Real Time Sound Processing & Synthesis on Multiple DSPs Using the Music Kit and the Ariel QuintProcessor

David A. Jaffe
P.O. Box 4268, Stanford, CA. 94309
david@jaffe.com

Julius O. Smith
CCRMA, Music Dept., Stanford, CA. 94305
jos@ccrma.stanford.edu

Abstract

This paper is an update on the 4.0 release of the Music Kit, including recent work in real-time sound processing and porting to multiple-DSP architectures such as the Ariel QuintProcessor.

1 Introduction

We have recently ported the Music Kit to a multiple-DSP architecture and extended it to do real-time sound processing. The focus of this discussion is on the Ariel **QuintProcessor,** a 5-DSP board for the NeXT cube. However, in view of NeXT's shift toward PC hardware, the implementation was designed to be readily portable to other DSP cards, such as will soon be available on PCs running NeXTstep 486.

1.1 What is the Music Kit?

The Music Kit is an object-oriented software system for building music, sound, signal processing, and MIDI applications on the NeXT computer. It has been used in such diverse commercial applications as music sequencers, notation packages, computer games, and document processors. Professors and students in academia have used the Music Kit in a host of areas, such as music performance, scientific experiments, computer-aided instruction, and physical modeling [Jaffe, 1991]. The Music Kit is the first to unify the MIDI and Music V paradigms, thus combining interaction with generality. It was developed by NeXT Computer, Inc. from 1986 to 1991 and has been supported since then by Stanford University and developers such as Pinnacle Research, Inc. For further information, see [Jaffe and Boynton, 1989] and [Smith *et al.*, 1989]. The Music Kit is available free of charge via **ftp** from **ccrma-ftp.stanford.edu** (email: **musickit@ccrma.stanford.edu**.) It is also available on CD ROM from the Bay Area Next Group at P.O. Box 1731, Palo Alto, CA 94302 (email: **Info@BANG.org**.) To subscribe to a Music Kit news group, send to **mkdist-request@ccrma.stanford.edu**.

1.2 What is the Ariel QuintProcessor?

The Ariel **QuintProcessor** [Ariel, 1990] is a board for the NeXT cube that contains five 27 MHz DSP56001 signal processing chips, each with its own bank of static RAM and pair of serial ports. The DSPs are arranged in a star configuration, with one "hub" and four "satellites." The 56001 is well-known as a low-cost and powerful signal processor that is well-suited to musical uses. The **QuintProcessor** ("QP") augments the power of the 56001 by providing the following additional capabilities:

- 0 wait-state static RAM (32K words for each of the satellite DSPs and 8K words for the hub DSP).
- 256K, 1M or 4M words of dynamic RAM for the hub DSP. Automatic refresh hardware for the DRAM.
- Interprocessor communication hardware.
- Two NeXT-compatible DSP ports and a larger connector that brings out six more serial ports.
- SCSI controller and real-time clock for hub DSP.
- Rapid NeXTbus access to the host interfaces of the DSPs and to the other QP hardware.

For information on purchasing an Ariel **QuintProcessor**, contact Ariel Corp., 433 River Road, Highland Park, NJ 08904. (201) 249-2900, (201) 249-2123 fax.

1.3 Porting Strategy

There are three features of the QP that make it ideally suited to a Music Kit port. First, it is based on the DSP56001, for which there is already an extensive body of Music Kit software. Second, it provide rapid access to the DSPs' host interfaces, which enables use of a simple, register-based software interface. Third, it supports DSP serial ports that allow real-time sound to enter or leave the QP without having to involve the NeXT's main CPU or the NeXTbus.

These three requirements—DSP56001, rapid host interface access, and serial ports—are precisely those needed from any PC card that is to be used with the Music Kit with a minimum of porting effort.

2 Serial Port Sound Output

In order to avoid using the main CPU to output sound, we support the Music Kit's sending its sound output to the DSP SSI serial port. The sound samples are then converted to analog sound by an external DAC such as the Ariel **ProPort**, or to AES/EBU digital sound by a device such as the Singular Solutions **AD64x**.

Note that the support for SSI output is not restricted to the QP. With a simple **NeXTstation** or **NeXTcube** and one of the above-mentioned (or other) serial port devices, it is now possible to do direct digital transfers to DAT or to use external DACs which often have better fidelity than the NeXT DACs. Another advantage to this approach is that we bypass the buffering ordinarily required when

communicating between the DSP and the NeXT DACs, thus removing a significant source of latency.

The programmer indicates that the output of the DSP is destined for the SSI serial port by sending the Objective-C message `setSerialPortOutput:YES` to the Orchestra object that represents the DSP. No change is necessary to the SynthPatch or UnitGenerator code. Since all SynthPatches already have an output UnitGenerator (such as *Out1aUG*), the DSP system simply routes the sound from this output to the serial port.

A new class called *DSPSerialPortDevice* encapsulates the details about the external device that is plugged into the DSP serial port. The Music Kit provides subclasses of DSPSerialPortDevice support various commercially-available devices. For example, if you have an Ariel **ProPort**, you simply send the message `setSerialPortDevice:[[ArielProPort alloc] init]` to the Orchestra object, which then defers to the *ArielProPort* object to set up the external device. The ArielProPort object also takes care of sending the appropriate commands to the DSP SCI port to set the sampling rate. The Music Kit's DSP system automatically handles half sampling rates. E.g. if the serial port device supports only 44100 and the sampling rate of the music is 22050, the Music Kit will automatically up-sample the sound data.

Commercial devices that the Music Kit currently supports include the Stealth **DAI2400**, the MetaResearch **Digital Ears**, the Ariel **ProPort**, the Ariel **DatPort**, the Ariel **Digital Microphone** and the Singular Solutions **AD64x**. If a hardware designer creates a new serial port device, he need only subclass DSPSerialPortDevice and override a few methods—the Music Kit then automatically supports the new device. One interesting new device is a quadraphonic interface, developed by Fernando Lopez-Lezcano, Stephen A. Davis and Atau Tanaka at CCRMA [Lopez-Lezcano, 1993] [Tanaka, 1991], which allows the Music Kit to generate four-channel sound.

3 Serial Port Sound Input

Up until now, the Music Kit has supported synthesis, but not sound processing. The 4.0 Music Kit adds support for receiving 16-bit sound sent to the serial port from an external ADC such as the **AD64x** or **ProPort**, or an external AES/EBU interface such as the Stealth **DAI2400**. This sound input runs simultaneously with sound output, enabling incoming sound to be processed and sent back out the serial port.

As with serial sound output, the sound input capability can be used independent of the QP. Thus, any NeXT computer can now do real time signal processing. **ResonSound**, an example applicaton and programming example, demonstrates the signal processing capability of the 4.0 Music Kit. It resonates the incoming stereo sound with eight parallel all-pole comb filters, each with its own resonant frequency, feedback gain, left-right panning and gain. The user can change these parameters in real time.

Sound input support is enabled in a manner similar to sound output—you merely send `setSerialSoundInput:YES` to the Orchestra that represents the DSP. To create a sound-processing SynthPatch, you provide a sound input source UnitGenerator. For example, to receive sound input from the left channel, you include an *In1aUG* UnitGenerator object, just as you use an *Out1aUG* to send sound output to the left channel. The output of In1aUG can then be connected to any other UnitGenerators to create a signal processing network. Any number of In1aUGs may be instantiated; each gets a copy of the incoming sound so it is easy to create parallel banks of processing modules.

Since the SSI input and output may be used simultaneously, and since the SSI output path gets rid of buffer latency as described above, a real-time signal processor with no noticeable latency can be implemented using the Music Kit tools we have described.

4 Ariel QuintProcessor

The primary Music Kit class that supports the QP is the *ArielQuintProcessor* class, which serves a dual purpose. First, it is a subclass of Orchestra that represents the hub DSP. As such, it can be sent Orchestra messages to allocate UnitGenerators, SynthPatches, etc. But it also represents the QP as a whole and provides master control methods. The satellite DSPs are represented by instances of the subsidiary class, *ArielQPSat*, which is also a subclass of Orchestra. Creating an instance of ArielQuintProcessor automatically creates the associated ArielQPSat objects. Since a NeXT cube can hold up to three Ariel QP cards, there may be up to three ArielQuintProcessor instances.

An ArielQuintProcessor object provides for two options as to how sound output is produced: Each DSP may have its own serial port output device or the DSPs may be used as a team, sending their output to a single output device.

3.1 Independent Sound Output

Giving each DSP its own sound output allows for large numbers of output channels. For example, if each serial port output device supports stereo, ten channels of sound are possible. To use independent sound outputs, simply send the message `setSatSoundIn:NO` to the ArielQuintProcessor instance. This tells the hub DSP that it is not to merge sound from the satellites. You then send each of the ArielQuintProcessor and ArielQPSat instances the message `setSerialPortDevice:`, passing it the appropriate serial port device object. The main drawback to independent sound output is the financial cost of that many serial port devices, each of which can cost as much as $1000.

3.2 Merged Sound Output

The alternative is to use a single sound output device connected to the hub DSP's serial port. The other DSPs then send their sound to the hub DSP. To enable this mode, send `setSatSoundIn:YES` to the ArielQuintProcessor instance.

If you simply want to mix the satellite DSP's sound to the hub DSP's sound output stream, allocate (on the hub DSP) an instance of *ArielQPMix*, a new SynthPatch subclass that takes care of the mix for you. If, in addition, you want to process the satellite DSP's sound in some more sophisticated manner, such as reverberating or delaying it, you can design a SynthPatch that uses an *In1qpUG* UnitGenerator object. Each instance of *In1qpUG* provides sound from one channel of one of the satellite DSPs. As with the In1aUG UnitGenerator, any number of In1qpUGs may be used at once and several may be reading the same satellite DSP's sound stream. This makes it easy to create parallel banks of processing modules.

Of course, you may still have individual sound input on any or all of the satellite DSPs. For example, if satellite DSPs A and B each are connected to their own MetaResearch **Digital Ears** unit and the hub DSP is connected to an Ariel **ProPort**, six channels of sound input and two channels of sound output are available.

3.3 DRAM support

The large bank of dynamic RAM is very useful for reverberation, delays and other purposes. To access a section of DRAM as delay memory,

you instantiate an instance of *DelayqpUG*, a new UnitGenerator subclass, from the ArielQuintProcessor object. For example, to delay a satellite's sound, allocate an In1qpUG and connect it to a DelayqpUG, which in turn is connected to an Out1aUG.

DRAM auto-refresh is supported by the hardware. However, the frequent access pattern of many sound operations makes it unnecessary in many cases. Thus, the ArielQuintProcessor object allows you to turn auto-refresh on or off. If auto-refresh is used, it must be temporarily disabled when accessing DRAM. To simplify creating DSP unit generators, the Music Kit provides a set of DSP macros. In particular, auto-refresh details are hidden in the `begin_dram_access` and `end_dram_access` macros.

3.4 Resource Allocation from a Bank of DSPs

Automatic dynamic allocation and loading on multiple DSPs was an early feature of the Music Kit, but it is only with the QP support that its potential has been realized. You rarely need to think about which DSP is being used for a specific resource. SynthPatches, UnitGenerators and DSP memory (SynthData objects) can be automatically allocated on the first available DSP, taking into consideration availability of both DSP memory and DSP compute power. To do this, send the `allocSynthPatch:` message (or `allocUnitGenerator:` or `allocSynthData:`) to the Orchestra class itself, which manages the pool of DSPs. Alternatively, you may request that a resource be allocated from a particular DSP by sending the `alloc<`*`resource`*`>:` message to the Orchestra instance representing that DSP. For further allocation and optimization details, see [Jaffe, 1990].

3.5 Some benchmarks

The Music Kit running on the QP is roughly five times as powerful as the NeXT DSP, in terms of compute power. There is a certain amount of overhead on the hub DSP involved with merging the satellites' sound output streams. This is approximately 30% of the hub DSP's compute power. However, this overhead is offset by the faster clock speed of the QP's DSPs. Hence, a factor of five is approximately correct. In addition, the QP's power is augmented by the additional serial ports, additional DSP memory (32K instead of 8K standard) and DRAM.

As an example, at 44khz on the NeXT DSP, we can run in real-time about seven Karplus-Strong plucked strings with the Jaffe-Smith tuning, dynamics and sustain filters and stereo panning [Jaffe and Smth, 1983]. On the QP, we can run thirty-six at that samplng rate. At 22khz, we can run fourteen such patches on the NeXT DSP and fifty-eight on the QP. Actually, since the NeXT DSP can be used at the same time as the QP DSPs, the total for one QP in a NeXT cube is seventy plucked strings at 22khz. For three QPs in a cube, the total is a whopping 188 strings. Comparable benchmarks are obtained for other types of synthesis and sound processing.

3.5 Design Restrictions

The current QP design has a number of restrictions. A single SynthPatch or UnitGenerator object is constrained to run on a single DSP. Also, satellite DSPs cannot receive data from other DSPs. There is a way around this limitation: DSP serial ports can be chained so that sound is sent out one DSP's serial port and received by another's. In fact, the chaining of DSPs is a viable alternative to the hub/satellite topology.

Another limitation is the Music Kit's orientation toward sixteen-sample vectorized computation, which may be inappropriate for some applications. However, it is possible to write a unit generator that computes sound in larger blocks, then doles out sixteen samples at a time to its output. At the other extreme, events which must occur within a 16-sample "tick" can be implemented within a unit generator.

Finally, the current implementation does not support the SCSI controller on the QuintProcessor.

4 Efficiency

The low-level DSP code is highly optimized in a number of ways. The serial port sound input and output both use fast two-word interrupt handlers which write/read double buffered DSP memory. The buffer boundaries are checked automatically by the DSP system once every sixteen computed samples. This results in almost no overhead whatsoever (less than one percent per channel).

Similarly, the satellite DSPs use a fast two-word interrupt handler to write to the hub DSP, allowing them to run at maximum speed. The hub, however, is not able to take advantage of this optimization when reading data from the satellites, which is one reason why there is a noticeable overhead associated with the hub DSP.

The Objective-C allocation, loading and parameter update software is highly optimized on many levels, as described in [Jaffe, 1990] and [Smith *et al.*, 1989].

5 Other Recent Music Kit Developments

A full description of the new features of the 4.0 Music Kit release is beyond the scope of this paper. However, we would like to mention a few highlights:

• **Grasp**, an application that lets the user design synthesis and processing instruments in real time by dragging icons from a palette and connecting them together with virtual wires. **Grasp** is actually a Music Kit application itself—hence, the sound is heard instantaneously, with no computation phase necessary. However, it can also "de-compile" a patch into Music Kit SynthPatch source code, thus providing a convenient graphical way to design complex SynthPatches.

• Upgrades of **ScorePlayer** (Music Kit scorefile player) and **Ensemble** [McNabb, 1990] (algorithmic composition and performance environment) that take advantage of the serial port support. Thus, it is now possible to send **ScorePlayer** sound to the DSP serial port and to include instruments in **Ensemble** that process sound received from the serial port. We also plan to include support for the QP in these applications.

• The Conductor object can now synchronize to incoming MIDI time code. To select time code synchronization, you just send the message `setMTCSynch:aMidi` to the Conductor object that is to syncrhonize, where `aMidi` is the Midi object representing the NeXT serial port receiving the time code.

• MIDI time code generation is provided by the new *MTCPerformer* class.

6 Conclusion

The introduction of real-time sound processing capabilities into the Music Kit greatly increases its scope and flexibility. Owners of NeXT hardware can immediately take advantage of this capacity with the addition of one of the commonly available ADC boxes. DSP serial-port sound out eliminates unwanted buffer delays in sound processing and increases fidelity. The serial port approach makes it easy to extend the Music Kit to devices such as the Ariel **QuintProcessor**.

The multi-DSP QP/Music Kit combination is powerful, efficient and cost-effective. We recommend the QP to anyone with a NeXT cube; we also expect DSP cards for the PC to make this capability available at a lower cost per DSP on the PC platform.

The Music Kit is an advanced system for research and production, and we hope its development continues.

7 References

[Ariel, 1990] Ariel Corp. *Ariel QuintProcessor Installation, Technical and Programming Manual*. Ariel Corp.

[Jaffe, 1990] David A. Jaffe. *Efficient Dynamic Resource Management on Multiple DSPs, as Implemented in the NeXT Music Kit*. Proceedings of the 1990 International Computer Music Conference, Glasgow, Scotland,Computer Music Assoc., pp. 188-190.

[Jaffe, 1991] David A. Jaffe. *Musical and Extra-Musical Applications of the NeXT Music Kit*. Proceedings of the 1991 International Computer Music Conference, Montreal, Canada, Computer Music Assoc., pp. 521-524.

[Jaffe, 1989] David A. Jaffe. *An Overview of the NeXT Music Kit*. Proceedings of the 1989 International Computer Music Conference, Columbus, Ohio, Computer Music Assoc., pp. 135-138.

[Jaffe and Boynton, 1989] David A. Jaffe and Lee Boynton. *An Overview of the Sound and Music Kits for the NeXT Computer*. Computer Music Journal, MIT Press, **14**(2):48-55, 1989. Reprinted in book form in *The Well-Tempered Object*, ed. Stephen Pope, 1991, MIT Press.

[Jaffe and Smith, 1983] David A. Jaffe and Julius O. Smith. *Extensions of the Karplus-Strong Plucked-String Algorithm*. D. Jaffe and J. O. Smith. 1983. Computer Music Journal, **7**(2):56-69. Reprinted in *The Music Machine*, ed. Curtis Roads, 1989, MIT Press, pp. 481-494.

[Lopez-Lezcano, 1993] F. Lopez-Lezcano. *A four channel dynamic sound location system*. Proceedings of the 1993 Japan Music and Computer Society.

[McNabb, 1990] Michael McNabb. Ensemble, *An Extensible Real-Time Performance Environment*. Proc. 89th Audio Engineering Society Convention, Los Angeles, CA, 1990.

[Smith *et al.*, 1989] Julius O. Smith, David A. Jaffe and Lee Boynton. *Music System Architecture on the NeXT Computer*. Proceedings of the 1989 Audio Engineering Society Conference, L.A., CA.

[Tanaka, 1991] Atau Tanaka. *Implementing Quadraphonic Audio on the NeXT*, Proceedings of the 1991 International Computer Music Conference, Montreal, Canada, Computer Music Assoc.