have a set of hidden entry points that control their dynamical behaviour, and because their physical implementation is strongly dependent on specific aspects of the SM1000’s hardware.

<table>
<thead>
<tr>
<th>BASIC FAMILY</th>
<th>GENERATORS FAMILY</th>
<th>TRANSFORMERS FAMILY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Some elements of the class</td>
<td>Class</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>Add, Mul,</td>
<td>Table look-up oscillators</td>
</tr>
<tr>
<td>Logic</td>
<td>Man, Min, Abs,</td>
<td>Vio, Tim, Syn.</td>
</tr>
<tr>
<td>Read/Write</td>
<td>Read Wave Table.</td>
<td>Complex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Modulator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Synthesizer</td>
</tr>
</tbody>
</table>

Figure 2. Table of Module classes

2.2 Interactive debugging tools
The user may test the data-flow correctness using a set of probes: each one can be inserted on every block diagram connection and then disconnected or moved to a different block diagram point. This allows the user to analyze and listen to the signal produced at each point of the algorithm, in one of the following ways:

* Acoustical probe: signal is sent to available loudspeakers.
* Numerical probe: signal value is displayed.
* Graphical probe: signal is sent to the four-track software oscilloscope.

During the debugging session it is also possible to control the X20 step-by-step to test the probes status after each microprogram cycle.

2.3 Tone interface
The algorithm entry points are configured by means of the TONE editor and can be divided into two classes: simple parameters, whose values may be constant or variable and which are directly loaded in the relative memory locations, and structured data, such as Envelopes and LFOs hidden entry points, that depend on the updating of more variables (Palermo 1992). The system allows one to link these variables with MIDI devices to control their dynamic behaviour. Configuration may take place during algorithm development in the ALGO editor: one fixes the values of the local points and builds a prototyping tone for the entry points. If this prototyping tone has been saved, it may be also loaded into the TONE editor.

3. ORCH EDITOR
This editor allows to set and connect a collection of different algorithms to build the ORCH environment. Here the algorithms are considered, like the modules in the ALGO editor, as atomic units defined only by their body and input/output pads. It should be noted that algorithms could be considered as instruments or as effects in relation to the presence of input pads. Algorithms that have input pads could take signals produced by others via a bus connection.

The Audio Buses are another object class handled by the editor. These are the link between different algorithms and constitute the graphical representation of the four ADGs and eight DA3s of the SM1000 board. The latter are divided in two subcategories of four boxes relative to each X20 chip. These two subcategories can be connected by four switches. The interaction with the user is driven by a menu and mouse-based interface that allows the following actions: adding, deleting and selecting algorithms, creating an algo-family by duplicating their clones (that is changing the algorithms polyphony), distributing the same algo-family between two X20 processors (the system handles these two objects like they were one), connecting all algorithm pads with audio buses, compiling an orchestra, sending the orchestra’s configuration to the SM1000 board, changing the status (open/closed) of the bus switches.

All these actions, except the last one, are performed first in the host computer memory and then sent to the DSP board. In
relation to the current number of connections and switched states, the system automatically controls the output gains to avoid saturation. It is also possible to adjust the right gain levels by hand, with a set of dedicated faces.

Once an orchestra configuration is defined by the selection of algorithms and their clones duplication, it is possible to use the MARS as a powerful MIDI musical instrument. This feature is performed by the TMA* editor (Pommeret and Sapin 1993).

4. EXAMPLES

Figure 3 displays what the user actually sees: on the left is shown an algorithm, realized in two different ways. The first one uses complex modules and the second one only basic modules. This example shows most of the objects normally used in the creation and debugging of an algorithm:

- OUT triangles are the audio bus connections which represent an interface toward the ORCH editor.
- The inputs which have a numerical value, also represented as little boxes with the letters C (constant), are local points.
- The inputs "any", "free" and "call word", also represented as little boxes with the letter D (dynamic), are examples of entry points. They are configurable parameters and represent an interface toward the TONE editor.
- The loudspeakers, numbered boxes and syringes are the probes useful in the debugging phase which give, respectively, an acoustical, numerical and graphical feedback to the processed signal.

On the right is shown a set of algorithms connected to boost. The algorithm named MY_INSTR has a polyphony of 5 clones and is connected to the second bus of the two X20s (the second switch is closed), while the processing one (MY_EFCT) is a reverberating unit and takes the signal of the last algorithm from the second bus, sends its stereo outputs to the other two buses on the second X20. IN/OUT acquires a signal from the second ADC and sends its output to the first DAC.

5. IMPLEMENTATION

The implementation is based on a "C" library that interprets the commands received from the user interface and installs the microprogram code on the DSP machine in real time. The Module Library uses a dynamic list of all the modules, that describes the actual program running on the two X20 chips. In this way, every modification performed on this list has an immediate effect on the processors via the library of microprograms. This library can be divided into two parts:

- the MSERVER (function library), which handles the microprogramming of the DSP board, performs memory allocation and interacts with the graphic editor;
- the Mod_DL (module class library), a collection of X20 precompiled and optimized microprogram routines.

All tasks are done in real-time when the library is under the control of the ALGO editor, and are developed in the host computer where it acts in the ORCH mode. The necessity of compiling and sending to the DSP board an orchestra configuration is justified by the resulting optimization of resources, which is possible only when all the algorithms are selected, duplicated, and connected. The Mod_DL is the only part of the program that has an exact knowledge of the particular DSP architecture used. So the Module Library masks the links with the X20 processor to the upper system levels.
MARS allows the user to create applications ranging from simple sound generation to a multimedia environment in which MARS interacts with other host computers. Some applications may require a higher optimization of MARS resources because of their microprogram complexity, while others may require a different interaction with the outside. In such a case, it can be useful to leave the EDIT20 environment and program or configure the machine in a different way.

In order to better employ the available microprogram space, the ASM20 language can be used. It lies close to the X20 architecture, exploiting the DSP parallelism and pipelined data flow. The code generated by ASM20 also allows a custom program to create an orchestra compatible with the TONE editors, but in this case the number of algorithms and their polyphony is fixed. An example, the ASM20 language can generate 32 PM 4-operator chains instead of the 18 generated with the ALGO ORCH editors. The use of ASM20 language also allows one to develop complex algorithms that simulate virtual instruments based on a physical model (Paladin and Rocchesso 1992).

Applications using MIDI communication in a different environment than the EDIT20, or not using MIDI at all, can employ the "C" Transmission Library (Andreucci, et al. 1992) to interact with the MARS system. Two examples of such applications, realized in IRIS, are the following:

1) LPC based voice synthesis: the algorithm, implemented on the X20 using the ASM20, is realized by means of a 25th order prediction filter, coupled with an excitation structure generating noise or glottal pulse. The code that synthesizes one voice occupies approximately half of the microprogram space of one X20. The coefficients of this filter, precalculated on a sampled phrase, can be read to the X20 memory locations by a dedicated process running on the host computer, that also displays the corresponding vocal tract shape. To allow interaction, the coefficients relative to the different parts of the phrase are mapped on the X-axis of the screen area, and the playback pitch is mapped on the Y-axis. Moving the mouse the user can "scratch" the phrase and vary its prosody, while changing other parameters with the keyboard.

2) 2048 points pipeline FFT analysis with generalized Hamming window overlap of 1024 points on the first X20. 214 triangular oscillators for additive resynthesis on the second X20. A dedicated process, running on the host computer, displays the analysis results with the usual spectrum-analyzer options such as 3D spectrum, lin/log scale, etc. A new spectrum is computed transforming the original one by parametric rescaling and translating of the bands. This spectrum is used to drive the amplitude of the 214 oscillators.

User applications can also be added to the SM1000 performance environment extending the RT20 operating system with new dedicated processors (Andreucci, et al. 1992).

7. CONCLUSION

The DSP environment of the MARS workstation allows multi-algorithm design and configuration using two powerful tools: the interactive graphical editor EDIT20 and the assembler language ASM20.

The capability of MARS to develop algorithms in an easy and interactive way will be used to explore new methods for sound analysis, synthesis and processing both through IRIS internal use and through collaborations involving other research centers, universities, conservatories and musicians.

MARS can be used not only for musical applications but also as a general purpose DSP unit. For example IRIS is developing an European project, called ECHOSSEA, to explore the sea bottom using an echography system.

REFERENCES


347