# Proposal To Conduct Research in Computer Music

(Preliminary)

Experimental Music Studio
Massachusetts Institute of Technology

Center for Computer Research in Music and Acoustics
Stanford University

Computer Audio Research Laboratory
University of California, San Diego

26 May 1983

## 1. Overview

The creation of music is a complex task. Despite expensive equipment, current computer music systems provide inadequate facilities for the exploration of many aspects of musical interest. Based on our experience with music making and computer science, we propose to create a satisfactory computer system for music composition, performance, and research. Specifically, we propose to execute the necessary research in the areas of music computation and control structures which will allow the development of the target system over the next 5 years.

Our overall goal is to produce a system capable of performing well under concert conditions, able to produce sounds as rich and varied as those of traditional instrumental ensembles, and flexible enough to support a wide variety of musical styles and applications.

### 1.1. Computational Considerations

Music making often involves the time-sensitive control of many interacting parallel processes. In jazz improvisation, for example, players communicate rapidly on a number of musical levels. We can obtain an estimate of the computational complexity of this activity by considering the characteristics of the sound produced by typical instruments, the nature of the control exercised by the performers, and the contributions of such factors as room acoustics. The required computational bandwidth is impressive – on the order of 100 to 1000 times the capabilities of current computers for the sound synthesis alone. One estimate shows that a typical sound (that of an orchestra) created by additive synthesis requires a von Neumann computer capable of a memory access every 1/2 nanosecond, a multiply every 3 nanoseconds, and an add every nanosecond, while using 100 million bytes of random access memory.

This extreme computational load has led to the development of a number of efficient synthesis algorithms. The underlying synthesis technique is, however, only the first of many steps in the production of a musical sound. In a qualitative sense, most synthesis algorithms end up requiring about the same amount of computation to achieve acceptable results.

Some progress on the computational problem has been made by the construction of special hardware. Acceptable musical results have been obtained from the 4X machine at IRCAM and the System Concepts Digital Synthesizer at Stanford, among others. Even unsuccessful synthesizers cost a great deal, and the good ones are found in only a few research centers. Access to these devices is very limited. Besides the cost, the main drawbacks encountered with such devices have been in the areas of realtime control, flexibility of programming, and numerical precision.

Therefore, one of our main goals is to develope a flexible, powerful sound synthesis machine capable of operation in concert situations. Necessary research work remains in the areas of realtime input devices, synthesis and processing algorithms and devices, and system architectures.

### 1.2. Control Considerations

The creation of a satisfying sound is only the first step in making satisfying music. The creation of computer music is often a very laborious process culminating in the rather odd spectacle of an audience in a darkened room watching a tape recorder. Because musicians are constrained at this time to provide every detail of the music far in advance, there is little chance to adapt to the peculiarities of the concert hall or to the actions of live musicians. Without interactive controls, there is little chance to study the actual adjustments made by performers. And without high level, extensible representations of the music, composition moves very slowly.

The control needs of the composer and the performer are only one level of the control problem encountered in computer music. Rich musical sound is a very complex signal. The partials of a musical sound change in relative amplitudes during a note, drift to form glissandos, and fluctuate together in nontrivial ways to create vibrato. In addition, all the sounds, which must be precisely synchronized, cooperate to satisfy requirements of tuning and gesture. Any of these events can require millisecond timing. Large gestures (such as a sudden change in tempo) can require that a large number of complex tasks be handled in a very tightly constricted time.

Therefore, a second major goal of this research is to find representations of musical events which fit the needs of musicians, but which also can be applied to the solution of the lower level control problems. The overall goal here is a flexible musical instrument capable of sustaining the interest of diverse musicians.

### 1.3. Development Considerations

The development of the target system consists of the following three steps:

1) Create a computational structure powerful enough to produce in realtime musical sounds as complex as those of an instrumental ensemble.
2) Create a control structure flexible enough to support a wide range of musical applications, including performance, composition, and research.
3) Integrate the result into a form that can benefit maximally from the constantly changing technology on which it is based.

The third step can bear some emphasis. Not only are many of the questions being addressed here unanswered, but in addition, the underlying technological base is constantly changing. The research and development work should be independent of particular hardware insofar as possible, so that we can exploit technological advances. Custom VLSI chips, for example, may soon be able to handle much of the computational load.

The first step given above appears to be solvable by subdividing the computational load into several sound production engines operating in parallel, each engine supporting computation at about the level of a single traditional instrument. Each such engine should be able to communicate with all the other engines in a musically intelligent manner.

The solution to the second step is less clear at this time. Research is currently being directed towards identifying suitable control structures, from allocation schemes for synthesizer resources to high level languages needed by composers.

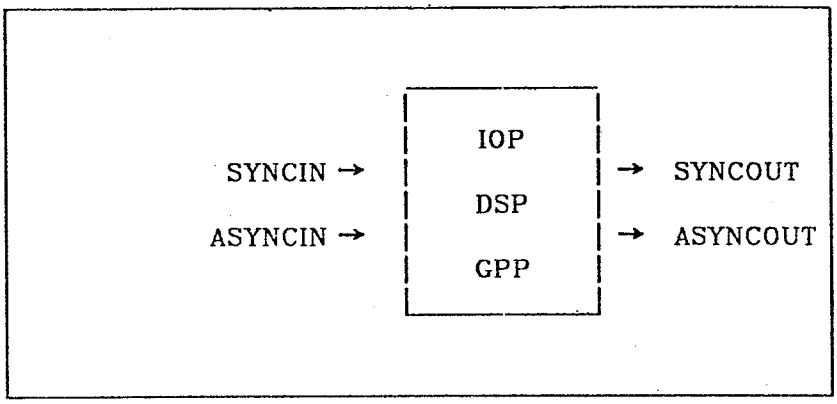
Before research can begin on the technical problems involved in these three steps, it is imperative that we acquire a sufficiently flexible set of research tools. The power of these research tools will be of crucial importance to every aspect of the project – from hardware and microcode design to final integration of the entire system.

In the end it is essential that the system be able to perform well under concert conditions, produce rich and satisfying musical sounds, and support a wide variety of musical interests. The next section provides a more detailed description of the main aspects of the system's functions.

## 2. Target System Description

In this section we describe the function of each portion of the target system and discuss some of the problems we have identified so far. We expect implementations of these functions to proceed in a fluid way, taking rapid advantage of improvements in hardware and software technology.

In the following specification we adopt the notion of a system element (SE). An SE reads and writes two kinds of data: synchronous multichannel digital audio data (SYNC), and asynchronous control data (ASYNC). The SE itself performs both standard digital signal processing (DSP) and general purpose processing (GPP), as well as coordinating the input and output of the data (IOP).




Most of the discussions that follow concentrate on the processing power and flexibility of the functions of the SE. In general each SE should have enough power to handle the computational load and control problems described in Section 1.

### 2.1. Hardware Description

The target system hardware should be modular, easily reconfigured, and sufficiently powerful to produce good music. Because SYNC data rates are so high, special interconnection strategies may be required. For higher level control, however, we expect to use standard hardware interconnections such as Ethernet and Multibus. The wide use of these standards is one of their greatest assets. Because many devices have been developed to use them, many of the interconnection problems have already been solved, and further developments will remain compatible with the rest of the system.

Currently, the favored target system processor is some form of the Motorola 68000 packaged as a "workstation". Many vendors provide relatively inexpensive versions of these computers, and many peripherals are being developed.

The signal processing and synthesis power, however, will have to come from specially developed or adapted hardware added to the workstation. A major goal of this project is to produce an inexpensive, but powerful, music workstation. Because digital technology is now changing very rapidly, however, it is premature to settle on the final hardware configuration at this time. In the following sections we summarize our current view of the functions the hardware should support.

### 2.2. Signal Processing

Signal processing is one of the basic functions of the target system. We expect, for example, to be able to perform waveform encoding and modification, spectrum analysis, simulation of room acoustics, intelligent music editing, and experiments in psychoacoustics. Commercially available boards already exist that provide some of these capabilities, and others may be designed during this project. If the special purpose hardware remains compatible with a common library of software functions, reconfiguration becomes simpler,

and less expensive versions of the target system can be envisioned. Much of this signal processing software already exists.

### 2.3. Synthesis Algorithms

A second major function of the target system is waveform synthesis and instrument simulation. Current synthesis techniques include additive and subtractive synthesis, frequency modulation, and waveshaping, among others. We expect to continue research into new algorithms to exploit existing hardware and to generate new ideas for future hardware. The target system should also support synthesis technique development. Because musical instrument design is often tied closely to the peculiarities of the underlying hardware, the set of synthesis algorithms available to the target system can affect the design of many higher level portions of the system.

It should be noted that instrument design and simulation is far more than just settling on a particular synthesis algorithm. In most advanced instruments, a noticeable portion of the computation involves general purpose programming for vibrato, envelope, tuning, and decisions about synthesis parameters based on the state of other instruments running at the same time.

### 2.4. VLSI Possibilities

The synthesis and signal processing power mentioned in the previous sections may come, at least in part, from custom VLSI chips. Digital oscillators, for example, are sufficiently well understood and heavily used that they may be best provided by VLSI. Similarly, digital filters, delay lines, envelope generators, and other standard "unit generators" map well into VLSI. Development of a suitable chip or set of chips may reduce the total chip count and interconnection complexity to a point where the entire music processor can reside in the workstation. To use VLSI technology appropriately, we must have a good characterization of the needs and architecture of the processing we intend to do. This specification can then lead to the design of new chips or to the exploitation of existing chips. The current exploratory work with Carver Mead is an initial step in this direction.

### 2.5. Board Design

Hardware design will be required by the work on realtime control devices and implementations of synthesis and analysis algorithms. In this regard, board design should be done in the context of the entire system with careful thought given to the interfaces between various portions of the system. Since the exact nature of these boards is one of the subjects of this research, an efficient and flexible prototyping environment is essential. Computer aided design tools are necessary for the development and evaluation of alternative hardware designs. In the target system, the special boards will provide extra processing power and I/O bandwidth, as well as interfaces to the realtime control inputs that we expect to develop.

As new devices are developed, microcode must be written to bring the devices into fruition. Presumably, the microcode compiler on the target system will be used only by the more sophisticated users. During the development stage, however, the compiler will be in constant use. It should therefore be as efficient and pleasant to use as possible.

It is conceivable that the performance control devices will require that microcode be compiled and loaded in realtime. This possibility adds to the importance of developing a flexible, reliable, and efficient compiler.

### 2.6. DAC/ADC

The digital/analog conversion system sets an upper limit on the quality of the sound produced by the workstation. At the minimum, the DAC/ADC should provide four channels of 16 bit linear conversion capable of running at sampling rates close to 50 kHz per channel. In addition, the DAC/ADC should provide sampling rate and data format conversion to reduce the load on the SEs. Ideally we would like to provide 20 or more bits of dynamic range – an orchestra, for example, produces a dynamic range that requires at least 20 bits.

### 2.7. Storage

Large amounts of storage will be needed both for the musical data (sound samples in particular) and for the programs that operate on these data. Conventional computer disc systems should be adequate for programs and parametric data. To store the sound data, we can take advantage of the new recording technologies being developed by companies such as Sony. By combining access to a standard disc with a digital recorder such as the Sony PCM-F1, we can provide random access to extremely large amounts of data. In any case, the target system should be easily interfaced to these new recorders. It should be noted that these Sony recorders have excellent DACs and ADCs.

### 2.8. Output Display

The video display of the target system must have high enough resolution to permit work in common music notation graphics, as well as supporting such obvious tasks as editing text and waveform displays. Since analysis output and control data may be used under performance conditions, the display include some realtime capabilities. High speed will also make interactive jobs like editing much more pleasant.

### 2.9. Control and Scheduling

The process control and scheduling functions are closely tied to the behavior of the target performance instrument. In general they provide a knowledgeable realtime process which implements the interpreted intent of a musical voice, and a control mechanism to handle the collection of such processes by performing musically intelligent arbitration between scheduling and resource conflicts.

At the hardware level, the control devices consist of objects such as knobs, joysticks, and keyboards for direct manual intervention, realtime analysis tied to musically intelligent programs for indirect, semi-automatic control (so that voices may play "in tune" for example), and various forms of indirect control based on cues from a conductor or gestural data present in a score.

Converting the data produced by these controls requires considerable signal processing coupled with reasonable and programmable responses to the results of that analysis and processing. The processing includes switch debouncing, downsampling, thresholding, pattern matching, and timbre analysis. The second stage (the response), requires programs for performance interpretation based on musical knowledge, realtime cues, and artistic whim.

The operating system, which must coordinate these activities, should provide facilities for interrupts with several priority levels, multiple pre-emptable processes, explicit control of swapping mechanisms and garbage collection, and various features to aid in handling cases of overload where degradation of some event is unavoidable.

Providing flexible and powerful controls of the resources needed to make music is a problem that pervades nearly all aspects of the system. Much of the discussion in the sections that follow about the target system software centers on exactly this issue.

### 2.10. Software Description

The software of the target system comprises the compositional and performance environment. The main issues to be addressed here are the following: an extensible representation for musical data must be developed, and a means is needed to capture, study, and manipulate the musical knowledge currently hidden under the term "performance practice". In addition, a friendly, powerful music making environment must be built in which the musician can have access to all the power of the computer system. The latter includes, of course, access to the editors, compilers, debuggers, and so on that make up a normal program development system.

### 2.11. The Compositional Environment

Our experience with composition languages indicates that composers make heavy use of every aspect of advanced program development facilities. These include the entire gamut of language support functions from abstract data types to interactive debugging. Currently there is no widely accepted computer representation for musical data. There is a clear need for user-definable representations coupled with a means to keep these representations consistent. In addition, we need to provide a way to embed musical knowledge into the objects handled by the composer. The extensible language should be interpretive to allow maximum flexibility, realtime potential, and high level debugging. To recapture efficiency, user-defined tools should be compilable into optimized low level code. Such compilation should be able to integrate effectively all available special purpose hardware in a variety of configurations. Research into this area is fundamental to computer music because it addresses the problem of molding computational resources into musical tools.

Given a satisfactory representation of musical data, the target system must provide a programmable, graphics oriented editor/compiler for this data. Common music notation graphics should be available, as well as high level programming language constructs for describing musical behavior. The composer should be able to build his own personalized environment using structures that reflect his peculiar compositional needs.

### 2.12. The Performance Environment

Beyond the performer's greater need for reliability and fast response, there is little qualitative difference between the composer's environment and that of the performer. The control console of the target system must be easily modifiable to accommodate a wide variety of control devices. The performer should be able to interact with the control devices and the ongoing music with high level constructs, and see quickly updated displays of the current state of the world.

In addition to direct controls such as clavier keyboards, knobs, and switches, input channel processing can capitalize on the computer's programmability, its analysis capabilities, and its ability to mimic certain human actions. The realtime synthesis and signal capturing capabilities of the target system should make it capable of taking full advantage of the gestural sophistication of a virtuoso performer. We expect to use the target system for researching skilled music communication through the study of performers coupled with programs for knowledge-based interpretation and performance control.

### 2.13. System Integration

The proposed system involves the coordination and integration of many diverse hardware and software components. Modularity at every level is of paramount importance. All communication should be among virtual machines, microcode should be compiled from high level descriptions of function, and access to global information should be kept to a minimum. Once the target system has become better defined, optimizations may be used to recover some of the efficiency lost through modularity.

During system integration, powerful debugging tools become extremely valuable. Good modern debuggers provide such facilities as fault context (including backtraces), decompilation with pointers to source code, dynamic displays of selected variables, and various operations like single stepping, stopping, breakpoints, and selective modifications.

It is essential that the prototyping environments at all research sites be fully compatible. We expect to do most of the programming in C and common Lisp, and expect the operating system used to be a modified version of Unix.

### 2.14. The Development System

Given the fast pace of technological change, the experimental nature of the initial design work, and the number of complete unknowns still to be dealt with, it is premature to make decisions about the exact nature of the target system. The development system, however, can be specified more fully because we know now what general tools are needed to make further progress towards the creation of the target system.

The development system need bear little resemblance to the target system, since we expect to be using portable languages for software development in any case, and the special purpose hardware will almost certainly go through many iterations before it settles into its final form. It is more appropriate at this stage to define a development system that provides the support needed to attain that target.

The nature of this research dictates that the development system support C, Lisp and UNIX software portability, especially towards 68000 based workstations, that it be based on an architecture giving large memory access and substantial computing power, that it contain sophisticated program development tools such as debuggers and incremental compilers, that it support the networks and busses we envision using (Ethernet and Multibus), and that it be supported by sufficient technical resources so that problems with the development system itself do not become insurmountable. It should also be available quickly so that we do not become bogged down developing our development tools.

The hardware development phase will require access to computer aided design tools, microcode compilers, and easily interchangeable modules, in addition to normal tools such as logic analyzers. If we intend to do chip design, we will need color displays as well.

## 3. Research and Development Plans

We have identified several problems which must be solved before the target system can be built. Some of these problems, such as specifying the signal processor, conversion systems, and interactive controls, largely involve the transfer of knowledge from one mileau into another. This is basically development opposed to research – work.

Other problems, such as how to control the parallel signal processing structure, need further research before detailed specifications can be given. We proceed, therefore, by creating a prototyping environment for hardware and software in which the needed research and development work can be integrated.

The main things to be done immediately are:

1) **Establish efficient communications among all sites.**
This is largely accomplished. Our computer mail connections are nearly reliable enough to begin to substantially augment telephone and travel.

2) **Select and purchase target prototypes for all sites.**
There is current agreement that the target system probably will be some descendant of a 68000 workstation running UNIX, augmented by special interactive controls and realtime music processing hardware. Until the special hardware can be built and tested, however, the most economical way to augment a current workstation for music is with an array processor (AP). The MIT group will spearhead a project (see Attachment 1) based on 68000-based workstations and AP500s. The Stanford group is investigating Charles River 68000 systems and Marinco APs. The UCSD group is a beta test site for 4.2bsd UNIX using the SMI/SUN workstation. These initial explorations will aid enormously in developing the standard prototype system for this project. We expect the target system prototype technology to change rapidly, hence to purchase improved components regularly.

3) **Select and purchase common, compatible development systems for all sites.**
The development system consists of the hardware and software prototyping environments. The hardware prototyping environment is basically a standardized card cage with separable buss interface, plus normal digital engineering tools, such as logic analyzers. The software prototyping environment requires advanced development stations compatible with both the target system and design/simulation tasks. The better the research tools, the further the target system can be developed in the time available. It is critical that the hardware and software prototyping environments be standardized as rapidly as possible. This will allow both researchers and their results to move freely among the research sites. After lengthy debate, we are prepared to select the best available tools which meet our requirements.

4) **Commission outboard rate changing converters (subcontract).**
A critical component of all systems is audio converters which meet the specifications stated previously. In addition to these specifications, the converter buss interface must be separable from the rest of the system, since we expect the buss and audio

conversion specifications to change asynchronously.

5) **Representatives from each site communicate in subgroups.**
Each site will have a contact person designated in each of the following main areas:

System Architecture
*control, scheduling, allocation, system integration, etc.;*
Signal Processing
*algorithms, synthesis, processing, analysis, computational strategies, etc.;*
Hardware Standards
*busses, processors, interconnection standards, fabrication, etc.;*
Software Standards
*operating systems, compilers, virtual machines, debugging aids, etc.;*
Related Research
*psychoacoustics, cognitive science, music theory, etc.;*

6) **Paying for the research.**
The money needed for this project pays for researcher salaries, development systems, target system prototypes, subcontracts, and some operating expenses. Some reserve is requested to fund related research projects.

BUDGET SUMMARY
(See Attachments)

| | MIT | Stanford | UCSD | Total |
|---|---|---|---|---|
| YEAR 1: | | | | |
| FTE | 218 | 250 | 100 | |
| HARDWARE | 379 | 330 | 320 | |
| TRAVEL | 20 | | | |
| SUPPLIES | 80 | 80 | 80 | |
| SUBCONTRACTS | | | 100 | |
| YEAR 1 TOTAL: | 697 | 660 | 600 | 1957 |
| YEAR 2: | | | | |
| FTE | 284 | 250 | 100 | |
| TRAVEL | 20 | | | |
| SUPPLIES | 80 | 80 | 80 | |
| SUBCONTRACTS | | | 100 | |
| YEAR 2 TOTAL: | 384 | 330 | 280 | 994 |
| YEAR 3: | | | | |
| FTE | 301 | 250 | 100 | |
| TRAVEL | 20 | | | |
| SUPPLIES | 80 | 80 | 80 | |
| SUBCONTRACTS | | | 100 | |
| YEAR 3 TOTAL: | 401 | 330 | 280 | 1011 |
| YEAR 4: | | | | |
| FTE | 319 | 250 | 100 | |
| TRAVEL | 20 | | | |
| SUPPLIES | 80 | 80 | 80 | |
| SUBCONTRACTS | | | 100 | |
| YEAR 4 TOTAL: | 419 | 330 | 280 | 1029 |
| YEAR 5: | | | | |
| FTE | 338 | 250 | 100 | |
| TRAVEL | 20 | | | |
| SUPPLIES | 80 | 80 | 80 | |
| SUBCONTRACTS | | | 100 | |
| YEAR 5 TOTAL: | 438 | 330 | 280 | 1048 |
| | 2339 | 1980 | 1720 | 6039 |
| OTHER: | | | | |
| Northwestern (Psychoacoustics Research) | | | | 183 |
| Undesignated Research Support Reserve (50k/yr) | | | | 250 |
| TOTAL: | | | | 6472 |

**ATTACHMENT 1: MIT Contributions to Workstation Development**

The MIT Experimental Studio has focussed its past research on the human/machine interface, real-time control of digital synthesis, and perceptually based signal processing. The following notes refer to line items in the MIT Project Schedule.

**Extensible Score Representation**

The rubric *extensible scores* denotes a convention for score representation, adhered to by all editors and performance software, which supports sophisticated formalisms of musical understanding. Without the ability to structure and process information in the musical domain as well as the acoustical, the target synthesizer will be unlikely to achieve the goal of fluent, idiomatic communication with musicians. Work on data conventions to support extensibility can begin right away. Ensuing work will be driven by the needs of two types of real musical applications: score preparation, and rehearsal.

Score preparation involves the non-real-time editing of basic score information in modalities most natural to the data. The eventual user of the COMITS machine might reasonably expect to communicate in standard musical notation, and we expect to complete a first-round implementation of a Common Practice Notation (CPN) editor in 1984-5, requiring about a man year. Other editing modalities we intend to provide will facilitate hierarchical approaches to categories of musical structure (phrases, pitch, timbre). The objective here is not to develop automatic composing programs, but rather to implement a system for the description and cognitive verification of the composer's "heard" musical structure. This research will need to include the integration of active objects and message passing protocols, which we see as a natural and necessary idiom for computer scores.

The score of a composition must also combine with information developed through rehearsals to create a performance. Human performers do not limit themselves to the contents of a printed score but maintain additional information through memory of rehearsals and by reference to systems of stylistic convention. Rehearsal can be seen as a special kind of editing, but one in which the modality of interaction is so different as to warrant special consideration. A prototype non-real-time "rehearsal" editor will be explored in 1983-4 simultaneous with the initial development of extensible representation. This will investigate (1) strategies for maintaining information about multiple rehearsals, and (2) strategies for understanding synthesis algorithms so that mapping of score into control parameters can be accomplished. As real-time synthesis capabilities are developed within the COMITS project, this work will be extended to real-time rehearsal.

**Music500**

Music500 will be a real-time synthesis system powerful enough for stage performance but very easy to interface to user-written software under UNIX. This will allow development of the performance interface for the target system to proceed without waiting for the target system's synthesizer. It will also allow us to find protocols for control of digital synthesis free of timing inaccuracies and excessive bandwidth requirements.

*Version 1* of music500 will be running at MIT in early August and will be ready for export to the other two sites by November 1, 1983. This initial release will also include special control process implementations, not detailed here but stemming from Max Mathews's residency at MIT during the summer months of 1983.

The *pipeline compiler* will speed up nonrepetitive user code in Music500 by transforming linear sequences of floating-point arithmetic operations into partially optimized AP500 pipeline microcode. This will make Music500 more efficient and allow general number-crunching for real-time mapping of control. The experience gained by writing this compiler will apply to the microcode compiler for the target system

synthesizer.

Real-time *interconnection* of array processors will allow coordinated performance using two or more synthesizers. This will be accomplished by reducing the necessary bandwidth between synthesizers, and by the development of exchange protocols robust enough to allow nondeterministic timing of communication. This communication will transpire between control processes on each machine, and will have a low enough bandwidth for a Multibus or even RS232 implementation. Its hardware independence will make it directly applicable to the final system.

A parallel study will investigate those cases requiring such high bandwidth or low latency that the communication must take place directly between number-crunching elements. This may require building a special high-speed synchronous bus; alternatively we might be able to make use of a prototype backplane in direct anticipation of the target machine.

**Control Structures for Computer Music**

This research will explore device-independent strategies for computer music synthesis and control. Its motivation lies in the wide separation currently found between real-time and non-real-time music systems, particularly in their representation of timing and control. Musically mature synthesis systems found in the most active production centers today exhibit highly idiosyncratic control protocols that actually impede the exchange of ideas and of useful data. This research will identify a unifying framework for the control-signal and audio-signal structures of real-time and non-real-time synthesis, allowing them to find common representations. It will also provide a test-bed for new concepts in music software systems using the latest programming methodologies and abstractions. The combination of new software techniques and orderly synthesis control will provide a suitable environment for the development of real-time knowledge-based interpretation and performance schemes. Applications will range from music500 to the target system.

**Real-time Mapping of Control Inputs**

The raw input data available to real-time control programs typically contains much irrelevant information, and requires intelligent preprocessing to reduce the bandwidth to amounts usable for real-time control. We plan to develop specialized "mapping algorithms" to transform input data of various kinds into forms more directly useful to control programs.

*Acoustic mapping* will permit selection of specific sound attributes for control programs. This will include programs for "listening", based on simplified models of auditory perception, for the detection of pitch, timbre, and note onset times. Alternative extraction models will be constructed and tested for their usefulness in live instrument control of synthetic sound. Internal feedback will enable the synthesizer to monitor the perceptual effects of its own parameter changes; external audio feedback will make it sensitive to ambient conditions of the performance space.

*Gestural mapping* will exemplify the kind of research Music500 is designed to support. Gestural mapping algorithms will find "best fits" of observed inputs to a lexicon of stored ones. This stored lexicon will be editable by the user by "rehearsing" the mapping software. A library of standard data reduction routines (downsampling, thresholding) will also be made available. Applications will be aimed at knowledge-based control programs, running initially on the AP500 but aimed eventually at the target synthesizer.

### Signal Processing Algorithms

Our recent development of new signal processing algorithms for artificial reverberation and perceptually-based processing of sound has benefited from collaboration with the MIT Digital Signal Processing Group. We next plan to facilitate further discovery of musically applicable algorithms by integrating classical signal processing procedures more deeply into a flexible music processing and control environment. A subset of these algorithms will be transferred to the AP500 to test their feasibility in real-time operation. This will anticipate the transfer of a similar subset to the target machine.

A parallel endeavor will be to explore a new class of signal processing algorithms that would exploit the special structure of the VLSI initiatives. We sense that speedy utilization of this architecture for powerful signal processing will be enhanced by the close relationship we presently enjoy with the MIT Digital Signal Processing Group. We expect to be able to supplement the classical repertoire of signal processing with an array of innovative procedures at about the time we begin moving our own research and production load onto the target machine.

### Target System

*Synthesizer design and simulation* will be aimed toward verifying and affecting the synthesizer chip design and hookup. This includes design work on multiprocessor connection both within a chip and between chips. Implementations of unit generators and real-time synthesis control down to the switch level will be conceived and simulated. This work is also concerned with specifications of a bus for sound transfer between several synthesizer cards, and for the level of intelligence required on the board.

Development of the *synthesizer microcode compiler* will draw on the experience of having developed the pipeline compiler for the AP500. The final result of this work will be an orchestra compiler for the resulting synthesis chip. It will emphasize flexibility and interconnectability of modules at all levels. At the signal-processing level it will support arbitrary connections of "unit generators" and "instruments", forming a hierarchy of active objects. At top level, the entire synthesis network will adhere to message-passing protocols emerging from research in control structures.

The *high-bandwidth sound/control bus* will connect synthesizers in the target system. It will feature a low latency of transfer, allowing signal-processing networks to be split between devices on the bus. This will allow easy expandability of the target system with little necessity for the user to worry about machine boundaries.

*Exercising and debugging* the final system will be partly achieved by its use in production. Intensive exercising by musicians will induce a stream of valuable comments and suggestions for the researchers to act on. The outcome will be a robust system with clear documentation and well-tested software.

Project Schedule, MIT subgroup

| calendar | '83 | '84 | '85 | '86 | '87 |
|---|---|---|---|---|---|
| Extensible Scores: | | | | | |
| Rehearsal input & interpretation | .5 | 1 | 1 | 1 | 1 |
| Composing langs & representation | .5 | 1 | 1 | 1 | 1 |
| Music500: | | | | | |
| Version 1 export | 2 | | | | |
| Pipeline compiler | | 1 | | | |
| Interconnection | | .5 | 1 | | |
| "Control Process" Music: | | | | | |
| Development: | | | | | |
| Mus500/cpmusic integration | 2 | 1.5 | 1 | 1 | 1 |
| | | .5 | | | |
| Real-time mapping of control: | | | | | |
| Acoustic | 1 | 1 | .5 | | .5 |
| Gestural | | .5 | .5 | .5 | |
| Signal processing algorithms | .5 | .1 | 1 | 1 | 1 |
| Target system: | | | | | |
| Synthesizer design & simulation | | | 1 | 1.5 | |
| Synthesizer microcode compiler | | | 1 | 1 | 1 |
| High-bandwidth control/sound bus | | | | 1 | |
| Exercise and debug final machine | | | | | 2.5 |
| Group Totals: | 6.5 | 8 | 8 | 8 | 8 |

Year 1 MIT budget

| | | |
|---|---|---|
| PERSONNEL: | | |
| 6.5 FTE research staff @ $33,475 (see note 1) | | 218K |
| HARDWARE: | | |
| Software Development Processors | | |
| 2 Analogic AP500 Array processors with 256KB 68000 program memory, 568KB data memory, Multibus and aux port interfaces @ 35000 | 200K | |
| 2 fully configured 68000 workstations @ 30000 | 70K | |
| (including Marinco APs) | 60K | |
| Special Purpose I/O | | |
| Polhemus position-and-attitude sensor | 19K | |
| Snell Front End Processor | 25K | |
| Ethernet hardware and interfaces for 11/750, 11/50 | 5K | |
| TRAVEL: | | 379K |
| SUPPLIES (INCLUDING HARDWARE UPDATES) | | 20K |
| Total first year | | 80K |
| | | 697K |

Year 2.

| | |
|---|---|
| 8 FTE research staff @ $35,484 | |
| SUPPLIES (INCLUDING HARDWARE UPDATES) | $284K |
| TRAVEL | 80K |
| Total year 2 | 20K |
| | 384K |

Year 3.

| | |
|---|---|
| 8 FTE research staff @ $37,613 | |
| SUPPLIES (INCLUDING HARDWARE UPDATES) | $301K |
| TRAVEL | 80K |
| Total year 3. | 20K |
| | 401K |

Year 4.

| | |
|---|---|
| 8 FTE research staff @ $39,869 | |
| SUPPLIES (INCLUDING HARDWARE UPDATES) | $319K |
| TRAVEL | 80K |
| Total year 4 | 20K |
| | 419K |

Final year.

| | |
|---|---|
| 8 FTE research staff @ $42,261 | |
| SUPPLIES (INCLUDING HARDWARE UPDATES) | $338K |
| TRAVEL | 80K |
| Total year 5 | 20K |
| | 438K |
| MIT total cost (see note 2.) | 2339K |

*Notes on MIT budget.*

1) A FTE costs the following in year 1:

| | |
|---|---|
| salary | 25000 |
| benefits @ 33.9% | 8475 |
| | 33,475 |

Adding 6% inflation gives a per-fte-cost of

| year 1 | year 2 | year 3 | year 4 | year 5 |
|---|---|---|---|---|
| $33,475 | 35,484 | 37,613 | 39,869 | 42,261 |

2) MIT overhead is not included.

## ATTACHMENT 2: STANFORD Budget Estimates

| | | |
|---|---|---|
| Development System (Year 1) | | |
| 6 Valid Workstations @ 20K (already funded) | | |
| I/O for Workstations: | | |
| 4 Marinco APs @4500 (already funded) | | |
| Special I/O | | |
| Sony Audio Processors & VCRs | 12K | |
| General Purpose I/O | 36K | |
| (mouse, tablet, modified mixer, etc.) | | |
| Snell Front End | 25K | |
| | 73K | |
| Software Development Processors | 200K | |
| 1 AP500 (already funded) | | |
| 4 Modems | 2K | |
| 1 Tape Drive | 5K | |
| Test Equipment | 50K | |
| | 257K | |
| Development Hardware Total | | 330K |
| Personnel | | |
| 1/2 John Chowning | | |
| Bill Scottstaedt | | |
| David Jaffe | | |
| Julius Smith | | |
| System Programmer | | |
| 1/2 Secretary | | |
| 4 Graduate Students | | |
| (@approx. 250K/yr) | 1250K | |
| Supplies including Hardware Updates @80K/yr | 400K | |
| Personnel & Supplies Total | | 1650K |
| Stanford Total: | | 1980K |

**ATTACHMENT 3: UCSD Budget Estimates**

| | | |
|---|---|---|
| Personnel: | | |
| 2 Visiting Researchers | 80K/yr | |
| 2 Graduate Students | 20K/yr | |
| | 100K/yr | 500K |
| Supplies including hardware updates | 80K/yr | 400K |
| Subcontract Expenses | 100K/yr | 500K |
| Development System Hardware (year 1 only) | | |
| High-speed processors | 200K | |
| 1 AP500 Array Processor | 35K | |
| 2 68000 Workstations (including Marinco APs) | 60K | |
| Special I/O | | |
| Snell Front End | 25K | |
| | | 320K |
| UCSD Total: | | 1720K |