T-MAX
A Parallel Processing Development System for MAX
Bruce Pennycook, D.M.A.
Chris Lea, M.Com. Sc.
Faculty of Music McGill University
brp@music.mcgill.ca
August 1991

Abstract
T-MAX is a parallel processing development system for MAX. It permits the implementation of MAX objects on an arbitrary number of INMOS Transputers. MAX objects developed on the Macintosh in Think-C are revised using Logical Systems C, a parallel C compiler for all types of Transputer hardware, to support a message passing communications scheme called TComm (Transputer Communications). T-MAX objects behave exactly like standard MAX objects except that instances of the object are spawned as parallel processes and execute remotely on Transputers. A system will be demonstrated using three TR80x Transputers (25 Mhz., floating point, 1 Mbyte DRAM per cpu) running in a Mac-Ilfx. Implementations of a real-time analysis system based on Cypher (R. Rowe, MIT Media Labs) plus other T-MAX objects will be shown.

Background
From 1987-1989, a live-performance software system called MIDI-LIVE [Pennycook, 1991] was developed. This system was written for the PC-AT/MPU-401 platform and is used by the author [Pennycook, 1989] and other composers to generate a variety of compositions. While MIDI-LIVE solved many of the problems of integrating computer-controlled synthesis, mixing, and sound processing via MIDI, the software did not attempt to encompass any form of musical understanding (real-time analysis) or generative operations (real-time composition).

In 1988, the author met with Robert Rowe at the MIT Media Labs and formed a collaborative effort to combine portions of Rowe's Cypher [Rowe, 1991], an analysis/generative system, with MIDI-LIVE, an event-based MIDI data production system. Rowe had demonstrated that real-time analysis and generation of musical events could be implemented using a collection of "cooperating agents" each assigned to different aspects of the problem. On the other hand, MIDI-LIVE offered a comprehensive MIDI data management system which could receive, capture, store, manipulate, and play MIDI data streams in a wide variety of ways all under temporal control of an external performer.

With the assistance of the Canada Council for the Arts and the Social Science and Humanities Research Council of Canada, a research program called the Automatic Improvisor Project was established by the author. This effort has led to a number of interesting results, one of which is T-MAX. Another aspect of the project has been the implementation as a stand-alone MAX Object of the principle analytical engine from Cypher - called, predictably, the Listener.

This paper presents the motivating forces, hardware and software architectures, and the object development process. It will also include descriptions and evaluations of two working programs implemented using T-MAX.

ICMC 229
Motivating Forces

The musical paradigm chosen for the Automatic Improvisor Project is tonal jazz improvisation. The reasons for this choice are: 1) most small ensemble jazz is performed without notated music; 2) the musical materials are well-defined harmonically and rhythmically; and 3) as a player of jazz, the 'soul' is well understood by the author. Although other musical styles could be investigated using the T-MAX environment, the Automatic Improvisor Project continues to focus on tonal jazz as an initial test case.

T-MAX stands for Transputer MAX. At the beginning of the project it was assumed that most personal computers could not provide sufficient computational resources to perform the many tasks required for the implementation of a real-time model of musical improvisation. Initially, a stand-alone parallel implementation of Cypher was undertaken but when MAX became commercially available from Opcode early in the project, all efforts were re-directed toward this powerful programming environment.

Another motivating factor grew out of a general interest in parallel architectures for modelling musical behaviors. Consider a jazz improviser listening to several streams of musical activity at once, analyzing these streams and constructing them to motivic, harmonic, melodic, rhythmic, and dynamic structures heard moments before in the piece or even during other performances while, at the same time, selecting and re-shaping these materials as accompanying or soloistic contributions within the overall musical fabric. This suggests that the tasks may be broadly partitioned into three basic musical operations - listening, pattern matching, and playing - using a large-grain parallel shema.

Admittedly, it may be presumptuous to suggest that our cognitive activities are necessarily based on parallel computational processes. After all, these are merely machine architectures and not biological systems. However, consider for a moment how an improvisor fails during a performance. When the tempo gets too hot or the chord progression too complex, we can hear the soloist's analytical skills begin to crumble resulting in poorly constructed melodic patterns or even long gaps (strategic rests?) in the output. During these moments of high information bandwidth certain aspects of the process significantly degrade yet at the same time, well-worn but possible incorrect patterns or fragments (perhaps of a motoric nature) are snapped from memory and sent to the hands to fill in the gaps. This suggests that some form of multiple processing must be in operation. Hopefully, the model under development described in part in this paper will permit more rigorous examination of these concepts.

Hardware and Software Architectures

The hardware platform for T-MAX and all of the research projects utilizing the Transputer parallel system is the Transputer/386 system at McGill. An Apple Macintosh IIfx. A NuBus card from Quantum Leap Systems provides the INMOS Transputers with up to eight, 25 MHz, T805 floating point devices each with 1 MWord of DRAM per board. The system is currently operating with 3 T805's. The MIDI interface (MOTU Midi Time Piece) and drivers (Apple MIDI Manager) are commercially available devices. Audio IO for the Waveguide Editor is provided via a Digidesign Sound Accelarator card with the DAT IO option.

The Transputer hardware was chosen as the development platform for several reasons including:

1. cost per unit (25 Mhz, 32-bit floating point, multi-tasking @ $600/cpu)
2. inter-processor connection points included in the hardware in the form of 20 Mbit/sec. serial paths called "links".
3. nearly linear improvement in performance with each additional processor
4. no recoding necessary to utilize larger arrays of devices (unlike collections of conventional CPUs such as 68040 accelerators which would require substantial reprogramming)
5. a complete parallel software development system was commercially available

These features permit easy expansion of the T-MAX system without the time-consuming (or impossible) explicit redistribution of code segments and inter-process communications. A set of utilities permit specification of the inter-processor connections (Link Configuration File) and the distribution of executable code segments among the T805’s (Code Map).

The Automatic Listener Project was predicated on large-grain parallelism such that each member of the multi-processing system performs the following operations:

1. system IO including MIDI, graphics, and mouse control (Macintosh)
2. Macintosh - Transputer communications (1 T805 and the Macintosh)
3. real-time musical analysis of multi-channel MIDI data streams (1 T805)
4. real-time pattern matching and improved output (1 T805)

The Waveguide Editor project, on the other hand, is based on small-grain parallelism. That is, many Waveguide nodes [Smith, 1987] are evenly distributed on each of two T805’s leaving the third processor to serve as the communications path for audio sample data from the Waveguides to Macintosh memory. (Samples are then passed to a disk file or directly to the conversion hardware.)

T-MAX Objects

T-MAX Objects such as the T-Listener are first developed and debugged as regular MAX Objects using the Symmetec Think C compiler and ResEdit. Next, the kernel (everything except the IO routines and MAX entry points) is ported to Logical Systems C, a parallel C compiler running under Apple’s MPW Shell.

The attached figure, T-MAX Object Development Cycle, shows the overview of the process of turning a working MAX Object into first, Transputer executable code, then finally into a T-MAX Object. A set of library routines called T-Comm were implemented to provide communications channels (message passing) between the MAX side of the Object and the code executing on the T805’s. Macintosh resource editors (REZ, ResEdit) are vital components of the development process.

This process results in a T-MAX Object which, when installed and launched under MAX, finds and initializes the Transputer board and processors, loads the executable code into the Transputers, and sets up communications channels between a MAX window and the remote processes. As each new instance of a T-MAX Object is called in MAX, the software generates process client-pairs on the Macintosh side and the Transputer side.

In the case of the T-Listener Object, MIDI data are first packetized by the MAX side of the code into Events (structures of time tagged MIDI packets). Next, the T-Comm routines prepare and route these packets to the Transputers. A companion set of T-Comm routines running on the first T805 in the Transputer network (acting as the communications handler) receive these packets and route them to the appropriate routines running on one or the other T805’s. The T-MAX Objects merely act as “empty” communications routines while the real work (pattern matching, harmonic analysis, etc.) is done on as many T805’s as required to maintain real-time throughput. Results from the various routines are returned to MAX by a similar procedure.
T-MAX Applications

The primary application of T-MAX within the Automatic Improvisor Project has been the T-Listeners. Incoming MIDI data streams are segmented into Events (several MIDI notes or chords) and passed to the Listener which then analyzes the data according to harmonic content and key area, register, rate of attack, dynamic levels, etc. These are returned to MAX in the form of bit-mapped 32-bit words and presented to the programmer as normal MAX data types through "callbacks" for further processing within the MAX program.

The Waveguide project required the development of a set of MAX editing windows to permit graphic description of the synthesis networks. Furthermore, a set of routines were required to buffer sample data generated by the Transputers and pass these as blocks to the Sound Accelerator card. The Waveguide project proved that an arbitrarily complex network of full-connected excitation elements, resonators, and junctions could be implemented in T-MAX. It also proved that complex editing windows could be constructed within the constraints of MAX.

Evaluation

Tests were constructed to evaluate the relative performance of Listeners vs. T-Listeners. These indicate that on a 40 MHz Macintosh IIx, the average computation time for the Listener Object is approximately 7 msec per MIDI Event. The time interval was measured from the arrival of data at a standard MAX notein to the output of the left-most Listener in the patch. The data was organized into time-tagged packets of up to 8 note chords. Using the T-Listeners, the same or slightly lower values are observed. Although no significant acceleration can be reported from this experiment, it should be noted that the design of the Automatic Improvisor includes real-time pattern matching and that both the Listeners and Matchers are run remotely on the Transputer hardware. We are certain that the computational demands of the these two tasks running on many MIDI channels at once will exceed even the MacIIx's throughput rates.

The Waveguide Editor cannot run without the Transputers so no direct comparison was possible. However, a simple table-lookup oscillator was shown to run faster than 3 times real-time @ 44.1kHz stereo. A small but fully-connected network of Waveguide unit generators runs at about 0.3 real-time @ 44.1kHz stereo. The small-grain parallel architecture and heavy inter-process message passing traffic for this type of sound synthesis result in very high T805 multi-tasking overheads.

Conclusions

A parallel computing platform for the pursuit of the Automatic Improvisor Project has been implemented and tested. By combining the powerful facilities of MAX with remote, parallel computation of analytical processes, a high-performance, expandable research environment has been developed at reasonable costs. The T-MAX programming environment will be used to implement new T-Objects as required and will be made available to other researchers interested in the application of parallel processing to music.

References

Pennycook, Bruce "The Praesicio Series: Composition Driven Interactive Software" CMJ, Fall 1991 (forthcoming).

ICMC 232
T-MAX Object Development Cycle

Object kernel sources

\textit{MPW/LSC}

Transputer executable code (.b4's)

Link Configuration File \quad Code Distribution Map

\textit{REZ} (MPW Utility)
(used to append .b4's, link map, and code map)

T-MAX Object's Resource File

\textit{ResEdit}
(used to append standard resources such as STR#)

MAX Object source code using T-Comm libraries

\textit{Think C}

T-MAX Object (code resource)