Mark Kahrs* and Tom Killian†
*Department of Electrical and Computer Engineering
Rutgers University
P. O. Box 909
Piscataway, New Jersey 08855 USA
kahrs@winlab.rutgers.edu

†AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey 07974 USA
tom@research.att.com

Gnot Music: A Flexible Workstation for Orchestral Synthesis

Existing electronic instruments have a number of difficulties. We have identified three problems that we believe can be solved. First, although versatile, the MIDI protocol (Loy 1985) is too slow to permit rapid changing of instrument parameters. This is a direct result of MIDI's design—it was really designed as a gestural language and not a language to transmit large amounts of data such as envelope parameters. It is too slow to control many instruments simultaneously without running into synchronization problems. (For a full description of the problems with MIDI, see Moore 1988.)

Second, algorithmic (as opposed to sampling) synthesis is useful because it provides a parameter space that may be explored for creative effects. Frequency modulation (FM) algorithms (Chowning 1973) in particular yield sounds with rich spectra for relatively few arithmetic operations. Currently, the largest commercially available FM synthesis capability is that of the Yamaha TX-816 synthesizer. This device has eight identical modules, each with the capability of a Yamaha DX-7. Access to parameters is slow (via eight separate MIDI streams), and only a fixed set of operator connection topologies is available. As a result, "orchestral" synthesis is not truly possible; it is difficult to layer enough timbres without running into bandwidth difficulties due to MIDI limitations. (This is particularly true if the eight MIDI streams are demultiplexed from a single source.) The TX-816 has another limitation—it provides only analog outputs; these must in turn be mixed together with an analog mixer, thereby providing extra opportunities for noise and programming complexity.

Computer Music Journal, Vol. 16, No. 3, Fall 1992,

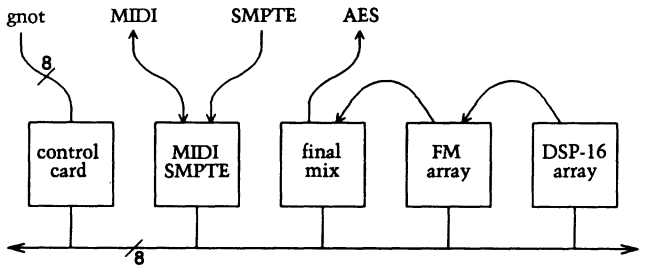
Finally, most signal processing is done by separate "effects boxes" after the sound is generated. This eliminates possible feedback between the signal processing and the audio output, except through the slow MIDI channel. It also requires extraneous digital-to-analog (D/A) and analog-to-digital (A/D) conversions that reduce the sound quality and add to the expense of the system.

The Gnot Music Workstation

The *gnusic* (Gnot music) project was created to overcome these difficulties. It is based on an experimental diskless workstation (the "Gnot," the name is not significant), featuring a Motorola MC68020 processor, a bitmapped gray-level display, and an interface to a 1.5 Mbit local area network. The Gnot was developed as part of a grand experiment in distributed computing using high-quality terminals connected to distributed file and compute servers over local area networks (Pike et al. 1990). We have been developing an experimental machine for sound synthesis based on using the Gnot machine as a real-time controller. Our principal purpose was to explore the notion of "orchestral" synthesis; that is, the real-time synthesis of large numbers of instruments comparable to that found in a modern orchestra.

In our system, instruments are controlled over a high-speed bus by a fast processor, sound samples may be produced by fully programmable digital signal processors (DSPs), and mixing and effects processing are integrated by using the same DSP for both functions. The output is an AES/EBU (Audio Engineering Society 1985) digital audio stream that

Fig. 1. The basic gnusic machine architecture.



can be fed directly to a compact disk mastering recorder or to D/A converters.

The processor (housed in the display base of the Gnot) has direct access to the computational resources of instrument cards over an 8-bit bus. The 8-bit bus is a limitation of the Gnot design; it is an extension of the MC68020's bus. Another implementation could have a wider bus with the accompanying increase in bus bandwidth. The bus is distributed over a backplane to the various cards; these cards can perform any number of functions depending on the card. The architecture of the basic machine is shown in Fig. 1.

To begin with, the 8-bit data bus and associated control signals are delivered to a control card at one end of the backplane. These signals arrive from the Gnot via a flat ribbon cable. The control card performs simple decoding tasks and buffers the signals onto the backplane. It also contains a SCSI interface for a disk for use when a local area network with a file server is not readily available. There are several different kinds of input/output cards. There is a SMPTE/MIDI card for interfacing to videotape recorders and keyboards, respectively. There are three different kinds of sound generation ("instrument") cards: a DSP-16 array, an FM array, and the "final mix" card. Each instrument card has an AT&T DSP-16 digital signal processor (AT&T 1987) for mixing the digital outputs. This processor must (a) mix the output from the card "upstream" with the sample generated locally, and (b) perform any effects desired with the leftover time. Such effects might include filtering or feedback control of the oscillators. The basic core includes a pseudo-dual-port static program memory for the DSP-16. Because the DSP-16 is so fast (80 nsec cycle time; the DSP-16A is even faster,

Table 1. Serial I/O cable lines

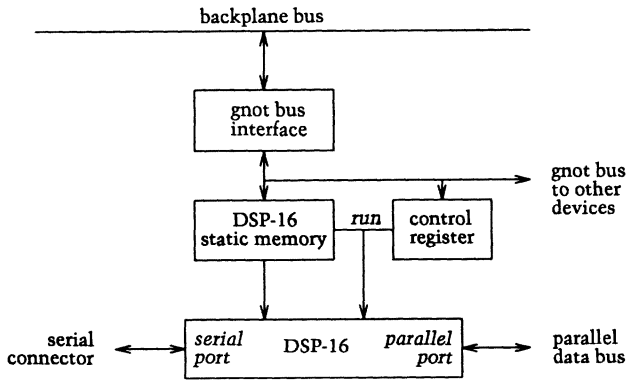
DI/DO	data in/data out
ICK/OCK	input/output clock
ILD/OLD	input/output load clock
IBF	input buffer full
OSE	output shift-register empty
SADD	serial address (for TDMA mode)
LR	left/right clock
BCK	bit clock (32 * 44.1 kHz)
YCK	Yamaha clock
SYNC	frame synchronization

with a 50 nsec cycle time), memory access must be shared between the 8-bit processor bus and DSP; when the processor sets the DSP to "run," it also prevents itself from accessing the memory.

The "final mix" card is last in the chain; it does the sample rate conversion from the Yamaha sample rate (58.8 kHz) to AES/EBU rate (44.1 kHz). The card also contains an AES/EBU converter attached to the serial port of the DSP-16.

The DSP-16 includes both a parallel and a serial I/O port. Each side of the serial I/O port has a shift register attached to the internal data bus of the DSP-16. The *input buffer full* (IBF) and *output shift register empty* (OSE) bits are available to external interfaces. The chip also provides load clocks for the shift registers and frame synchronization. The DSP-16 can be either a master (called "active" in DSP-16 terminology) or a slave ("passive"). We put all the DSP-16s except the final mix DSP in passive mode. They are fed clocks from the final mix DSP. This guarantees that all DSPs use the same output clock. The final mix card generates the left/right clock so that the channels are synchronized as well; that is, all core DSPs start out on the same foot. The serial data output of the DSP-16 is fed to the serial data input of the next DSP-16 in line. All of the serial I/O is done via flat ribbon cables on the end of the cards (we use a 26-conductor cable with alternating grounds). The signals transmitted over the cable are listed in Table 1.

Fig. 2. The core DSP architecture.



DI and DO are the serial data bits; ICK and OCK are the shift register bit clocks, and ILD and OLD are the shift register word clocks. IBF and OSE can be used for flow control. SADD is the serial processor address, and SYNC is the frame synchronizers; these signals are not currently used. LR distinguishes between the left and right channels. BCK is the bit clock for the serial interface, and YCK is the Yamaha serial bit clock. They are different because changing the Yamaha clock from its specification would result in lower pitch values, thereby reducing the useful range of the FM instruments.

The Final Mix Card

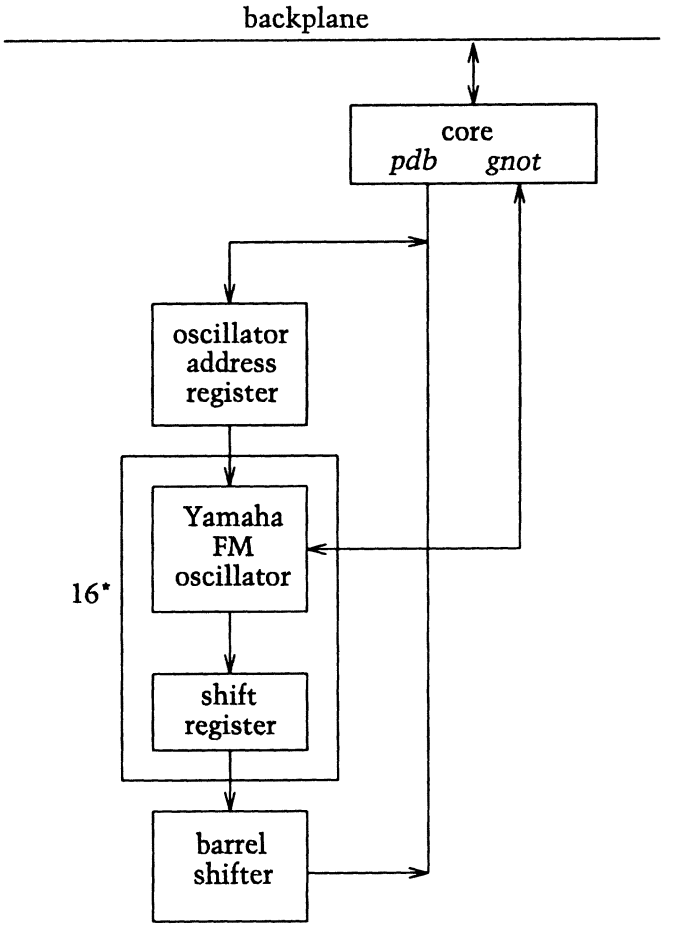
The final mix card has the basic ("core") DSP-16 circuitry found on all instrument cards. A block diagram of it can be found in Fig. 2.

The Gnot can write into the memory of the DSP-16, but only when the DSP-16 is stopped. The DSP-16 is too fast to allow true dual-port memory access. This is perfectly acceptable since the core program typically does not change when the synthesizer is running. The Gnot can also set a DSP-16 control register that contains the run flag and other useful bits.

The FM Array Card

Our FM instrument card uses 16 Yamaha YM-2151 oscillator integrated circuits. This chip is approximately the same as the chip used in the Yamaha FB-01 synthesizer. The YM-2151 is an eight-voice,

Fig. 3. FM array block diagram.



four-oscillator chip with on-chip envelope generators. The card is organized as shown in Fig. 3, the layout of the card is shown in Fig. 4a, and a photograph of it in Fig. 4b.

Note how the core DSP circuitry integrates into the card; the core DSP provides a mechanism to collect samples from all the FM chips as well as scaling and possible signal processing hacks inside the DSP. Since the output of the Yamaha chips is in 13-bit pseudo "floating-point" format (10 bits of mantissa, 3 bits of binary exponent), it must be converted to linear 16-bit format for use by the DSP-16. A barrel shifter accomplishes this, although it is "overkill" for the application. The output of the barrel shifter is put on the parallel input bus of the DSP-16. An auto-incremented address register is decoded and selects which FM chip is driving the shifter input bus.

Fig. 4. FM array circuit board layout (a) and photograph of the FM array circuit board (b).

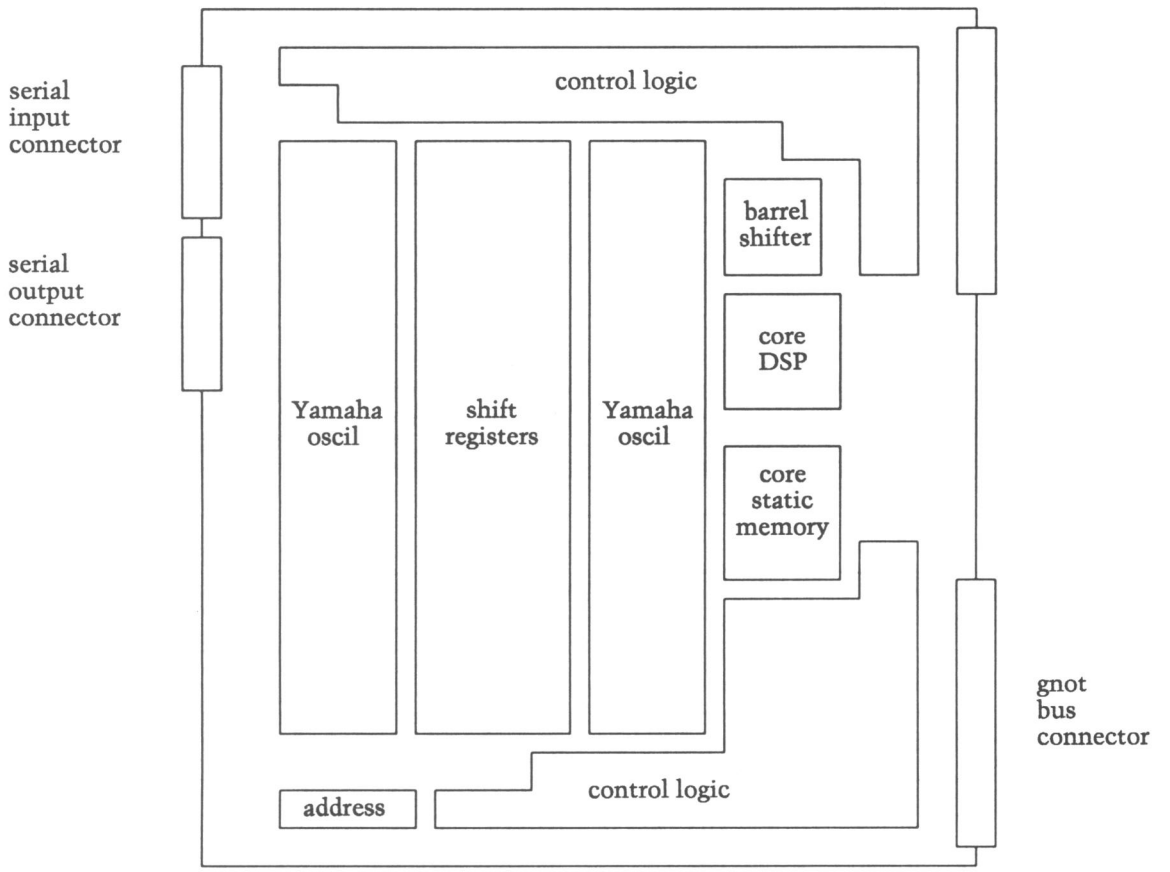


Fig. 4a

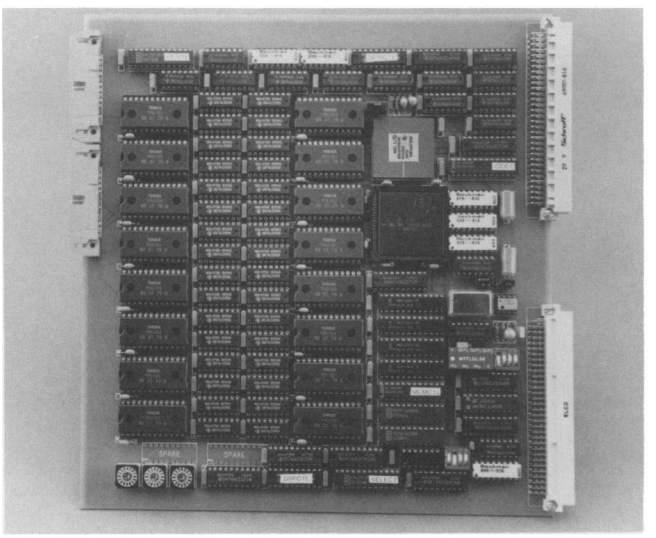
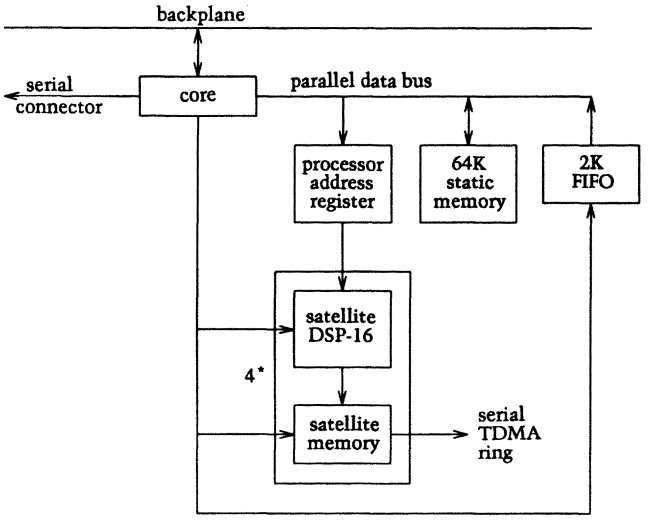


Fig. 4b

Meanwhile, the serial output of the previous card is fed to the serial input of the DSP-16, which is mixed with the sum of the barrel shifter outputs (computed by the DSP-16). The final result is put in the serial output register, which is then shifted out on the serial output pin, which goes to the next card in the chain.

The Gnot programs the Yamaha chips one at a time by setting a chip-select register and then writing data to the on-chip registers. The YM-2151 is fairly slow; a bus cycle takes about 500 nsec, and incurs 12 μsec of dead time (i.e., the chip is busy during this time). It takes 10 bytes to reset a channel (i.e., gracefully bring its output to zero), 38 bytes to configure a new instrument (voice), and an additional 22 bytes to sound a note, so the worst-case reload time is about 850 μsec. Data intended for different chips may be interleaved (at the expense of more writes to

Fig. 5. DSP-16 array block diagram.



the chip-select register), so most of the dead time can be recovered when things are busy. The equivalent FB-01 system exclusive MIDI message is 139 bytes long, requiring 44 msec for transmission (a ratio of about 50:1).

Unfortunately, the voices for the YM-2151 do not sound as good as the voices for the DX-7, the well-known Yamaha product with a six-oscillator FM algorithm, nor is there available anything comparable to the enormous library of DX-7 instruments. There are fewer oscillators per voice on the FB-01 (four oscillators per "algorithm" versus six), and their envelopes are more restricted. Of the 32 DX-7 algorithms, 25 can be simulated (ignoring the problem of converting the envelopes from DX-7 format to FB-01 format), using a set of YM-2151 channels. It is interesting that the remaining seven algorithms (for aficionados, numbers 4, 6, 12, 13, 16, 17, and 18) are commonly used in our favorite DX-7 voices. In spite of all this, the YM-2151 provided a reasonable way to get a fair number of instruments in a short period of time.

Note that each Yamaha chip has 32 oscillators (16 FM oscillators counting 2 oscillators per FM pair); therefore each board provides 512 total oscillators. By comparison, the 4B (Alles and Giugno 1977) developed at IRCAM provides a total of 64 oscillators per card using totally medium-scale integration logic.

DSP-16 Array Card

The DSP-16 array card has four DSP-16s and a core DSP. A block diagram of the card is shown in Fig. 5. The layout of the card is shown in Fig. 6a, and a photograph of it in Fig. 6b.

The core DSP addresses the four satellite DSPs via the 16-bit wide parallel I/O bus. The core can address any of the satellites and also a 64 kiloword external memory. This is specifically designed for reverberation (64k is about 1.5 sec at the 44.1 kHz sampling rate). The serial I/O ports of the satellites are connected in a ring using the on-chip logic of the DSP-16. The DSP- 16 multiprocessor interface permits up to eight processors to be connected on a serial bus. Slots must be reserved (i.e., statically allocated) beforehand, as there is no contention mechanism. Furthermore, each processor must have a unique address.

The Gnot has the same interface to the memory of the satellites as it does to the core. It also has a 2 kbyte FIFO buffer attached to the parallel I/O bus of the core for use in parameter-passing from the host. Status bits from the FIFO buffer can be used to interrupt the core DSP should the FIFO become too full and risk data overrun.

The slave DSPs are for general algorithmic synthesis and DSP algorithms (including pitch detection and various filtering algorithms). They are fast enough to compete with specialized VLSI solutions for additive and FM methods.

SMPTE/MIDI Card

The SMPTE/MIDI card does not have and does not need a core DSP section. It provides a simple interface to the SMPTE time code (via the Otari I0055 chip) so that the synthesizer can be set to operate as a slave to a videotape machine. The MIDI channel is formed around the Yamaha YM-3802 MIDI Controller chip. While providing almost too much of everything, it does provide a way to accept keyboard input from a keyboard controller such as a Yamaha KX-88 (or even a DX-7). A block diagram of this simple card is shown in Fig. 7; its layout is shown in Fig. 8a, and a photograph of it in Fig. 8b.

Fig. 6. DSP-16 array circuit board layout (a) and photograph of the DSP-16 array circuit board (b).

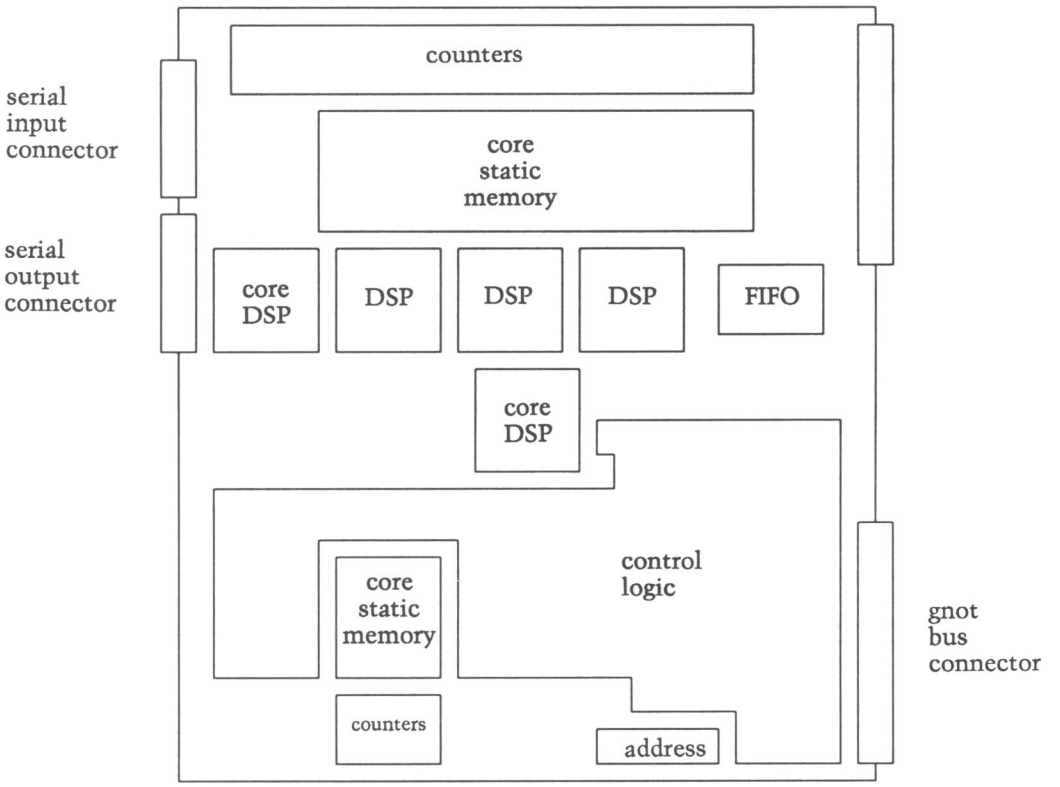


Fig. 6a

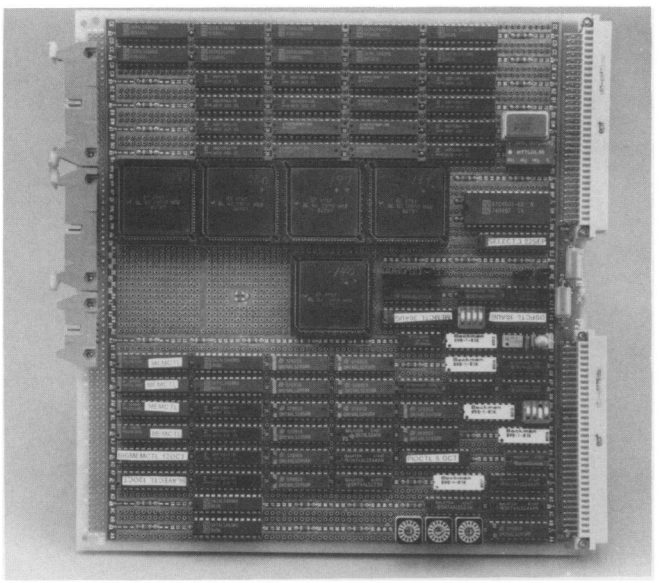


Fig. 6b

Fig. 7. SMPTE/MIDI interface block diagram.

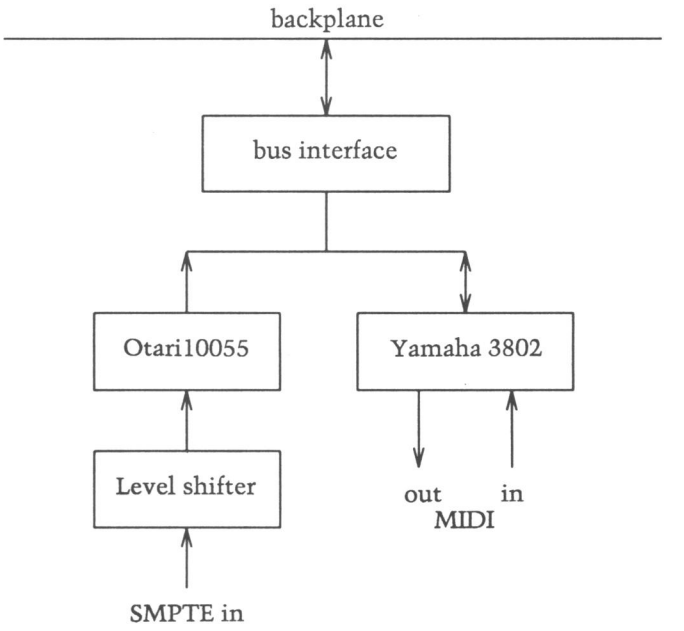


Fig. 8. SMPTE/MIDI interface circuit board layout (a) and photograph of the SMPTE/MIDI interface circuit board (b).

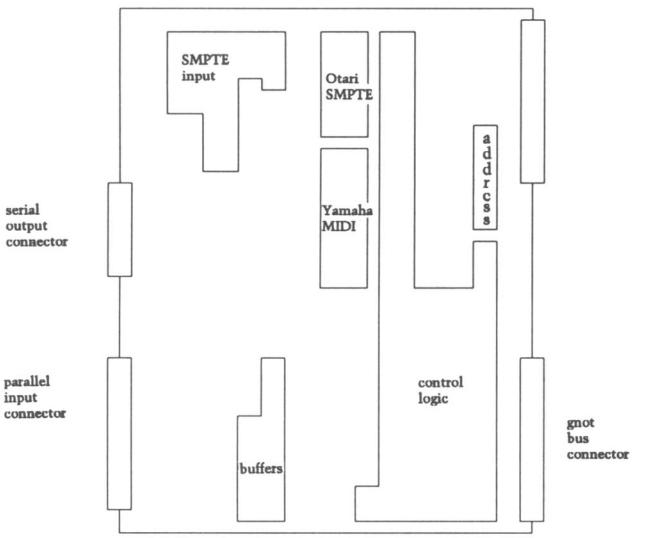


Fig. 8a

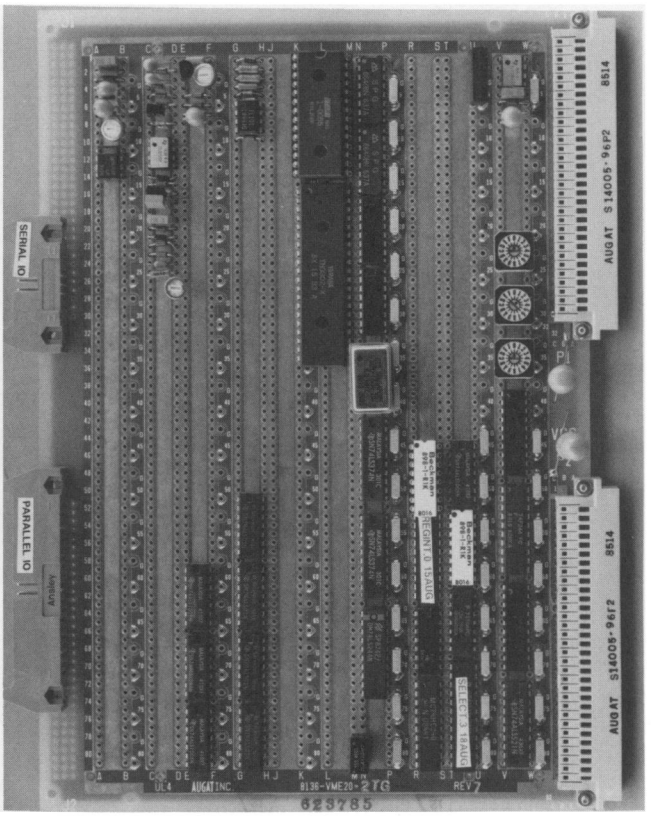


Fig. 8b

AES Converter

The final mix card cohabitates with an AES/EBU receiver and transmitter. This uses the (in)famous Sony CX-23053 and CX-23033 chips. The Yamaha sample rate (58.8 kHz) is converted to 44.1 kHz by interpolation and decimation by a ratio of 3:4. The output of the core is connected (via the standard serial flat ribbon cable) to the AES transmitter section. Likewise, the AES receiver section can be connected to the serial input of the core DSP of the final mix card.

Software Details

The Plan 9 system is organized around distributed file systems and CPU servers (typically multiprocessors). Computing is not done on local workstations (such as the Gnot) in this network; instead is it performed on fast, hot, and noisy machines located in air-conditioned rooms. Likewise, the file servers are not located adjacent to the workstation; they also are placed in climate-controlled environments. The file and CPU servers are interconnected via a variety of high-speed networks; the terminal workstations are connected via a medium-speed network (throughput is about 120 kbytes/sec).

The Low-Level Interface

The different DSP cards on the Gnot bus are programmed through a uniform interface. The memory of each core DSP looks like the UNIX system's memory device */dev/mem*, that is, it can be addressed, read, and written via system calls. The Yamaha card interface looks like a continuous string of oscillators; the oscillators are programmed by writing a [address, value] pair to the internal registers

of the YM-2151 chip. Above the devices (in the software sense) is a real-time scheduler that uses Plan 9 streams (Presotto 1990). Streams are a way of inserting coroutines (called "line disciplines") between low-level drivers and high-level code. The real-time line discipline is responsible for picking the next time-tagged event out of the input queue and executing it (i.e., reading or writing the appropriate register). The effect of the real-time scheduler is invisible to the higher-level users, as it should be.

The programs for the DSP-16 are written in assembly language and cross-compiled on a CPU server. The peculiar nature of the DSP-16 register set makes it very difficult to write a code generator for a higher-level language such as C. The resulting object files can be downloaded from the Gnot via the mechanism described above.

The High-Level Interface

Our first cut at a "music" interface is a MIDI file interpreter that builds on an existing score compiler (Killian 1986; Kahrs, Killian, and Mathews 1986) called *m*; which *m* was named in 1985, long before *M* (Zicarelli 1987) appeared. Compiler *m* accepts an ASCII representation of common music notation and generates a stream of MIDI events; these events are interpreted and translated by the *play* program. Notes are assigned to one of the 128 channels on an oscillator card in a strict least-recently-used fashion. Thus, each note-on event potentially causes a full voice load, but we can guarantee full sonority to a timbre with a long decay time because the large number of available oscillators makes it unlikely the oscillator will have to be used again for a long time.

Conclusions

Gnusic has been tested using a complicated movement from Stravinsky's *Le Sacre du Printemps*, but no contemporary use has been made to date.

The Gnusic organization offers potentially more oscillators than any other synthesizer available at present. However, the limited nature of the Yamaha YM-2151 chips precludes truly general use of the oscillators. One possible use is additive synthesis—using the "algorithm" that just adds together multiple oscillators, the outputs of the Yamaha oscillators can be added together (as usual) by the core DSP.

Eventually, the parameter update problem (Moorer 1981) will "bite" back at Gnusic too. As the number of active oscillators increases, the need to update parameters will increase as well. This problem will remain for the foreseeable future. Wider and faster process or buses will address this problem somewhat.

Although blazingly fast, the unusual nature of the DSP-16 register set makes coding an unpleasant task. One can also argue that floating-point numbers would be a better representation for use in signal processors. When Gnusic was designed, the DSP-32 was not fast enough to handle the mixing task. Any new version would use a fast floating-point chip (like the DSP-32C) with a C compiler.

Gnusic was created to implement various ideas in orchestral synthesis and signal processing that are difficult to try in MIDI systems. The coexistence of a powerful workstation and a flexible combination of versatile instrument I/O cards makes such experimentation possible.

References

Audio Engineering Society. 1985. "Serial Transmission Format for Linearly Represented Digital Audio Data." *Journal of the Audio Engineering Society* 33(10):976–984.

AT&T. 1987. WE DSP-16 *Digital Signal Processor Information Manual.* Murray Hill, New Jersey: AT&T.

Alles, H. G., and G. D. Giugno. 1977. "The 4B: A One-Card 64-Channel Digital Synthesizer." *Computer Music Journal* 1(4):7–9. Reprinted in C. Roads and J. Strawn, eds. 1985. *Foundations of Computer Music.* Cambridge, Massachusetts: MIT Press, pp. 250–256.

Chowning, J. M. 1973. "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation." *Journal of the Audio Engineering Society* 21(7):526–534. Reprinted in *Computer Music Journal* 1(2): 46–54, and in C. Roads and J. Strawn, eds. 1985. *Foundations of Computer Music.* Cambridge, Massachusetts: MIT Press, pp. 6–29.

Kahrs, M., T. J. Killian, and M. V. Mathews. 1986. "Computer Music Research at Bell Labs." In *Proceedings of the International Computer Music*

Conference. San Francisco: International Computer Music Association pp. 199–201.
Killian, T. J. 1986. "Computer Music under Unix Eighth Edition." In *Proceedings of the European Unix Systems User Group (EUUG) Spring Conference* Cambridge, UK: EUUG.
Loy, G. 1985. "Musicians Make a Standard: The MIDI Phenomenon." *Computer Music Journal* 9(4): 8–26. Reprinted in C. Roads, ed. 1989. *The Music Machine.* Cambridge, Massachusetts: MIT Press, pp. 181–198.
Moore, F. R. 1988. "The Dysfunctions of MIDI." *Computer Music Journal* 12(1):19–28.
Moorer, J. A. 1981. "Synthesizers I Have Known and Loved." *Computer Music Journal* 5(1): 4–12. Reprinted in C. Roads, ed. 1989. *The Music Machine.* Cambridge, Massachusetts: MIT Press, pp. 589–598
Pike, R., D. Presotto, K. Thompson, and H. Trickey. 1990. "Plan 9 from Bell Labs." In *Proceedings of the Summer 1990 UK UNIX User's Group Conference.* Cambridge, UK: EUUG. pp. 1–10.
Presotto, D. L. 1990. "Multiprocessor Streams for Plan 9." In *Proceedings of the Summer 1990 UK UNIX User's Group Conference.* Cambridge, UK: EUUG. pp.11–19.
Zicarelli, D. 1987. "M and Jam Factory." *Computer Music Journal* 11(4): 13–29.