Real-time sound synthesis on a multi-processor platform

Itagaki, Takebumi

Real-Time Sound Synthesis on a Multi-Processor Platform

Ph. D Thesis

University of Durham
School of Engineering Department of Music
Music Technology Group

January 1998

Takebumi ITAGAKI
B. Eng (Waseda, JAPAN), P.G.Dipl. (City)

Real-Time Sound Synthesis on a Multi-Processor Platform

Ph.D Thesis January 1998 Takebumi ITAGAKI

Abstract

Real-time sound synthesis means that the calculation and output of each sound sample for a channel of audio information must be completed within a sample period. At a broadcasting standard, a sampling rate of 32,000 Hz, the maximum period available is 31.25 μsec. Such requirements demand a large amount of data processing power. An effective solution for this problem is a multi-processor platform; a parallel and distributed processing system.

The suitability of the MIDI [Music Instrument Digital Interface] standard, published in 1983, as a controller for real-time applications is examined. Many musicians have expressed doubts on the decade old standard's ability for real-time performance. These have been investigated by measuring timing in various musical gestures, and by comparing these with the subjective characteristics of human perception.

An implementation and its optimisation of real-time additive synthesis programs on a multi-transputer network are described. A prototype 81-polyphonic-note-organ configuration was implemented. By devising and deploying monitoring processes, the network's performance was measured and enhanced, leading to an efficient usage; the 88-note configuration. Since 88 simultaneous notes are rarely necessary in most performances, a scheduling program for dynamic note allocation was then introduced to achieve further efficiency gains. Considering calculation redundancies still further, a multi-sampling rate approach was applied as a further step to achieve an optimal performance.

The theories underlining sound granulation, as a means of constructing complex sounds from grains, and the real-time implementation of this technique are outlined. The idea of sound granulation is quite similar to the quantum-wave theory, "acoustic quanta". Despite the conceptual simplicity, the signal processing requirements set tough demands, providing a challenge for this audio synthesis engine.

Three issues arising from the results of the implementations above are discussed; the efficiency of the applications implemented, provisions for new processors and an optimal network architecture for sound synthesis.

Acknowledgements

I would like to express my gratitude to the many people who have supported and encouraged my research, including Dr Paul Archbold (Music) who helped my subjective testing, and Dr Desmond Phillips and other members of the Music Technology Group who gave various suggestions, criticisms and contributions.

I would like to acknowledge my thanks to my supervisors Professor Peter D. Manning (Music), whose enthusiasm for electro-acoustic music, especially in sound granulation, has driven my work and Professor Alan Purvis (Engineering) for his administrative support. I am also indebted to Professor Robert Provine (Music) in his support and help for the publication of my papers at international conferences.

I wish to thank Professor Barry Truax and Dr David Murphy of Simon Fraser University (Vancouver, CANADA) for their generous help and suggestions on the sound granulation techniques. My thanks are extended to my parents for their support, including financial matters; not only the university fees and my maintenance but also for a large portion of the costs incurred in publication of my works.

Contents

Chapter 1. Introduction (continued)

Chapter 2. Transputers

Chapter 3. Measurements of Keyboard Performance through MIDI

Chapter 4. Implementation of Real-Time Additive Synthesis on The Network

Chapter 5. Optimisation of Real-Time Synthesis Additive on the Network

Chapter 6. Granular Synthesis and Sound Granulation

Chapter 7. Analysis of Grain (Granulation Parameters and Sound)

Chapter 8. Implementation of Real-Time Granular Synthesis and Sound Granulation on the Network

List of Figures

List of Figures (continued)

List of Tables

List of Program Listings

Declaration

I confirm that no part of the material offered has previously been submitted by me for a degree in this or in any other University.

T. Itagaki

Statement of Copyright

Chapter 1. Introduction

1.1. Preamble

Composition and performance of music are plural activities that combine the outcomes of a number of procedures, many involving functions that operate in parallel. In terms of sound synthesis operations, a significant number of generative and Digital Signal Processing [DSP] processes involve a combination of concurrent elements, ranging from the production of simultaneous notes by a single instrument to the superimposition of totally independent outputs, where a number of different components contribute to the audio spectrum.

Traditional computer processors are serial devices, restricted for the most part to the execution of instructions as a single stream of events. Hence processes that require the aggregation of functions executed in parallel have to be simulated by some means of cyclical tasking and data accumulation. In the case of digital audio synthesis and signal processing applications, the resultant effects on overall processor performance quickly become significant, thus limiting the number of individual components that can be handled in real-time.

To synthesise a realistic tone of an acoustic instrument by additive [Fourier] synthesis using a bank of sine oscillators, a minimum of about 20 to 30 individual sine waves are usually required; with an ideal specification extending to more than 50 components in certain circumstances. A number of approximations to sounds, such as organ tones, can be achieved with as few as eight or ten components, but to the discriminating listener, the results will often prove of restricted value other than in a

purely synthetic context. Additive synthesis provides sophisticated control ability in complex timbres, and this technique has attracted the attention of many researchers since von Helmholtz (von Helmholtz 1863). The issue is also related to the analysis and re-synthesis of sound, which has led to systems such as the phase vocoder (Dolson 1986) and techniques such as group additive synthesis (Kleczkowski 1989).

Real-time digital additive synthesis means that the computation has to proceed rapidly enough to provide sound samples without failure within the required time interval: in the case of a sampling rate of 32 kHz, the lowest practical rate for digital audio in broadcast quality, this interval is 31.25 μsec. Solutions to meet the demands and constraints could be custom-made hardware like "Sine Circuitu" (Jansen 1991) and MIMIC (Wawrzynek and von Eicken 1990), a spectral modelling approach such as IFFT (Rodet and Depalle 1992), and stochastic decomposition (Serra 1994). Another approach our Music Technology Group has taken is based on a distributed parallel processing technique, using a multi-processor-based audio computer.

Since 1988, the Music Technology Group in Durham has reported on issues concerning multi-transputer audio processors on a number of occasions; for example (Purvis, Berry and Manning 1988) and (Bowler, et al. 1989). A prototype architecture for a transputer network, using sixteen T800 transputers, was described and demonstrated at the 1990 International Computer Music Conference [ICMC] (Bailey, et al. 1990). This was developed into a distributed parallel audio processor using 160 transputers inter-connected as a ternary tree. The network has

subsequently been used as a test-bed for a variety of network architectures for real-time synthesis.

In this introductory chapter, the criteria for real-time implementations of analogue and digital synthesis are reviewed, both in terms of traditional control methods and strategies, and also more modern approaches based on parallel and distributed processing, upon which the implementations described in this thesis are based.

1.2. Analogue Synthesis

In 1964, an American engineer Robert Moog built a transistor voltage-controlled oscillator and amplifier for the composer Herbert Deutsch. The development stimulated widespread interest, and led other American engineers to join the race to build a novel machine.

To control a musical sound, the basic requirements are measurable in terms of frequency [oscillator], harmonics content [filter] and amplitude [amplifier]. These three components were made voltage controllable, thus providing a common denominator of control. Varying voltages are easy to generate and to distribute to one or a number of associated devices, hence their attraction as a means of regulating the generation of sound. Analogue synthesis is often misleadingly called "subtractive synthesis", due to the preference of most users for configurations where timbres are generated by filtering the harmonics of raw electronic wave forms.

1.2.1. Voltage-Controlled Oscillator

A typical Voltage-Controlled Oscillator [VCO] can produce a number of wave forms; such as saw-tooth, triangle and square [or pulse] waves, in addition to the basic sine wave. A change in the control voltage means a change in pitch [frequency]. A space/mark ratio of a "pure" square wave is one. By varying the space/mark ratio with the controlled voltage supplied by an oscillator, the harmonic structure of the sound is changed by the oscillation; a narrower mark produces richer harmonics. This technique is known as Pulse Width Modulation [PWM], a very useful feature of a VCO.

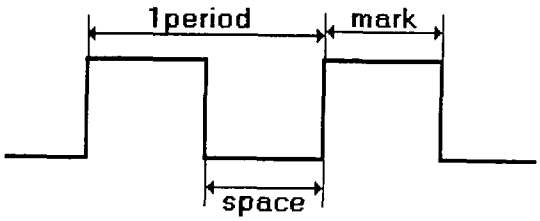


Figure 1.2.1.: Square Wave.

1.2.2. Voltage-Controlled Filter

The controlled variable of a typical Voltage-Controlled Filter [VCF] is either the cut off/centre frequency or the "Q" value [resonance]. For example, when the "Q" is kept constant [constant Q] and the frequency is set to track at a fixed harmonic spacing to the fundamental of a compound wave, such as a square or saw-tooth function, a variety of consistent timbres can be generated at different pitches.

1.2.3. Voltage-Controlled Amplifier

The controlled variable of a typical Voltage-Controlled Amplifier [VCA] is the output amplitude. VCAs are sometimes known as two-quadrant multipliers. When a VCA is modulated at low frequencies, the result is a "tremolo" effect, but higher rates of modulation fuse the spectra to give sum and difference frequencies, a phenomenon known as "Amplitude Modulation" [AM].

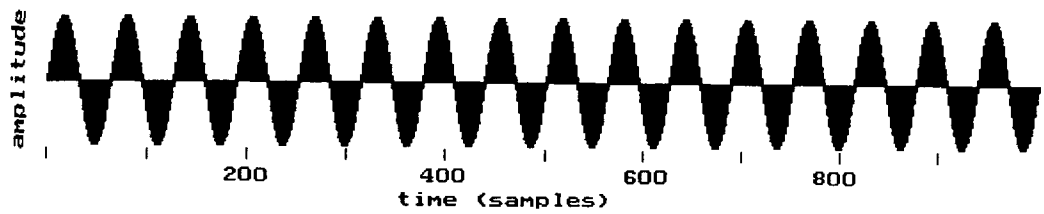


Figure 1.2.2.: Source Signal (500 Hz sine wave).

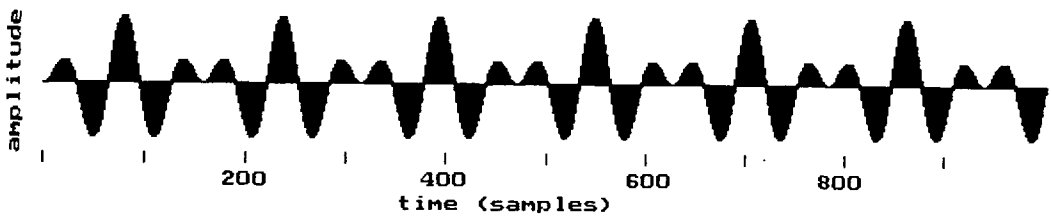


Figure 1.2.3.: Modified Signal (CV 100 Hz, Source 500 Hz).

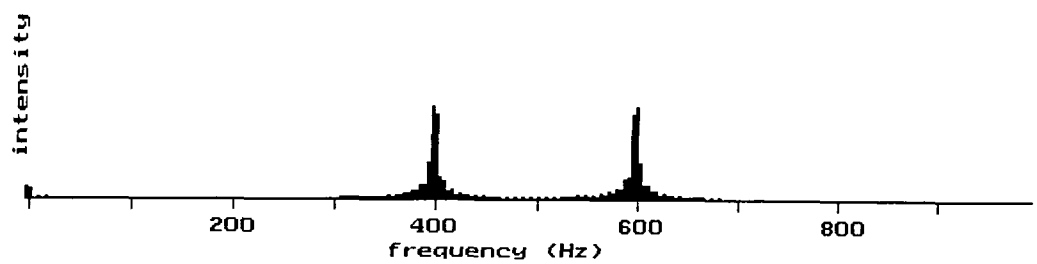


Figure 1.2.4.: Frequency Spectrum of Modified Signal (500 ± 100 Hz).

The side-band frequencies are generated as the source frequency [carrier] plus the control voltage frequency [modifier] and the source frequency minus the control voltage frequency. When the control voltage becomes negative, the output of the VCA is zero. If, however, the design is modified to create a four-quadrant multiplier with two-inputs and one-output, a device known as a "ring modulator", the negative control voltages can be used to produce negative frequencies with reversed phases by reflection.

It is significant to note that the early digital software synthesis systems allowed easy replication of many of the features in analogue voltage-controlled systems, and many of the techniques are still widely used in the current generation of software synthesis program, for example CSOUND.

1.2.4. Frequency Modulation

The theory behind the Frequency Modulation of high radio frequencies, in the order of MHz, dates back to the early twentieth century (Carson 1922). Chowning applied and explored the technique in the sound spectrum for musical synthesis purposes (Chowning 1973), commonly referred to as "simple FM" or "Chowning's FM", where a "carrier" oscillator is modulated in frequency by a "modulator" oscillator.

Before the development of Chowning's FM method, most digitally generated sounds were produced by means of fixed wave forms based on fixed spectrum techniques, a consequence of the high computational costs of time-varying additive and subtractive synthesis. Chowning developed the FM technique as an efficient way of generating synthetic sounds that have time-varying spectral characteristics. In 1975, Yamaha [known as "Nippon Gakki" at that time] obtained a licence for the patent. In 1980, this Japanese firm introduced the algorithm as a hardware fabrication for the GS1 digital synthesiser.

The basic FM technique is such that a carrier oscillator is modulated in frequency by a modulator oscillator. When the carrier and the modulator are both sine waves, the equation for a frequency modulated signal at time *t* is;

$$A \times \sin(Ct + [I \times \sin(Mt)])$$

where

A: amplitude of the carrier

I: index of modulation

$Ct = 2\pi \times C$, $Mt = 2\pi \times M$

The positions of the modulated side-band frequencies depend on the ratio of the carrier to the modulator frequency; "C:M ratio". The side-bands are multiples of the carrier and modulator; C + *n*M and C -*n*M, where *n* is an integer number. If the "difference" frequencies turn negative, these are folded over to the positive side with phase inversion: the wave forms flip over the *x*-axis. This "fold-over" can cause cancellation of the positive partials if the negative partials overlap exactly with the positive counterparts. In the case of a digital implementation, this "fold-over" also occurs where the "sum" frequencies exceed the Nyquist limit, [*see* Chapter 1.3.1.], a phenomenon known as aliasing.

1.3. Digital Technology and Sound Synthesis

1.3.1. Conversion between Analogue and Digital

A digital signal is represented in discrete-time, contrary to an analogue signal which is represented in continuous-time. The core concept of digital audio is sampling, converting continuous analogue signals into discrete time-sampled signals. An Analogue to Digital Converter [ADC] is employed for this job, the device converting analogue voltages to a string of binary numbers at each period of a sample clock; the timing, called "sampling period", "sampling frequency" or "sample rate", represents the resolution in the time domain.

Another parameter, the number of bits, is responsible for the resolution of amplitude measurement. For an ADC for audio sampling, at least a 12-bit bandwidth is needed, whereas the Compact Disc [CD] uses a 16-bit format, to achieve a much better dynamic range; 96 dB as opposed to 72 dB, much closer to that of the human ear which is about 120 dB between the threshold of hearing and that of pain (Roads 1996).

When a continuous analogue signal is quantised, the signal turns into a stream of binary numbers. Each of these is in an integer format, and is often fractionally smaller or larger than the analogue figure at the sampled point. This difference is called "quantisation noise". As the available bit-size is increased, the resolution in amplitude, "dynamic range", becomes higher [wider], and the associated distortion of the signal reduces proportionally.

Two processes present in the digitisation of audio signals occur together, but can be considered separately; quantisation in amplitude and quantisation in time. The former measures the amplitude of a signal, and assigns it to a scaled value drawn from the finite range of binary numbers that are available. In the case of the latter, finite binary numbers are registered to span a finite time internally, corresponding to the signal generated.

In the conversion process, the "sample-and-hold" method, is used. There is one practical problem; it assumes that the unknown voltage input does not change during the course of the conversion. If the voltage changes, significant conversion errors may occur. Thus the higher the rate at which the function is sampled, the lower the risk of error from rapid changes in the function itself [see below]. To convert a stream of binary numbers into an analogue signal, a Digital to Analogue Converter [DAC] is used. This device changes the string of numbers to a series of voltage levels that are smoothed with a low pass filter to a continuous-time waveform and amplitude.

The "sampling theorem" is the theoretical basis of digital audio and specifies the relation between the sampling rate and the audio bandwidth. Nyquist described it as follows:

> For any given deformation of the received signal, the transmitted frequency range must be increased in direct proportion to the signalling speed.... In order to be able to reconstruct a signal, the sampling frequency must be at least twice the frequency of the signal being sampled.
>
> Nyquist (1928) "Certain topics in telegraph transmission theory."

The highest frequency that can be produced in a digital audio system is called the "Nyquist frequency", to honour his contribution. In musical applications, the limit is usually at or above the upper limit of the human auditory range [about 20 kHz]; for example a Compact Disc's sampling frequency is 44.1 kHz while its Nyquist limit is 22.05 kHz. In some earlier commercial applications, and some for broadcasting purposes [32 kHz sampling rate], the Nyquist frequency was set at 16 kHz, still within the range of hearing, but at a point where the sensitivity of the human ear starts to decline significantly.

When a sine wave of 1 kHz and another of 7 kHz are sampled at a frequency of 8 kHz, the results of both are indistinguishable. In this case, where the Nyquist frequency is 4 kHz, the 7 kHz wave is folded over at the Nyquist frequency; 4 - (7-4) = 1, a phenomenon called "aliasing". To prevent aliasing, the input analogue signal would have to be filtered with a low pass filter set to cut off frequencies above the Nyquist limit.

In the case of digital-to-analogue conversion, similar folding occurs at the higher frequency band of the Nyquist limit and on both sides at every multiple of the sampling frequency, called "image spectra". To remove the image spectra, the converted analogue signal has to be filtered with a low pass filter with a cut off frequency at or preferably just below the Nyquist limit. The filter is also called a "data recovery filter", because of the task it performs.

1.3.2. Compact Disc

In 1982, digital sound reached the general public by means of the Compact Disc [CD] based upon specifications jointly developed by Sony [Japan] and Philips [Holland]. Their applications for the optical reader [Compact Disc Player], however, are slightly different, especially the movement of the reading laser head; Sony's moves on a straight line, whereas Philips' follows a banana-shaped slit. This leads to another difference in their control strategies of the disc rotation speed. The sampling rate of the format is 44.1 kHz and the digitised numbers are in 16-bit integer format.

In the mid 80's, many CD player manufacturers used a DAC chip set designed by Sony or Philips, the sole patent holders, that brought the benefits of over-sampling technology, using a Finite Impulse Response [FIR] filter, to home audiences. Digital filters in the DAC chips provide a much more linear phase response then the steep analogue filter used in a regular DAC chip. In a CD player, the original samples, at 44.1 kHz, may be "up-sampled" four or eight times to 176.4 kHz or 352.8 kHz, far higher then the human auditory range [maximum about 20 kHz], by interpolating the samples.

In the case of the four-times-over-sampling mode, three interpolated samples are placed between two original samples to recover the required samples. To do so, the total quantisation noise is spread over a wider frequency range, thus providing a much higher signal-to-noise ratio in the human auditory range. The image spectrums are also shifted far away, thus a less steep data recovery filter can be introduced. If a steep filter

has to be used for the purpose, its phase response is not so satisfactory, and the computation cost of a steep digital filter is very expensive. This over-sampling technique is used in the optimisation of additive synthesis by application of multi-sampling-rate, described in Chapter 5.

1.3.3. Wave-Table Synthesis

Since musical sound waves are highly repetitive, a more efficient technique is to have the hardware calculate a digital waveform as a series of numbers, for just one cycle, which are then stored n a list called a "wave-table". To generate a periodic sound, the wave-table is read through repeatedly, and the samples are sent to a DAC. This process is called "wave-table synthesis", "table-lookup synthesis" or "fixed-waveform synthesis". Since a memory access operation is much faster than the calculation of a sample, the method is highly effective for a digital oscillator.

To generate a frequency other than that obtained by reading every sample at the basic sampling rate, a read pointer, or an address generator, is employed that steps the wave-table and then outputs its contents. To generate an octave up, the pointer skips every other sample. In the case of an octave down, the pointer reads every sample twice. In this way, frequency multiples of the stored more can be generated.

When a frequency between these octaves is needed, the frequency can be obtained by an accumulator/increment method.

$$i = \frac{n \times f_o}{f_s}$$

where

i : address increment.

n: table size.

f_o: output frequency.

f_s: sampling frequency.

The figure given by the equation is not always in integer. As the digitised table address is always in integer, the fractions of the increment have to be rounded. A more precise waveform, with a lower distortion factor, can be generated by interpolation of samples. This requirement, however, carries a significant computational overload (Mathews 1969).

1.3.4. Analysis/Re-Synthesis

One way to synthesise a sound is accumulating sine waves in different frequencies and phases. This method is called "additive synthesis" or "Fourier synthesis", and requires a large number of oscillators with accurate frequency and phase control. Most additive synthesisers imitate natural instruments by analysing sounds in terms of their harmonic profiles via filter banks or fast Fourier transforms [FFT] that bridge between the time domain, wave forms and sample values, and the frequency domain, the amplitude and phase of frequency components.

In an early analysis/re-synthesis method (Fletcher et al. 1962), the system was entirely analogue. Filtered input signals were measured via filter banks, and the information was then used to drive a bank of oscillators

corresponding with the filter bank. Solutions to meet the demands and constraints could be custom-made hardware like "Sine Circuitu" (Jansen 1991) and MIMIC (Wawrzynek and von Eicken 1990), a spectral modelling approach such as IFFT (Rodet and Depalle 1992), and stochastic decomposition (Serra 1994). The "FFT^{-1}" or Inverse FFT [IFFT] method is a hybrid method of overlap-add and oscillator bank re-synthesis. At the re-synthesis stage, an inverse FFT is carried out, hence the naming of the method.

1.3.5. Software Synthesis

Software synthesis is the most precise and flexible way to generate digital sound on a general-purpose computer. Software controls all the computation involved in a stream of samples, and the control functions themselves can be changed in arbitrary ways by the programmer. Authoritative examples of the method include the "Music V" language (Mathews 1969), and its predecessors, the "Music *N*" varieties.

As the importance and the flexibility of the programming language C became more generally recognised, in 1980, Moore and Loy developed "CMUSIC"; a much expanded version of "Music V". In a similar vein, Vercoe translated "MUSIC 11" into a "C" based sound synthesis language, known as "CSOUND" (Vercoe 1986). These languages are based on the concept of "unit generators", which are signal processing modules, such as oscillators, filters and amplifiers, that can be introduced to form "instruments" or "patches" generating sounds.

The languages based software synthesis identified above requires a basic knowledge of computing that leads to user-unfriendly "instruments", especially in the case of musicians without such detailed prior technical knowledge. For example, in the case of CSOUND, two files, a "score file" and an "orchestra file", are required to generate a sound output. The "score file" dictates the timing of the sound and key performance data such as pitch and amplitude. The "orchestra file" consists of details of the "instruments" themselves. These files are "compiled", like that of "source files" in C-language. As a solution for the problem, a "toolkit" concept was introduced that offers musicians a set of modules for creating interactive performance situations. MAX (Puckette 1985) is an example of one of these approaches.

MAX is an iconic toolkit targeted for interactive music performance. Some of the MAX icons take musical information from MIDI and others from audio sources. Other icons, connected by "patch cords" on the display screen of a NeXT or a Macintosh [MIDI data only, in a commercially released version], decode and transform these data. The icons, or "patches" are, therefore, black boxes for the musicians who only know the contents of input and output data. In the case of MAX, sound production control could be through MIDI input, or interaction with an icon or a programmed patch. What distinguishes these black boxes from equivalent functions in programs, such as CSOUND, is the case with which the various functions can be manipulated graphically as object-oriented components, rather than in terms of integrated lines of programming code.

1.3.6. Real-Time Synthesis

Real-time sound synthesis means the calculations for a sound sample must be completed within a sample period, such as 31.25 μsec at a 32 kHz sampling rate, so that the stream of sound samples flow constantly without failure. This is a definition of "hard" real-time. For real-time sound processing, therefore, a large amount of data processing power is often required. Measured by the processing time required [or computation costs], the task may prove too large to complete within the real-time window. The result, thus, is the introduction of some delay between the commencement of computation and the production of sound.

In a wider meaning of "real-time" ["soft" real-time], however, about 10 to 20 msec of constant delay can be tolerated, since the human auditory system can not detect such a short delay providing it is kept constant; [see Chapter 3.2.1.]. A system with a time-lag that is long enough to be detected is called a "non-real-time" system, and can only function with the aid of some intermediate storage and accumulations systems to aggregate the stream of sound samples for subsequent retrieval and output.

1.4. Controller for Music Performance

1.4.1. History of Electronic Music Keyboard Controller

Early electronic synthesisers were usually monophonic and their keyboards only supplied one control voltage output corresponding to one of the keys pressed, used typically to feed a voltage controlled oscillator. The control voltage produced was directly proportional to the position of the key on the keyboard; usually one volt per octave irrespective of the absolute frequency; 0.0833 [1/12] V increase per key. In the case of a very simple design, when more than one key is pressed, serial key resistors are shorted and an untempered voltage is produced. More sophisticated keyboard designs disable all but one key whenever two or more keys are depressed. The usual arrangement is that only the highest key in a note cluster is allowed to made contact with the voltage source.

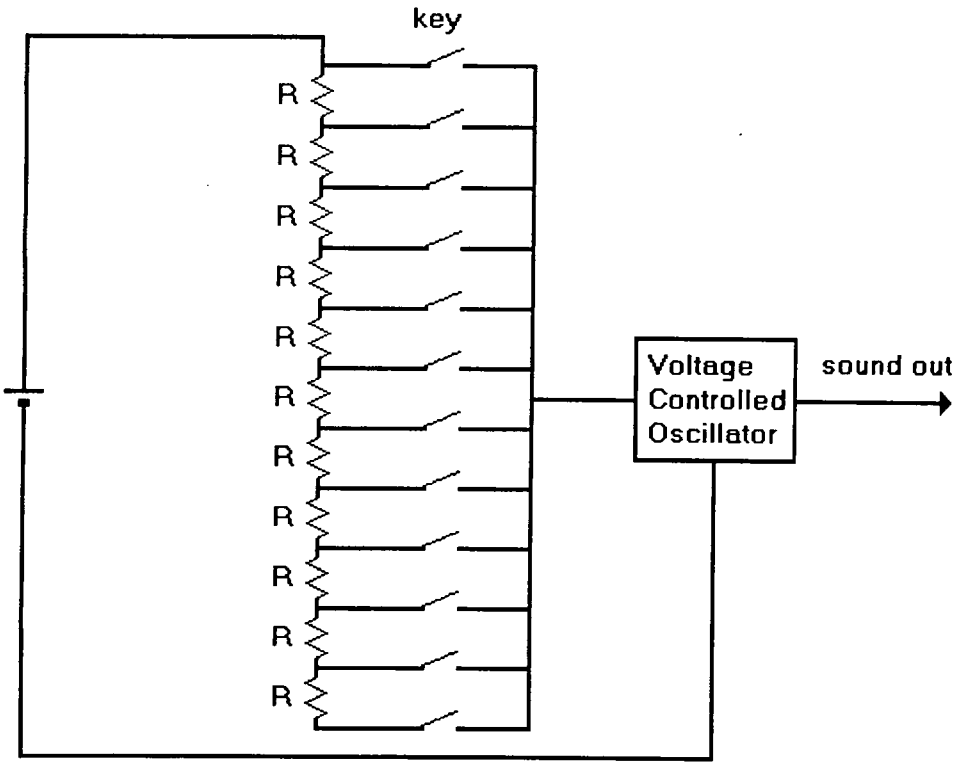


Figure 1.4.1.: Basic Diagram of Voltage-Controlled Synthesiser.

In electronic terms, this can be achieved by driving the keyboard chain with a constant current, instead of a constant voltage. Such a basic keyboard controller, however, is still not a suitable device to control a voltage controlled oscillator. When keys are pressed, no control voltage is generated with the result that the voltage controlled oscillator's pitch will be maintained only for the duration of the key-press. To solve this problem, a voltage memory for each key is required; this sample-and-hold facility is sometimes added as a module in its own right.

Around 1976, an American company "Sequential Circuits" released a polyphonic synthesiser called "Prophet V" that was digitally controlled by a microprocessor, the Z80. This eight-bit microprocessor, released in 1975 from Zilog, made many more features available for keyboard-synthesisers; such as a polyphonic mode, programmable sounds, automatic tuning and communication, and also required a DAC for the synthesiser itself, since the processor generates the control voltages in a digital format. As a minimum specification twelve bits are usually required for the generation of pitch control voltages of an adequate resolution in a standard equal musical temperament. For the control of amplitude, a minimum of eight bits are required for acceptable smooth envelopes.

In 1981, an all-digital keyboard synthesiser, called "Synergy", was released from an American company; Digital Keyboards, Inc. The synthesiser was based on a digital circuit that simulated 32 oscillators and provided convenient mechanisms for controlling their amplitude and frequency independently in real-time with an inexpensive microprocessor; the Z80.

In the early 1980s', the development of commercial systems for electronic and computer music came to a turning point, as the technology progressively shifted from the analogue to the digital domain. This forced the synthesiser manufacturers to consider fundamental issues of compatibility more seriously than before and led to an historic agreement; a common protocol for connecting different items of equipment together at a control level. For example, without a common standard, especially in a control voltage environment, connecting one manufacturer's keyboard to another's synthesiser would result in some quite bizarre consequences, because of the different voltage-to-pitch rules employed. Although connecting unmatched analogue voltage-controlled devices will produce some form of response, any mismatch in digital systems will usually produce no response at all.

In the early summer of 1981, the idea of establishing an industry-standard of digital protocol for connecting synthesisers and associated devices together at a control level was informally discussed at a meeting of the National Association of Music Merchants [NAMM], an American organisation. The initiative became a forum for a feasibility study of a universal communication system that came from the President of Sequential Circuits, Dave Smith. At an Audio Engineering Society [AES] convention, Smith and Chet Wood presented their proposal for the Universal Synthesizer Interface [USI] that was an outline description of a protocol to transmit note/event information between synthesisers.

In the summer of 1982, the initiative passed to the Japanese manufacturers, including Casio, Kawai, Korg, Roland and Yamaha, leaving only Sequential Circuits to represent the American interests. By September, this new grouping of companies completed a draft of an expanded specification for what became known as the Musical Instrument Digital Interface [MIDI] finally announced by Robert Moog in an article that appeared in the October edition of the magazine *Keyboard*. In the spring of 1983, the definitive version of the specification, *MIDI specification Version 1.0*, was published by the newly formed International MIDI Association based in the USA.

MIDI is a specification of a communication protocol that makes it possible to exchange information such as musical notes and expression control between different musical instruments or other devices, such as a sequencer, computer and mixer. These abilities to transmit and to receive data to a common specification were originally conceived for live performance, although subsequent developments have had an impact in the recording studio, audio and video production, and also the composition environment. Now, it is regarded as a standard for the digital representation of musical events at a control level.

The original motivation for the standard was to allow commercial synthesisers to be connected together, thus they might share performance information. Other benefits sought included hardware extendibility, protection from obsolescence, and interfacing to digital computers. It is important to note that MIDI was designed as an event-based network, not a sample-based one. MIDI was loosely adapted from the serial data

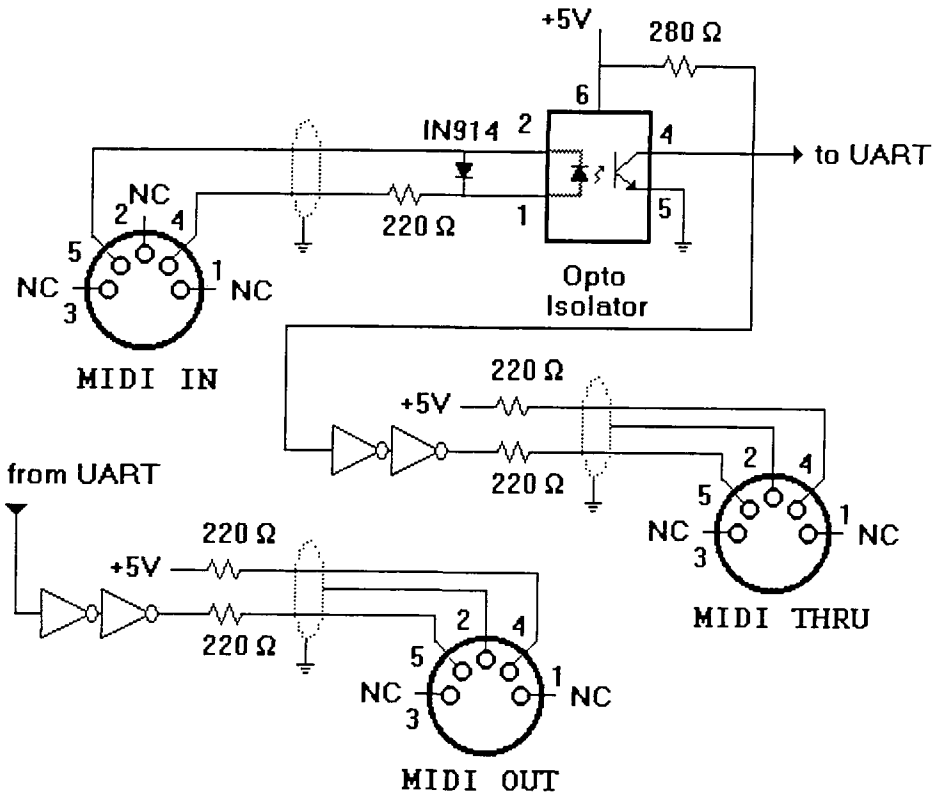
transmission technology developed for computer terminals. The basic idea involves a two-layer specification; a physical interconnection scheme, and a code to communicate information across the channels so created.

MIDI has been an industrial standard for music instruments, including the keyboards, for more than ten years. This well-established standard is commonly the best available controller for real-time synthesis systems, rather than custom-made keyboards that became extremely rare. An additive synthesis implementation basically requires control of amplitude, frequency and possibly phase. The amplitude and frequency of a sine wave can be controlled with MIDI signals; key-number [frequency] and velocity [amplitude]. MIDI's hardware and software specifications are described later in this Chapter. The suitability of a MIDI keyboard as a real-time controller is discussed in Chapter 3.

1.4.2. MIDI Specification

1.4.2.1. Physical Specification

The physical medium is a simple point-to-point opto-isolated 5 mA current loop, utilising a unique 180 degree 5-pin DIN connector: pins 1 and 3 are not used, and should be left unconnected. The cable is made of a shielded twisted pair; the shield being grounded only at the source end. Each twisted pair is a separate link that implements a one-direction transmission line. To avoid ground loops and subsequent data errors, the transmitter circuit and the receiver are internally separated by an opto-isolator.



International MIDI Association. (1983)
"MIDI Musical Instrument Digital Interface Specification 1.0"

Figure 1.4.2.: MIDI Standard Hardware.

A MIDI device will normally have a MIDI input [MIDI IN] jack and an output [MIDI OUT]. A device can have a through [MIDI THRU] jack, which passes a buffered electrical copy of the input signal: MIDI IN is connected to MIDI THRU through an opto-isolator, but not to MIDI OUT. Information is transmitted as asynchronous serial data at an aggregate data rate at 31.25 Kbaud with an allowance of ±1%. Serial MIDI data is transmitted as ten-bit code bytes; a start bit, eight data bits [00h to D0h] and a stop bit.

Some Interconnection schemes for multiple synthesisers are as follows:

Uni-directional a master talks to a slave.

[Chain one master drives several slaves.]

Bi-directional two masters drive each other as slaves.

[Ring bi-directional connection to more than two.]

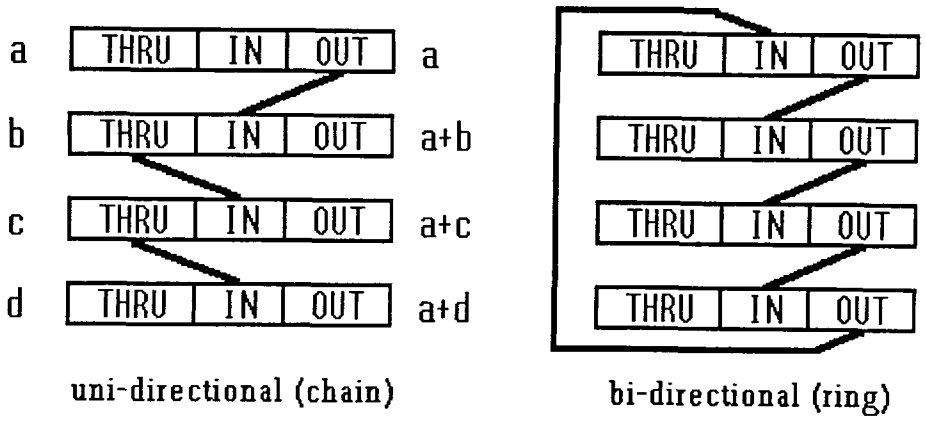


Figure 1.4.3.: MIDI Network Configurations.

Nevertheless, due to signal degradation largely attributable to cable capacitance and the response time of the opto-isolator, there are some limitations on the length of cables and the number of devices that can be chained in this function. The limitation on the number of chained devices is not defined in the specification. The maximum cable length in any chain, however, is restricted to 15 metres.

1.4.2.2. Code Specification of MIDI

MIDI communication is achieved through multi-byte messages consisting of one "Status" byte followed by one or two "Data" bytes, the only exceptions being "Real-Time" and "Exclusive" messages that permit

longer data strings. Thus each MIDI event is transmitted as a message and consists of one or more bytes. Messages are divided into two main categories: "Channel" and "System".

There are sixteen channels and three modes: the channels provide for multi-synthesiser control with a single MIDI network, while the modes establish the relationship between the channels and the voice assignment method within a synthesiser. A "Channel" message uses four bits in the "Status" byte to address the message to one of sixteen MIDI channels, and four bits define the message. "Channel" messages are thereby intended for those receivers in a system whose channel number match the channel number encoded into the "Status" byte.

There are two types of "Channel" messages; "Voice" and "Mode". Voice is to control an instrument's voices and Mode is to define the instrument's response to Voice messages. The modes are called "Omni" [on/off], "Poly" and "Mono". Four mode messages are available for defining the relationship between the sixteen MIDI channels and the instruments. These modes operate between a receiver and a transmitter assigned to the channel *N* [one of sixteen channels]:

MODE	ACTION
Omni on, Poly	Voice messages are received from ALL channels, and assigned to voices. All voice messages are transmitted in Channel N.
Omni on, Mono	Voice messages are received from ALL channels, and control only one voice. Voice messages for ONE voice are sent in Channel N only.

MODE	ACTION
Omni off, Poly	Voice messages are received in Channel N only, and are assigned to voices. Voice messages for ALL voices are sent in Channel N.
Omni off, Mono	Voice messages are received in Channel N through N+M-1, and assigned monophonically to voices 1 through M: where M is specified by the third byte of "Mono Mode" Message.

"System" messages are not encoded with channel numbers. There are three types of "System" messages: "Common", "Real-Time" and "Exclusive". "Common" messages are intended for all receivers in a system, regardless of channel. "Real-Time" messages are used for synchronisation and are intended for all clock-based units in a system. They contain "Status" bytes only. "Exclusive" messages can contain any number of "Data" bytes, and can be terminated either by an "End of eXclusive" [EOX] or any other "Status" byte.

There are two types of bytes sent over MIDI: "Status" bytes and "Data" bytes. "Status" bytes are eight-bit binary numbers in which the Most Significant Bit [MSB] is set to binary one. They serve to identify the message type, and also the purpose of the "Data" bytes that follow it, except in the case of "Real-Time" messages. For the "Voice" and "Mode" message only, when a "Status" byte is received and is processed, the receiver remains in that status until a different "Status" byte is received; called the "Running Status". Following a "Status" byte, there are either one or two "Data" bytes that carry the content of message. "Data" bytes are eight-bit binary numbers in which the MSB is always set to binary zero.

For each "Status" byte, the correct number of "Data" bytes must always be sent.

In the case of a single keyboard performance, three bytes are required for a note. Since the "running status" is applied in such cases, a status byte is only sent whenever another controller, such as the pedal, is used or other messages, such as "system exclusive" or "real-time", are sent. It means that most "note" messages are coded in two bytes requiring about 640 μsec for each message.

These specifications were established more than a decade ago, based on the technology available at that time. Due to the rapid evolution in the speed of modern computing technology, the restrictions of such components are now open to review, not least in the light of the far greater communication demands which may be seen in contemporary MIDI configurations. In order to assure the viability of MIDI as a control mechanism for the systems which are inherently complex due to the nature of their intensive parallel architecture developed for this research project, it was deemed necessary to carry out an evaluation of these characteristics in the context of recorded performances. The results of these investigations are described in Chapter 3.

1.5. Parallel and Distributed Processing

1.5.1. Background

Parallel and distributed processing may be defined as a technique for increasing the computation speed for a task by dividing the various algorithms into several sub-tasks distributed between multiple processors that execute the tasks concurrently. There are many problems to be solved in implementing sub-tasks over a parallel and distributed system. One of the classical examples is the producer/consumer problem: a process [the producer, such as oscillators] generates a stream of data to be sent to another process [the consumer, such as a DAC]. As there may be fluctuations in the rate of production/consumption of data, failure in the integrity of the data flow can lead to a deadlock of the system, locally or globally. An overloaded processor often causes such fluctuations and, where several processors are involved, the chances of any one failing to maintain the required throughput is significantly increased. Such problems can only be resolved by careful programming and real-time performance monitoring.

1.5.2. Parallel Processors

A parallel processor is a computer consisting of two or more unitary processing modules that are linked together physically and computationally such that applied tasks can be divided up and computed concurrently. There are two major features of a parallel processor:

1) the processing units themselves.
2) the inter-connection network which ties together the series of processors.

Parallel processors can be categorised by the topology of their interconnection network, and by their use either of shared-memory, such as Cray Y-MP, or distributed memory, such as nCUBE. Within the shared-memory category, the machines are further divided into "vector" or "MIMD" types.

The simplest inter-connected network is a bus connecting many processors to a single shared memory. A classical problem of designing a bus connected network is that of cache memory design, since a bus-structured processor without sufficient cache memory would quickly saturate the bus.

Distributed-memory designs offer higher levels of parallelism through the interconnection of thousands of processors that may require programmers to adopt a message-passing paradigm, since there is no realistic possibility of a global memory that could act as a shared resource for a global program. The design of a distributed-memory processor places great demands on communication speed and routing.

The distributed-memory approach is, in principle, scaleable to massive proportions: the number of processors can be increased without a significant decrease in the efficiency of the operation. When programmers are willing to adopt a programming model based on message passing or data parallelism, the scalability of a distributed memory computer becomes attractive.

A distributed-memory inter-connected network consists of processors and their local memories connected by communication links. Since there is no global memory it is necessary to move data from one local memory to another by message-passing, achieved by a send-receive pair of commands which are software-generated.

The simplest network is a linear inter-connected network, where each node contains a processor with its local memory, for example a one-dimensional cascaded pipe-line; such as NeXT boards on an IMW (Puckette 1991). A linear network of N nodes requires N-1 links to construct. On average it takes approximately N/3 hops, or point-to-point links, to send a message from a source processor to a destination one.

It is possible to reduce the number of hops by increasing the dimensionality of the inter-connected network using configurations such as ring, star network, mesh (Aspnäs, et al. 1990, and de Vel and Thomas 1990) and tree (Boittiaux, et al. 1992, and Maehle and Obelöer, 1992). The goal of inter-connected network design is often to reduce hardware costs by reducing links, at the same time minimising the time taken to send a message by reducing the number of hops.

1.6. Summary

In this chapter, the essential principals of key methods of analogue and digital sound synthesis have been reviewed with particular reference to the desired characteristics of keyboard interfaces and control methods, notably those based on MIDI. Real-time implementations of digital additive synthesis methods require large quantities of calculation power, and whereas one solution to performance limitations clearly lies in simply increasing the speed of processors, this overlook the possibility of developing new architectures which make more efficient use of finite resources, and may thus also be more cost-effective.

It is proposed that a suitable investigation of the latter approach may ultimately reap greater reward. One such method of exploration lies in the development of parallel and distributed computation techniques, and the account which follows describes the result of research into a series of such investigations using a specially fabricated network of processors which are described in Chapter 2.

Subsequent chapters deal in turn with the requirements for real-time control of a MIDI-based system; the implementation of real-time additive synthesis on the network and the optimisation of this method for particular applications, from organ-like synthesis engines to functions for granular synthesis and sound granulation.

Chapter 2. Transputers

2.1. Transputers and other DSP chips

2.1.1. Transputers

The Transputer™ family of devices, designed by INMOS Ltd. [now a part of the SGS-Thomson Group], offer versatile building blocks for the construction of multi-processor computing engines that are capable of establishing a high degree of parallelism. The word "transputer" was derived from TRANSmitter and comPUTER. Each transputer is a self contained high-performance single-chip computer with a RISC [Reduced Instruction Set Computer] architecture and distinctive inter-processor communication facilities. The transputer architecture defines a family of programmable VLSI [Very Large Scale Integration] components.

A typical member of the transputer family is a single chip consisting of processor, memory and communication links. In comparison with other micro-processors, the transputer has two very special features: it has on-chip serial links for "talking" to other transputers and hardware support for time-sharing. The serial communication links allow networks of transputers to be connected by direct point-to-point connections without additional external logic.

A T800 transputer, first announced in 1986, consists of a 32-bit CPU, a 64-bit Floating-point Processing Unit [FPU], four standard transputer communication links, a 4k-byte of on-chip RAM and an external memory interface. This general purpose DSP chip achieves 8.77 MIPS at a clock speed of 17.5 MHz [10 MIPS at 20 MHz], that is almost as fast as Motorola's DSP56000 [10.25 MIPS at 20 MHz], and its FPU performs in

excess of 1.32 MFLOPS sustained in 32-bit. The communication links can perform a sustained 1.74 Mbyte/sec in a uni-directional mode or 2.35 Mbyte/sec bi-directional.

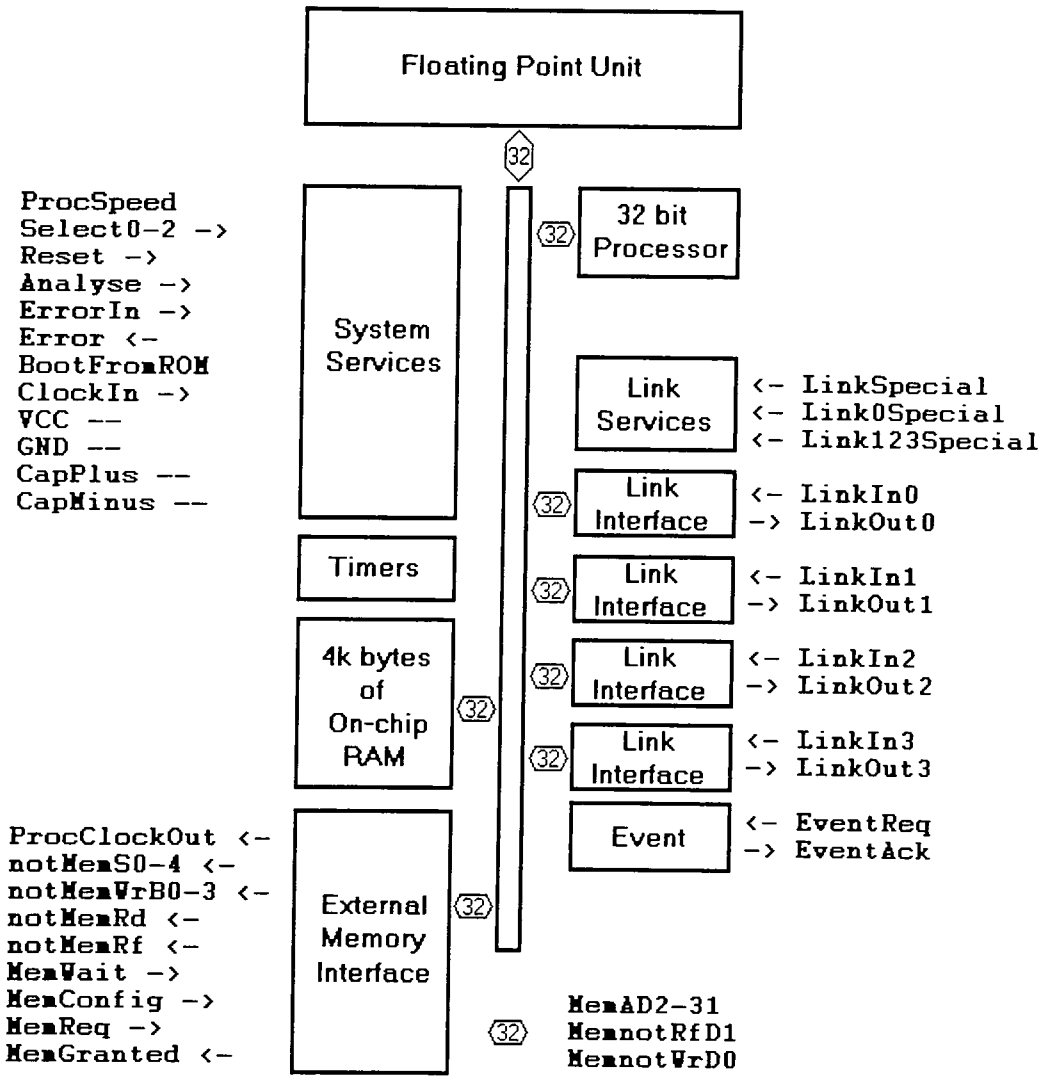


Figure 2.1.1.: Block Diagram of T800.

2.1.2. Other Architectures

2.1.2.1. DSP56000

Motorola's DSP56000 family, announced in 1986, is a popular user-programmable DSP chip that has been used for a number of applications in our Music Technology Group and also features in a number of commercial products for digital audio. As a general purpose DSP chip, a DSP56000/1 has a 24-bit data-bus, a 15-bit parallel port, and a 9-bit SSI and SCI interface for communication with another device.

Being read-only memory [ROM] based, the DSP 56000 version of the processor is factory programmed with user software for minimum cost in high-volume applications. Being random-access memory [RAM] based, the DSP 56001 is also capable of loading its program from an external source.

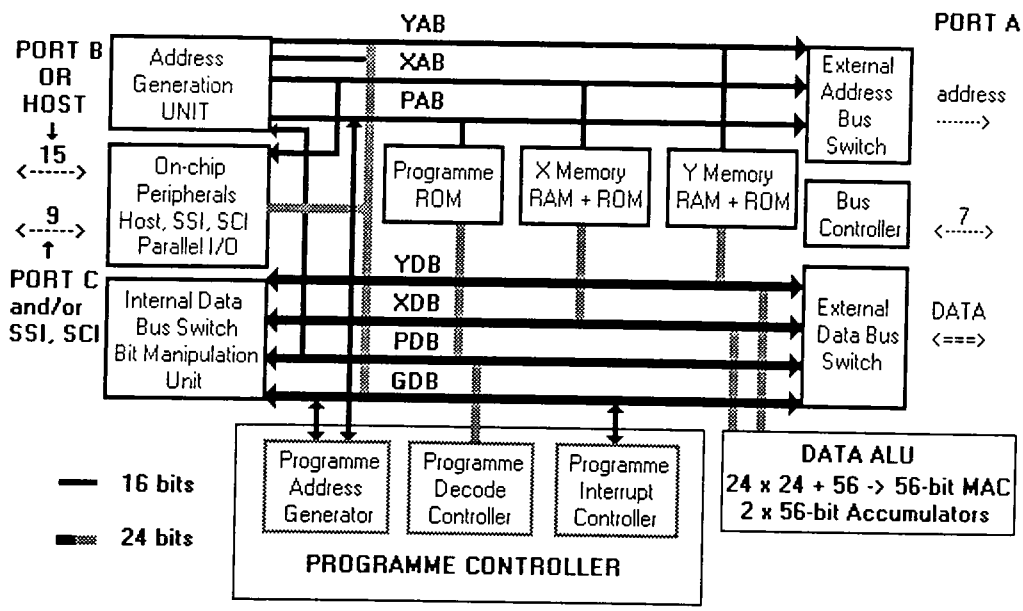


Figure 2.1.2.: Block Diagram of DSP56000/1.

The DSP 56000/1 can perform 10.25 MIPS at a 20 MHz clock rate, as fast as the T800. Each on-chip execution unit, memory and peripheral communications link operates independently and in parallel with other units through the bus system: this is a self-contained time-multiplex system establishing a significant degree of internal parallelism, but in other respects is no different to a modern monolithic processing unit.

2.1.2.2. TMS320C40

Texas Instruments' TMS320C3x generation of processors took an important first step in addressing the needs for parallel processing by means of pipe-lined processing units, providing designers in addition with two external ports with associated memory interfaces. In the next generation, announced in 1987, TMS320C4x, the devices go several steps further by incorporating on-chip hardware to facilitate high-speed inter-processor communication and concurrent I/O without degrading performance.

The TMS320C40 [C40] has six communication ports capable of 20-Mbyte/sec asynchronous transfer-rate at each port, whereas the T800 has four communication ports. The DSP chip has a register-based CPU architecture that contains a pair of CPU-buses with register-buses. The CPU is capable of 275 MOPS [25 MFLOPS], with 40-ns and 50-ns instruction cycle times [25 MHz and 20 MHz]: this is thus a parallel DSP chip with more than one CPU on board. The C40s have been used by our Music Technology Group for the analysis and re-synthesis of musical sound.

2.2. Transputer Development System

2.2.1. Introduction

The Transputer Development System [TDS] is an integrated development system that can be used for developing Occam programs for a transputer or a network of transputers. It consists of a plug-in board, "mother board", for an IBM PC [Transtech TMB04] and all the appropriate development software.

2.2.2. Hardware Descriptions

The TMB04 is an expandable transputer board for an IBM XT or AT and their compatibles. The board consists of a transputer [T800] with local memory and 2M Bytes of fast DRAM. There are four slots for adding further transputer modules [TRAM] as daughter-boards.

Each slot on the mother-board is made up with 160 dual-in-line [DIL] pins and is arranged in a hard wired pipeline: link No. 2 of the No. 1 daughter-board is connected to link No. 1 of the No. 2 daughter-board. The TMB04 board is equipped with a 37 way D-type connector with a special adapter. The adapter allows the master transputer's links to be connected to external transputers and to the daughter boards' links that are not used for forming the pipeline.

There are two ways to communicate between the mother-transputer on the TMB04 and the host PC's IBM bus: the link adapter and a DMA mechanism. The former is the simplest form of data transfer between the PC bus and the TMB04. It is often referred to as a "B004" interface using

a transputer link adapter chip IMS C012. The latter is achieved with the DMA interface chip 8237.

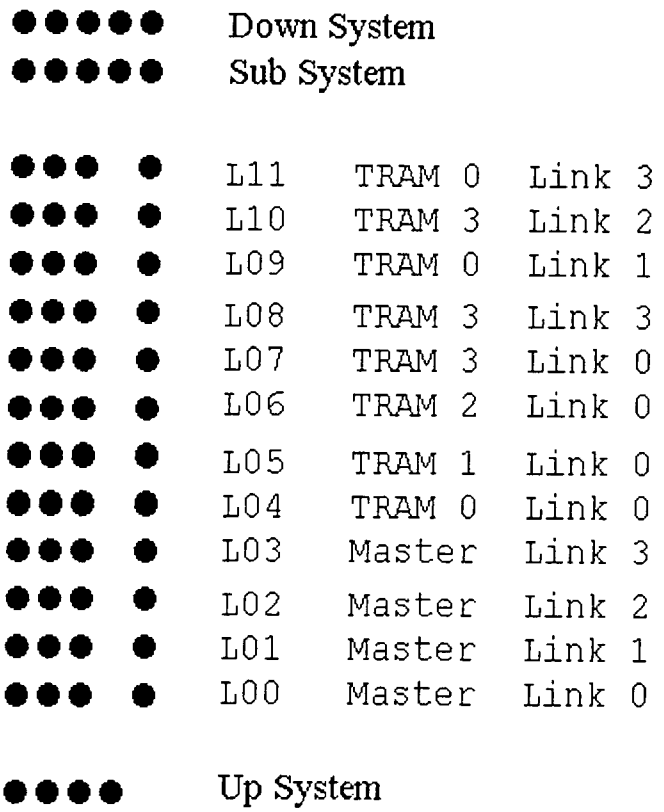


Figure 2.2.1.: Pin Alignment of the 37-way D-type Adapter.

2.2.3. Software Descriptions

Most of the development system runs on the transputer mother board; a program "server" on a host PC that provides the TDS with access to the terminal and filing system of the PC. The TDS allows programmers to edit, compile, run and debug Occam programs entirely within the development system. Occam programs can be developed on the TDS and configured to run on a network of transputers with the code being loaded from the TDS. The TDS is accompanied by all the necessary

software tools and utilities to support this kind of development, including a variety of Occam library routines to support mathematical functions and input/output routines.

There are, however, few provisions for tools designated to the networked transputers, with the exception of the network tester that examines each processor and connection. This means that each program to be loaded on to a networked transputer has to be developed and tested on the single-transputer system on the motherboard without communication links to others, and a configuration of the network has to be designed and developed manually, without a tool on the TDS.

The current version of the TDS restricts access to the PC filing system to one reading or one writing task only. This means, for example, that programs requiring simultaneous file inputs with outputs cannot be implemented. Also, the TDS is not equipped with a graphic interface or library. It may, however, be possible to build the interface by means of manipulating the PC bus and the PC interrupts, but this has not been investigated.

2.3. Occam

2.3.1. Background

Transputers are designed to implement the Occam language but also support other languages; C, Parallel C, Pascal and FORTRAN, that may produce larger codes than Occam. Occam is a simple language based on the Communication Sequential Processes [CSP] model of concurrency and communication, and is a message-oriented language where the basic unit is called a "process". Occam's processes communicate using "channels", inter-process data paths that provide a zero-buffered, uni-directional data path between two processes running in parallel. A channel can be placed between two transputers; processes are thus not restricted to the same transputer, allowing both internal and external parallelism.

Traditional computers are designed for the sequential execution of programs. A sequential programming language is thus characterised by its actions occurring in a strict, single execution sequence. A parallel program, however, may consist of a number of processes that themselves are purely sequential, but which are executed concurrently. A parallel programming language is required to handle a number of concurrent processes that may communicate with each other to share variables or to synchronise the processes.

Occam processes are built from three primitive processes:

assignment `a := b`

where "a" and "b" are variables.

input `a ? b`

where "a" is an input channel, and "b" is a variable.

output `a ! b`

where "a" is an output channel, and, "b" is a variable.

These are combined to form constructs: "SEQ"; sequential, "PAR"; parallel, "ALT"; alternative, "IF"; conditional and "WHILE"; iteration.

2.3.2. Structure of Occam

The "SEQ" construct signifies that statements under process are to be executed sequentially, as in traditional sequential languages. The "PAR" construct denotes that the following processes are to be executed independently, but, concurrently.

```
PAR
  process A
  process B
  process C
```

List 2.3.1.: Example of PAR Structure.

The processes under a "PAR" construct are supposed to be given equal time slots, however, that depends upon the compiler's best decision and load balancing of the concurrent processes. In some cases, concurrent processes may actually require more execution time than a combined sequential process, unless some background overheads, such as communication with an external processor, are involved. It thus does not necessarily follow that a parallel processing configuration will always be more efficient than a sequential equivalent. When arranged in a suitably efficient manner, the distribution of tasks is highly condition-dependent,

and therefore requires considerable programming skills. It is also worth noting that each process placed in parallel should have as similar loading as possible, otherwise the program sequence has to wait until all the main processes have been executed.

In the case where one process has to be assigned more time slots than another, the "PRI PAR" structure may be used, providing the construct only involves two processes.

```
PRI PAR
  process A
  process B
```

List 2.3.2.: Example of PRI PAR Structure.

In the above example, priority, or allocation of more time slots, will be given to process A. In some extreme cases, this will mean that no time slots can be allocated to process B, if process A is heavily loaded. Alternatively, if time slot is given to the lower priority process, incoming communications to the higher priority process may be ignored, since the T800 transputers are time-multiplex parallel processors without any hardware buffer on their communication links. A solution to this is a "software" buffer.

Another usage of the "PRI PAR" statement is for higher resolution clock control. In a low-priority process, a "TIMER" input provides a 15.625 kHz clock; 1 tick = 64 µsec. When the TIMER function is used in a prioritised process, the clock frequency changes to 1 MHz; 1 tick = 1 µsec. To avoid

loss of time slots, the "SKIP" statement is usually placed as a lower priority process that literally does nothing and therefore consumes no time slot.

The "ALT" construct operates a "first-come-first-served" style switching procedure.

```
ALT
  channel A ? x
    process A
  channel B ? x
    process B
  TRUE
    process C
```

List 2.3.3.: Example of ALT Structure.

In the above example, when the input to channel A comes first, process A will be executed. If there are no inputs to either channel A or channel B, process C will be executed. However, as long as one process under an "ALT" structure is being executed, the "ALT" may not observe another channel interrupt. This may result in a loss of the incoming data, since the T800s have no hardware buffer on their communication links, and the Occam does not place a communication buffer automatically. To prevent the loss of incoming data, a buffering process for each channel may be required, depending upon the frequency of interruption.

The Occam text books always recommend the use of a "TRUE" guard whenever an "ALT" structure is placed as a safety precaution or a time-out facility, since the absence of a "TRUE" may lead to a deadlock of the program; an infinity loop accidentally caused by the compiler. Indentation in the list above signifies the level of the nested process.

A variable or a communication channel cannot be shared among the concurrent processes placed under a "PAR" construct that necessitate the usage of an "ALT" structure. For example, consider the case where there are two processes placed in parallel, and both results have to be displayed on the terminal screen. If the "TRUE" guard is fired when no input from either process is available, the "SKIP" process will be executed.

```
PAR
  SEQ
    process A
    channel A ! result A
  SEQ
    process B
    channel B ! result B
  SEQ
    ALT
      channel A ? x
        screen ! a, x
      channel B ? x
        screen ! b, x
      TRUE
        SKIP
```

List 2.3.4.: Example of Channel Sharing.

As in the "PAR" construct, a "PRI" can be applied over the "ALT" to provide more attention to information from an input channel. In this case, there can be more than two input channels under a "PRI ALT" structure. Priority is given to the input channel immediately below an "PRI ALT" statement.

In a sequential program, there is only one execution path, which is relatively straightforward to terminate. Concurrent programs, however, may have many execution paths. If a path is not terminated in the correct manner, it may lead other processes to a deadlock situation.

When two processes are placed under a "PAR" structure, both processes must be terminated within a "good" timing and in a proper manner. If one process is ended far earlier than the other, the remaining process is executed with the same time slot allocation, as if the process already ended was still running. This means that load balancing between the concurrent processes is a very important consideration. A possible solution to synchronising the termination of two or more concurrent processes is to employ a global variable as a flag to signal termination. Since Occam does not allow the usage of shared variables, the termination information should be sent through a communication channel.

Occam also has similar repetitive and conditional features in common with other conventional computer languages, such as "IF", "FOR" loop, "WHILE" loop and "SWITCH" case switch. The "WHILE" loop is often used for controlling concurrent processes.

In a typical development procedure, an Occam program may be tested on a single transputer, as an "EXE" file. The ultimate goal, however, is usually to map a number of programs onto inter-connected transputers, by means of a "PLACED PAR" statement in a "PROGRAM" file. The number of processors does not always match the number of processes, since more than one process can be situated on a processor under a "PAR" structure.

A "PLACED PAR" statement is followed by at lest one placement statement allocating a specific processor for the execution of the process that follows, the hardware communication links to be used and where they are to be connected, and a "PROC"; procedure for the processor. A "PROC" declaration is followed by the name of the procedure and the necessary communication links and given variables. The main body of a "PROC" declaration, called the procedure body, consists of one or more "SC" source code instructions.

```
SC extra
SC mouse

[3]CHAN OF ANY s.out:

PLACED PAR
  PROCESSOR 0 T8    -- T800-17 128k
    PLACE s.out[0] AT link0in:
    PLACE s.out[1] AT link1out:
    extra(s.out[0], s.out[1])

  PROCESSOR 1 T8    -- T800-17 128k
    PLACE s.out[1] AT link1in:
    PLACE s.out[2] AT link0out:
    mouse(s.out[1], s.out[2])
```

List 2.3.5.: Example of "PROG" File.

One or more "SC" codes for the processes under a "PLACED PAR" have to be included in a "PROGRAM" file by means of the source code themselves or a "#USE" statement for the inclusion of a function library; this is similar to the "#include" statement in C-language.

The numbering of the processors has to be matched with the booting path of the processors: TDS understands that the lowest numbered processor is the first to be booted, and it then follows the hardware links specified. This means that there should be a path that covers all the processors, and

it is sometimes necessary to introduce a dummy communication link to bridge a gap, only for the purposes of booting.

2.3.3. Variable Types

Occam provides the following elementary data types:

BOOL	Boolean logic value; TRUE or FALSE.
BYTE	integer value between 0 and 255.
INT, INT64, INT32, and INT16	signed integer values; default [32-bit in T800], 64-bit, 32-bit and 16-bit.
REAL64 and REAL32	floating point values; 64-bit and 32-bit.

The "RETYPE" statement provides a quick type conversion between different types of INT [INT64, INT32, INT16] values or from an INT value to a BYTE array of four. Traditional type conversions are also available between any type of variables, but this takes more time slots than a "RETYPE".

The 32-bit based T800 transputer is capable of operations in boolean, 8-bit [BYTE], 32-bit and 64-bit [double precision]. In the case of 16-bit, however, the INT16 variables are handled as 32-bit integer types [INT32 or INT] with zeros in the upper bytes. This means that whenever an operation using INT16 type is executed, the type conversions from INT16 to INT32 and from INT32 to INT16 are performed internally, requiring more time slots than an operation using INT32 types.

2.3.4. Communication Channels

Communication between processes [or processors] is achieved by means of channels. Occam communication is point-to-point, synchronised and unbuffered. For this reason, a channel needs no process queue, no message queue and no message buffer. For the same reason, however, high density communications placed in parallel may cause loss of the data, by overwriting or lack of a time slot, and a deadlock situation, by waiting for an acknowledgement signal [for synchronisation] which can never be sent.

A channel communication between two processes on the same transputer is implemented by a single word in memory whereas a channel between the processes executing on different transputers is created by point-to-point links using two signal wires that provide two Occam channels, one in each direction. In the case of the latter, as the T800 transputers do not have a hardware buffer for the communication links, if a channel is placed between two processes in a different priority, data loss may occur on the receiver side unless a buffering process is employed. For communication with non-transputer-family devices, an external link adapter, IMS C011 or IMS C012, acts as an interface and a dummy transputer.

The link protocol provides the synchronised communication of Occam that is based on "message-and-acknowledgement" protocol. An inter-transputer link sometimes causes a local deadlock; when the receiver side is busy and fails to send an acknowledgement to the sender, this lack of acknowledgement freezes the sender process until the acknowledge

signal comes. This means that load-balancing between the processors is also important.

2.4. 160 Transputer Network

A prototype architecture for a transputer network, using sixteen T800s, was presented and demonstrated at the 1990 ICMC held in Glasgow (Bailey, et al. 1990). This was developed into a distributed parallel audio processor using 160 transputers inter-connected as a ternary tree. The network has been used as a test-bed for network architecture for real-time synthesis.

Each T800 transputer has four hardware communication ports that permit the construction of a ternary tree, providing hierarchical control. While a tree structure provides short path lengths between the arbitrary nodes, the modified ternary tree can also achieve this between siblings at the same level (Bailey 1992), thus increasing the scope for achieving both efficiency and flexibility in the flow of data between transputers.

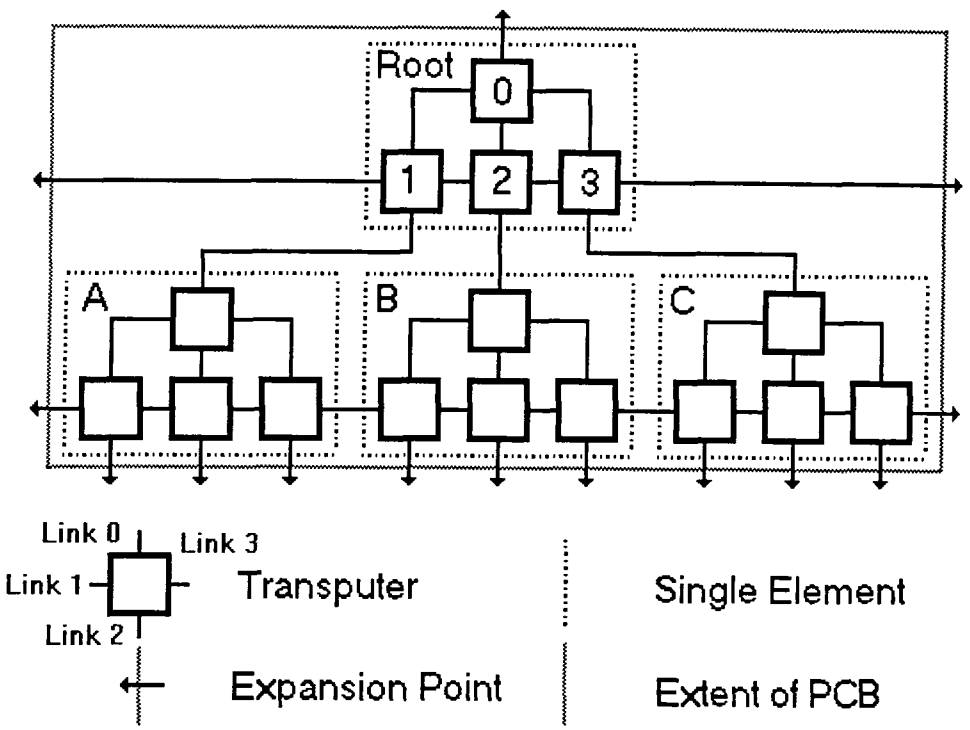


Figure 2.4.1.: Basic Topology of the Transputer Network.

Four of these single elements, a total of 16 transputers, are accommodated onto a standard 3U printed circuit board, as in the prototype architecture published in 1990 (Bailey, et al. 1990). The transputers are hard-wired to each other permanently. The software configuration of the network, however, is flexible and re-programmable.

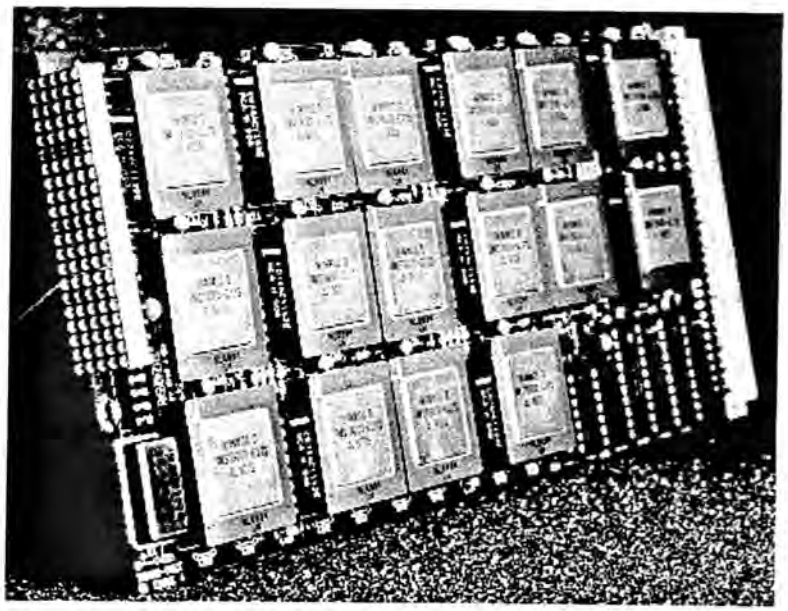


Figure 2.4.2.: 16-Transputer Network on a PCB.

A block of LEDs is situated on the edge of each PCB to monitor the status of error flags and the activity of the links number one, two and three of the transputers. The organisation of the LEDs is shown in the diagram below. This is the only way to monitor the performance of the transputer, as there is not a CPU performance measurement probe situated on a T800. In the case of the latest T9000 transputer, there is a fifth link designed especially for this purpose.

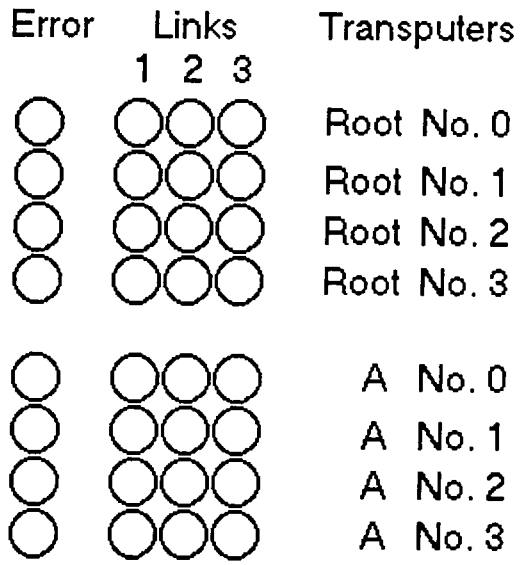


Figure 2.4.3.: Monitoring LEDs on a PCB (part).

An unusual feature of the design is the absence of any external memory local to the board. The rationale behind this is that a real-time distributed system should not require a large amount of on-board memory for intermediate data storage. It might be said that this expectation was a little idealistic. As a number of applications, described in the later Chapters, have proved, this limited storage size had to be overcome by design modifications at a later stage.

Each transputer thus uses only its internal 4k-byte memory for programming purposes. This results in a total of 160 transputers with 640k bytes of internal memory distributed across the network of our ten-board system, and a maximum processing power of 1,400 MIPS. The absence of local external memory necessitates compact algorithms for execution at audio sampling rates and the use of a compact code that leads to Occam.

One of the primary objectives is the use of many small and cheap processors to increase the overall computing power in a cost-effective manner. In other words, individual processors in such a network need not offer exceptional performance since the allocated sub-tasks are significantly smaller than the complete task when executed on a single processor system.

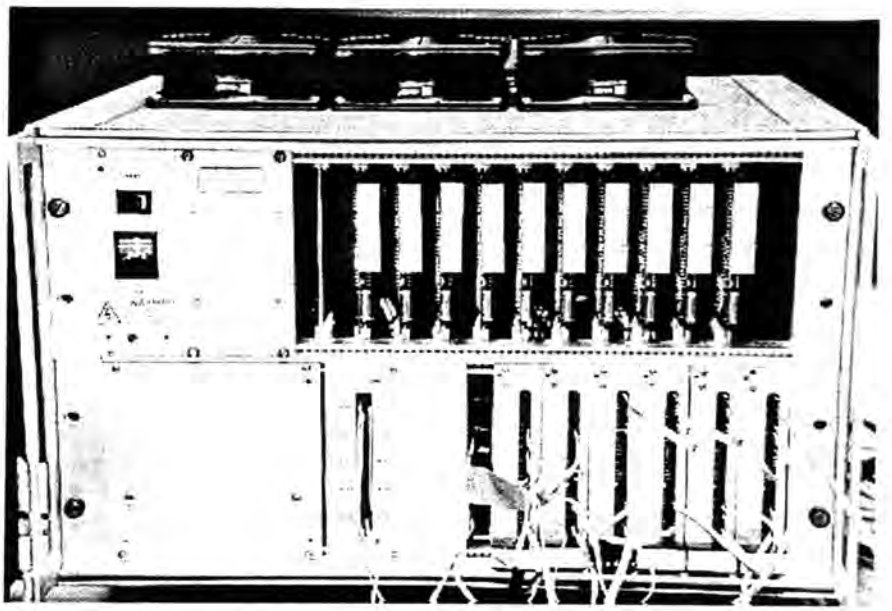


Figure 2.4.4.: Overview of The 160 Transputer Network with six external Transputers.

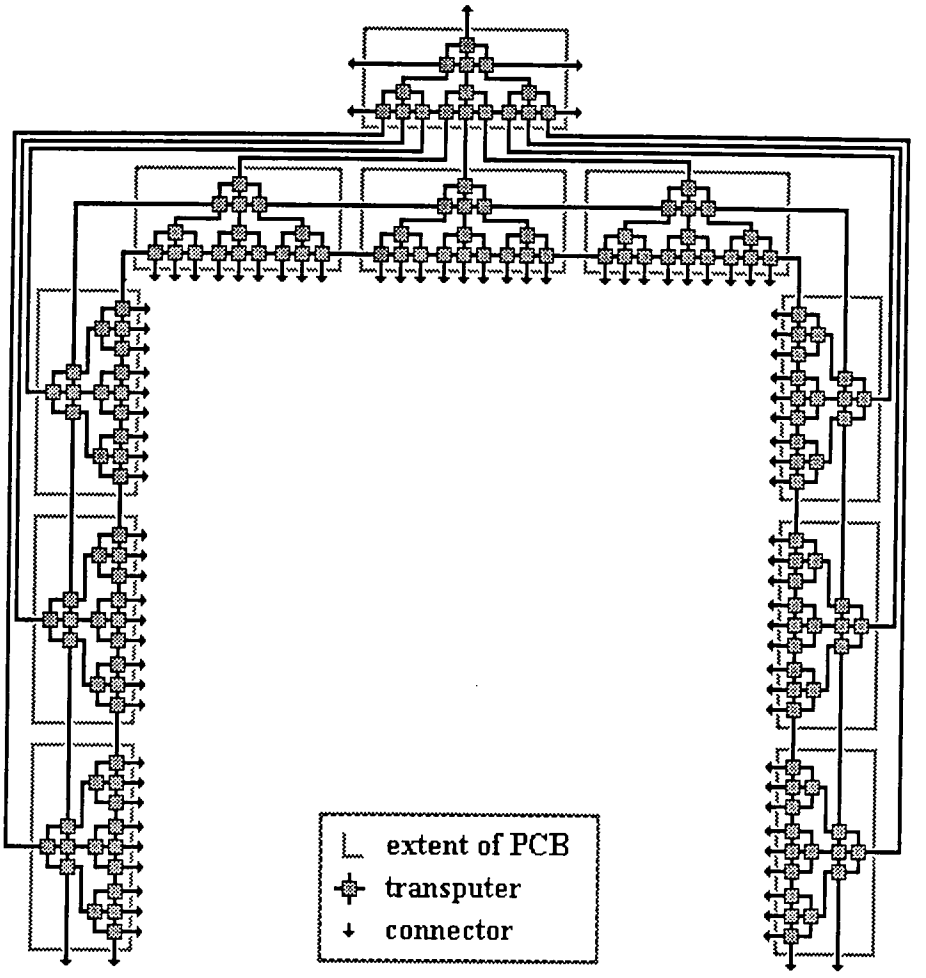


Figure 2.4.5.: Connection Diagram of The 160 Transputer Network.

2.5. Summary

In this chapter, the capabilities and the limitations of the transputer, its software development system [TDS] and the programming language designed for the transputers [Occam] have been described. Transputers are time-multiplex parallel processors with non-buffered communication links that allow high-speed communication between processors, and also the development of software and hardware in a variety of configurations.

These resources offer considerable opportunity for the construction of parallel and distributed networks which can be customised to the specific requirements of audio synthesis, and this proposition is tested in practical terms in the chapters which follow. Connectivity, however, does not automatically solve in itself some functional problems of process control in any system which makes significant use of parallel and distributed architecture, and solving these is critical to the successful implementation of any such algorithm whether for audio or any other computational process.

The most fundamental of these concerns are the possibility of data loss in the buffer-less communication hardware, which may require the introduction of software buffers, and also the ever-present risk of timing errors which may lead to problems of local deadlock and thence ultimately system failure. Both considerations are described further in the subsequent discussions of the various implementations which form the core of the research project.

By way of further background, the essential hardware characteristics of the 160 Transputer Network used for these investigations are also described, as there will ultimately dictate what can and cannot be achieved in software terms. It has also been noted in these constraints that the software development system [TDS] imposes some important limitations on essential communications, some of which will prove significant, as will be described in due course.

Chapter 3. Measurements of Keyboard Performance through MIDI

3.1. Hypotheses

These tests are based on a question that is frequently asked by professional musicians: is MIDI really fast enough to represent accurately a piano performance played by a skilled executant? Such a proposition raises a fundamental question from an engineers' point of view: is the MIDI standard suitable as a real-time controller for contemporary multi-voice and multi-device systems? When MIDI was standardised almost fifteen years ago, user expectations, in terms of digital signal processing devices and means of control, were constrained by the limitations of the hardware available at the time, such as 8 MHz clocked eight-bit processors with 64k-byte memory. Over the past decade, due to the rapid evolution in the speed of modern computing technology, the restrictions of such components are now open to review. The question will be discussed by comparing the limitations of MIDI technology, using digital signal processors and their technology, and their perceived characteristics on the human side with specific reference to the target applications.

The MIDI specification is fifteen-year-old technology that was designed for an eight-bit Z80 standard, some orders of magnitude less than that of a modern PC. But, is it too slow to transmit musical information? Moore made some assumptions in his article (Moore 1988) five years after the standardisation. In this article, the following assumptions concerning MIDI concepts were stated:

> One of the fundamental assumptions of the MIDI concept is that these small delays introduced by serial transmission are either imperceptible or -- if not exactly imperceptible -- that they don't make any difference in a musical context... The first dysfunction results from the fact that in some musical situations, millisecond delays do matter... A second problem with the MIDI assumption is that there is an unpredictable amount of delay between the time a performance gesture occurs and the time it is communicated to the synthesizer... MIDI is designed to report on musical events in a timely manner... It is known, however, that even small amount of "sample jitter" can degrade a digital recording significantly...
>
> F. R. Moore (1988) "The Dysfunctions of MIDI"

I have some doubts about these claims. For example, in the case of the first argument concerning small delays, if there is a few milli-seconds of "constant" delay, does it matter in most musical situations? How significant in practice are any "special cases"? I would agree that there are some buffers on a MIDI network communication; an opto-isolator between a MIDI-IN and a MIDI-THRU, between a MIDI-IN and a synthesiser, and between a controller and a MIDI-OUT. All of these cause incremented delays, but these are constant for any given condition.

As to the second claim, an "unpredictable amount" of delay, unfortunately, the author has not cited an example of it. I would assume that it can be quantified to a certain degree, and might be regarded as a small amount of "constant" delay. Such disagreements require a careful analysis.

I conducted a preliminary experiment of human auditory response to different delay conditions described in the next section. The possibility of unpredictable delay must be considered on the hardware side, such as variations in the response of a MIDI key sensor. For example, in the case of simultaneous events, like a multi-note-chord, information has to be accumulated in time, and then sent serially via the communication protocol of MIDI. This issue will be examined later in this chapter.

I totally agree with his third claim. The MIDI standard is clearly not suitable for "sample oriented" tasks, since MIDI is an "event oriented" standard. Other representations, such as those used in a software synthesis program such as CSOUND (Vercoe 1986), can be more suitable for "sample oriented" communication. Due to the difference of the concepts, however, discussion of this particular claim is not relevant to this project, because the matter that is concern of this thesis is event-oriented communication. As well as the "event oriented" characteristics, I would like to add another limitation; keyboard-oriented nature of MIDI: the MIDI standard is basically specified to represent synthesiser performances driven from keyboards, and to transmit this information, rather than the more complex characteristics of orchestral instruments.

Hence the information transmitted over MIDI should ideally be limited to simple on-off key-stroke actions, perhaps with the additions of key pressure information to control amplitude. In other words, information other than basic key actions will invariably require a higher bandwidth for transmission. In this measurement, therefore, the target machine should

be a MIDI keyboard, and so, some MIDI information may thus be ignored as irrelevant to the application; for instance, pitch bend [strings origin] and after touch [extra controller]. Under these conditions, I would like to begin my discussion: is MIDI fast enough?

My hypotheses are:

1) The MIDI standard may be still fast enough to represent a keyboard performance without an extra controller.

2) If the performance is extended to include some accompaniments, MIDI can be fast enough, provided the transmission delay is constant and is small enough to be ignored by listeners. Some claims about the "dysfunction of MIDI" can therefore be attributed to their "misuse" of MIDI or their "excessive demands" upon MIDI.

3.2. Measurements

3.2.1. Distorted Chord

Some articles indicate that the minimum resolution of timing in the human auditory systems between 10 and 30 msec, depending upon the listener and some context-dependent conditions, including the pitch [frequency] and the amplitude of the sound.

Gabor (1946) examined the "threshold information sensitivity" of the ear, using 500 Hz and 1,000 Hz sine waves. He concluded that for the most critical of listeners, a 10 msec sound could be recognised as a "tone", rather than a "click" or "noise", but at least 21 msec was required to recognise its "pitch".

Green (1971) also postulated that about 25 msec was needed to distinguish differences in starting times between high- and low frequency "sinusoids". To distinguish between a "click" or "two clicks", however, the limit of temporal resolution was invariably about 1 or 2 msec. In Rasch's examination (Rasch 1978) of on-set times of high and low tones, with 20 msec-long rising and decaying ramps, the results showed that a time interval of 10 to 15 msec seems to be the largest detectable interval.

These claims suggest that a constant small amount of delay, less than 10 msec, should not be detectable, and that the limit of temporal resolution is 1 or 2 msec, whereas MIDI can send a note information in less than 1 msec.

I conducted an experiment designed to determine the minimum distinguishable timing in the human auditory response, and to discover a suitable resolution in timing for MIDI transmission. A sequencer software program on a PC was used together with a MIDI keyboard to produce sounds. A four-note-chord was played in an *arpeggio* in six varieties; in different sequences [upward/downward] and at different intervals [10, 20 and 30 msec] for each component. Among the distorted chords, a genuine chord [no interval between each component] was played twice. In this experiment, four subjects, who happened to work in our laboratory, sat near two loud speakers, at an approximate distance of 3 metres. Two of these listeners were well-trained musicians, and the other two had strong interests in music. The listeners were asked to judge whether the chord was played as an upward *arpeggio,* a downward *arpeggio* or a plain chord. They were not informed of the order of the chords that were selected at random. To avoid possible external psychological effects, a series of chords were played twice in different orders. The results are shown in Tables 3.2.1. and 3.2.2.

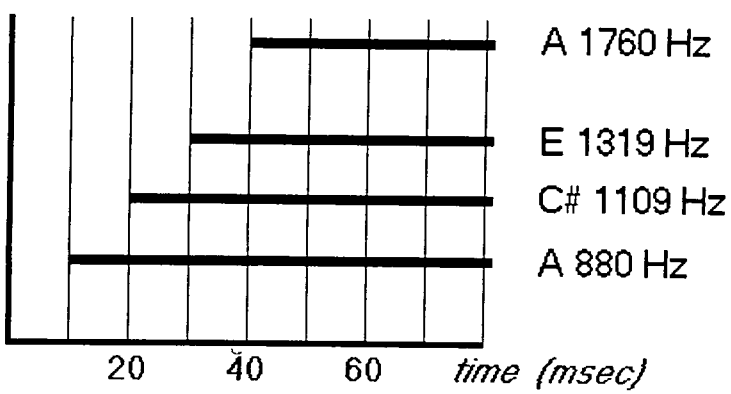


Figure 3.2.1.: Timing of the Distorted Chord (10 msec, Rise).

		direction and interval (msec)							
		down 30	down 20	down 10	even (1)	even (2)	up 10	up 20	up 30
subjects	pe	↓	↑	↓↑	↓↑	↓	↓↑	↓	↑
	pa	↓	↓	↓↑	↓↑	↓↑	?	↓	↓
	a	↓	↓	↓	↓↑	↓	↑	↑	↑
	t	↓	↑	↓↑	↓	↓↑	↑	↓	↑

down ↓ even 0 can't judge ?
rise ↑ nor even ↓↑

Table 3.2.1.: Listening Experiment (1) (Slightly Distorted Four Note Chord).

		direction and interval (msec)							
		down 30	down 20	down 10	even (1)	even (2)	rise 10	up 20	up 30
subjects	pe	↓	↓	↓	↓↑	↓↑	↓↑	↑	↑
	pa	↓	↓	↓↑	↓↑	↓↑	↓↑	↑	↑
	a	↓	↑	↓	↓	↓↑	↓↑	↑	↓
	t	↓	↓	↑	↑	↓↑	↓↑	↓	↑

Table 3.2.2.: Listening Experiment (2) (Slightly Distorted Four Note Chord).

The above results suggest that more than half the listeners could distinguish the sequence of *arpeggi*, upwards or downwards, in the case of 20 and 30 msec intervals. Due to the small number of candidates, these results can only be regarded as a preliminary experiment and further experiments are therefore required for a more authoritative analysis.

3.2.2. Timing in Piano Performance

Using a digital signal processing tool on an Apple-Macintosh, another preliminary experiment was conducted. I assumed that the quickest gesture in a piano performance is a "trill" or a "grace note". These two cases were examined as follows. The sound source was from a CD, performed by a well-trained executant. According to the Fourier Transform analysis on the frequency domain, there were three peaks; two strong peaks at the frequencies of "A flat" and "B flat", and the other at the difference of these two; beating frequency. On the time-amplitude domain, I could see the sound of the trill as a beating noise. The sound, however, could not be separated into a single tone; one of "A flat" or "B flat". This means that due to natural acoustical conditions, notably echo, the original sound was distorted.

As a result of the first experiment, to avoid echo and other effects, a digital audio tape recorder was used with a microphone situated immediately above the piano strings. Some "trills" and "grace notes" of a piano performance were played by a music student as a sound sample. These were then analysed in the same manner as for the CD sound. On an analysis, the onset of the notes was clearly identifiable; ranging between about 5 and 10 msec. This suggests a minimum timing in the case of a piano performance that lies somewhere between these limits, and arguably towards the higher of the two figures. It seems reasonable to presume that about 10 msec of constant delay would not be distinguished by most listeners, and therefore would not effect a piano performance in most cases.

3.2.3. The Quickest Gesture in Keyboard Performance

Along with those fundamentals shown before, Moore (1988) made another assumption concerning capturing musical gestures from keyboard performances; resolution and rapidity.

> Assuming that there are 88 notes on a keyboard, we might start by considering how quickly notes can be played by a skilled pianist. I am able to play a glissando across all 88 notes of a piano in about half a second with one hand. This means... about 176 events per second.
>
> F. R. Moore (1988) "The Dysfunctions of MIDI"

This is an extreme, or an unrealistic example, but it could be acceptable as a part of a modern piece. Another example using a different method is:

> The fronts of seven piano keys were filmed with a Hicam camera at 2000 frames/s, yielding a visual record of about 1.5 seconds of playing. On one film, the pianist was playing C to G back to C as quickly, loudly and evenly as possible. He made about 13 key presses per second with his right had.
>
> C. L. MacKenzie (1985)
> "Structural Constraints on Timing in Human Finger Movements"

This example is more realistic and is a controlled performance. I tried to perform both cases on a MIDI keyboard, that had 60 non-weighted keys, and to record the information, using the MIDI-to-transputer interface. In the musical context, a *glissando* usually means that using some fingers, white keys on a keyboard are pressed sequentially from low notes to high [or *vice versa*] continuously. In Moore's assumption, however, he argued that the measurement should be a "*glissando* over the 88 piano keys".

Since there are only 52 white keys on a conventional acoustic piano, I interpreted and performed a gesture that placed some fingers on an initial black key and the thumb on an adjacent white key, followed by a hand movement from left to right.

Performance	Gliss 1	Gliss 2
Length (sec)	4.716	0.9046
MIDI Events		
note on	234	59
note off	234	59
status byte	27	6
total (bytes)	955	242
Transmission Rate		
bits (baud)	2025	2675
keys (keys/s)	48.77	65.22
Interval of Notes		
average (msec)	20.19	15.3
standard deviation	12.26	10.45
minimum (msec)	4.027	4.864
Duration of Notes		
average (msec)	85.29	72.16
standard deviation	22.34	27.14
minimum (msec)	18.94	16.64

Table 3.2.3.: Quick Gesture in Keyboard Performance. (Glissando through a keyboard)

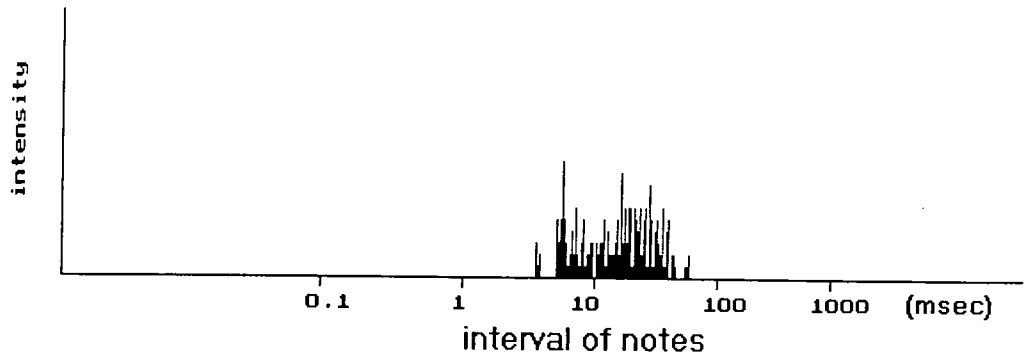


Figure 3.2.2.: Quick Gesture in Keyboard Performance (Glissando through a keyboard) [glissando 1].

In the first performance, because of a shorter range keyboard, I had to turn over my hand movement few times. This introduced some unexpected time intervals between the note progressions. To avoid that, only one movement was executed in the second. The graph above shows the interval as a histogram; it is written using a logarithmic scale for time and a linear scale for the intensity.

These results below show that the quickest gesture recorded in a piano performance is in the order of 10 to 100 msec. This means that a piano performer can just manage to play a 10 msec gesture.

Performance	scale
Length (sec)	41.17
MIDI Events	
note on	388
note off	387
status byte	229
total (bytes)	1779
Transmission Rate	
bits (baud)	432.2
keys (keys/s)	9.401
Interval of Notes	
average (msec)	106.2
standard deviation	24.85
minimum (msec)	60.06
Duration of Notes	
average (msec)	106.7
standard deviation	15.93
minimum (msec)	53.08

Table 3.2.4.: Quick Gesture in Keyboard Performance (C-G-C scale).

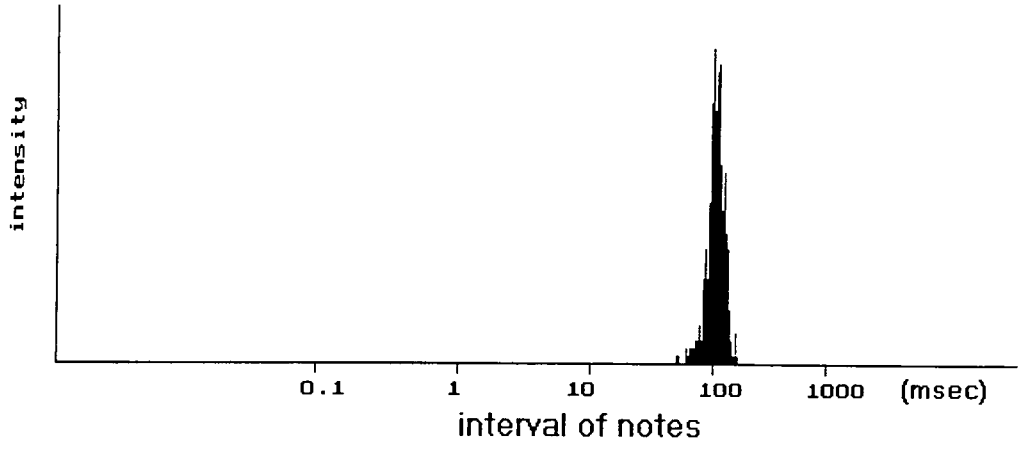


Figure 3.2.3.: Quick Gesture in Keyboard Performance (C-G-C scale).

3.2.4. The Busiest Gesture in Keyboard Performance

Moore claims:

> It is clear that MIDI transmission rate is just on the edge of this rough calculation for single notes. If, however, we consider the case of a piano chord in which a dozen or more keys are played simultaneously with two hands, we note that the time needed to transmit the data representing the note events is now about *N* msec (where *N* is the number of noted depressed)...
>
> F. R. Moore (1988) "The Dysfunctions of MIDI"

Transmitting a note event as a MIDI signal actually takes three bytes; one status byte, including channel information, with two data bytes, which are a key number and a magnitude of the event. Each takes 320 μsec; 960 μsec [about 1 msec] in total. In the case of the piano performance, however, the situation is slightly different: only one channel is used, mainly transmitting just "note on" and "note off" information, and, if required, the "running status". It means that the status byte, the controller and its channel information is not always necessary for each event, unless another controller, such as a pedal, is used. For that reason, the time needed to transmit the data representing *N* note events from a MIDI piano is less than *N* msec.

I tried to inspect another claim; a piano chord in which a dozen or more keys are pressed simultaneously with two hands. It seemed, however, to be impossible to play a conventional chord featuring more than five keys per hand, since I could not spread my hand more than one and a half octaves. For that reason, I tried to play a conventional four-note-C major

chord; C, E, G, C, with my right hand, and, then a five-note-G_7 chord; G, B, D, F, G, with both hands.

Performance	C major	G7
Length (sec)	30.66	3.513
MIDI Events		
note on	681	214
note off	631	161
status byte	167	20
total (bytes)	2791	770
Transmission Rate		
bits (baud)	910.4	2192
keys (keys/s)	20.58	45.83
Interval of Notes		
average (msec)	6.62	16.29
standard deviation	3.077	28.16
minimum (msec)	3.44	2.375
Duration of Notes		
average (msec)	71.54	63.54
standard deviation	39.34	23.22
minimum (msec)	38.3	3.379

Table 3.2.5.: Busy Gesture in Keyboard Performance (Eight Beat Chord).

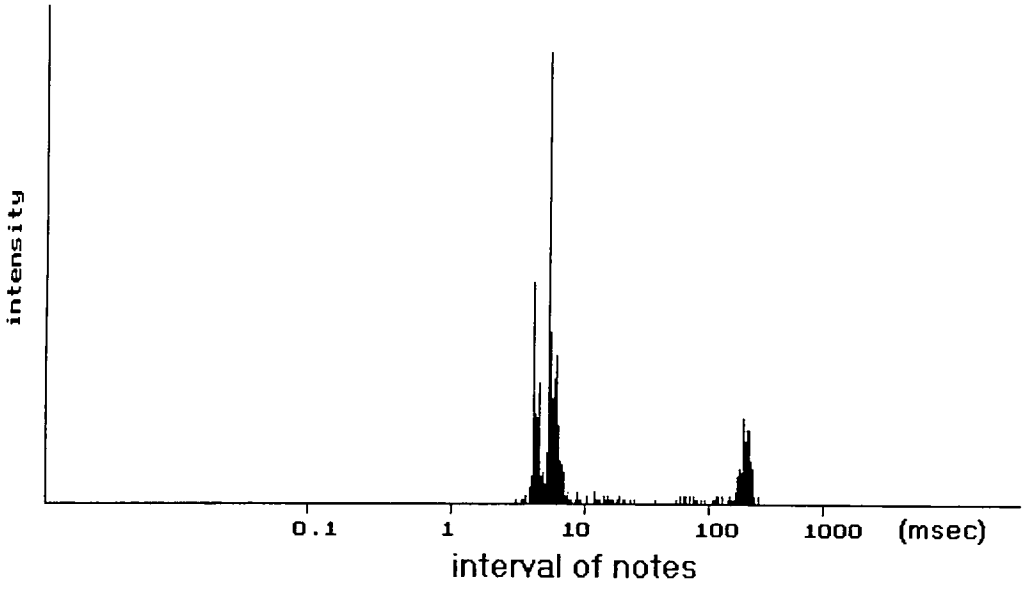


Figure 3.2.4.: Intervals of Notes (8 beat C maj).

Even when I played a ten-note-chord as quickly as possible, the transmission rate was not more than 2.2 KBaud. In the MIDI specification, the transmission rate is 31.25 KBaud on 16 channels; about 2 KBaud per channel in average. It means that if I try to send a ten-note-chord per channel multiplied through more than fourteen channels, it is possible to saturate the MIDI transmission.

However, do we need such kinds of densely packed information in the course of an ordinary performance? I believe that such a question is inevitably speculative and subjective to some degree, since the boundary between the classical ideas of music performance and the more experimental modes of creating music in a contemporary manner, which might prove complex enough to defect any conventional system, is ill-defined. For the purposes of this investigation, the proposition tested was that of a maximum requirement of eight channels to cover an expected degree of complexity in a rhythm part.

3.2.5. The Shortest Timing in Keyboard Performance

I attempted to examine two other examples of quick keyboard gestures; grace notes and trills, to measure the resolution of MIDI. Two performances were recorded: one played with B flats as grace notes, and Bs as main notes. The other one was done with C sharps, as grace notes, and Bs, as main notes. The result shows that the resolution of MIDI transmission is fast enough to send a ten-msec-long grace note.

Performance	grace 1	grace 2
Length (sec)	20.61	19.35
MIDI Events		
note on	182	192
note off	182	191
status byte	113	107
total (bytes)	841	873
Transmission Rate		
bits (baud)	408	451.1
keys (keys/s)	8.83	9.869
Interval of Notes		
average (msec)	113.3	101.3
standard deviation	94.83	78.94
minimum (msec)	8.861	10.6
Interval of Main Notes		
and Grace Notes		
average (msec)	19.8	20.26
standard deviation	5.852	10.6
Duration of Notes		
average (msec)	33.14	35.33
standard deviation	16.78	17.78
minimum (msec)	11.9	10.24

Table 3.2.6.: Short Timing in Keyboard Performance (Grace notes).

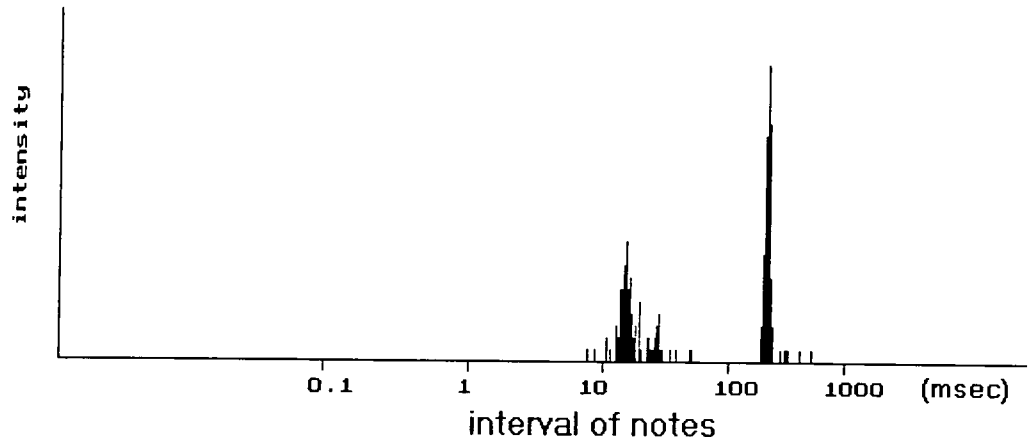


Figure 3.2.5.: Intervals of Notes (Grace Notes 1).

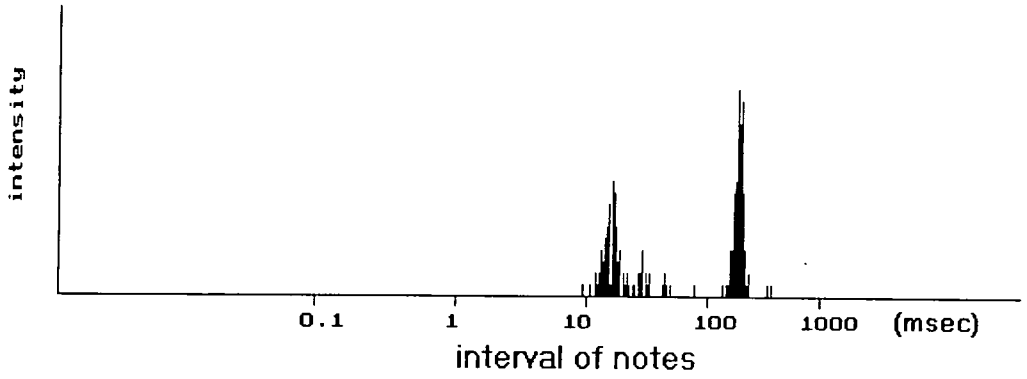


Figure 3.2.6.: Intervals of Notes (Grace Notes 2).

Two trill performances were recorded: one using an A flat key and a B flat, and the other using an F and an F sharp keys. These results suggested that the MIDI specification is fast enough to transmit an event of the order of milli-seconds.

Performance	trill 1	trill 2
Length (sec)	17.33	16.43
MIDI Events		
note on	188	188
note off	188	188
status byte	97	92
total (bytes)	849	844
Transmission Rate		
bits (baud)	489.8	513.6
keys (keys/s)	10.85	11.44
Interval of Notes		
average (msec)	92.22	87.38
standard deviation	14.07	11.77
minimum (msec)	5.305	42.97
Duration of Notes		
average (msec)	71.65	67.06
standard deviation	11.51	15.31
minimum (msec)	38.72	30.3

Table 3.2.7.: Quick Gesture in Keyboard Performance (Trill).

Figure 3.2.7.: Quick Gesture in Keyboard Performance (Trill 1).

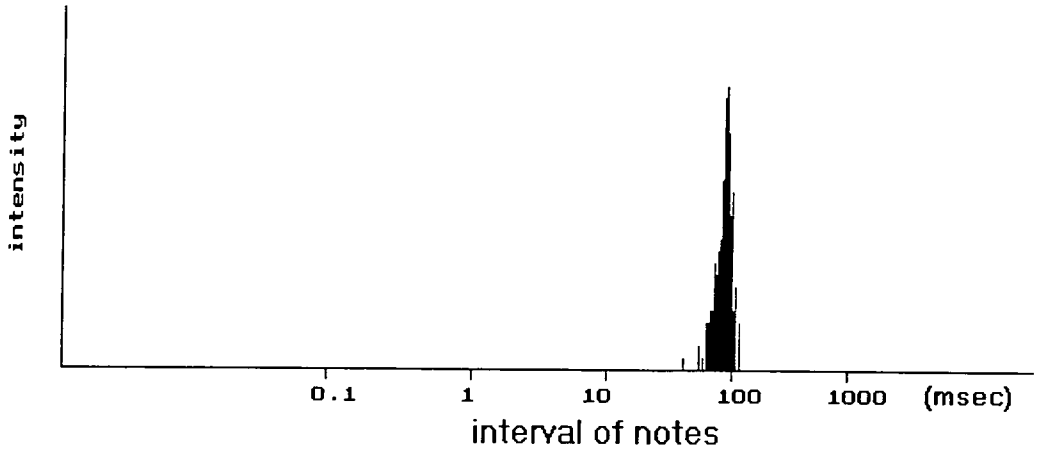


Figure 3.2.8.: Quick Gesture in Keyboard Performance (Trill 2).

From these experiments, I could presume that the transmission rate in the MIDI specification is fast enough to send several channels of conventional polyphony keyboard performance. To confirm this assumption, another endeavour has been carried out; a traffic analysis of MIDI keyboard performance.

This experiment is aimed measuring the traffic conditions on the MIDI communication line. The information was collected using the MIDI-to-Transputer interface. I performed a few tunes on a MIDI keyboard lasting about 30 seconds each. All the MIDI bytes were time-stamped and their intervals accumulated.

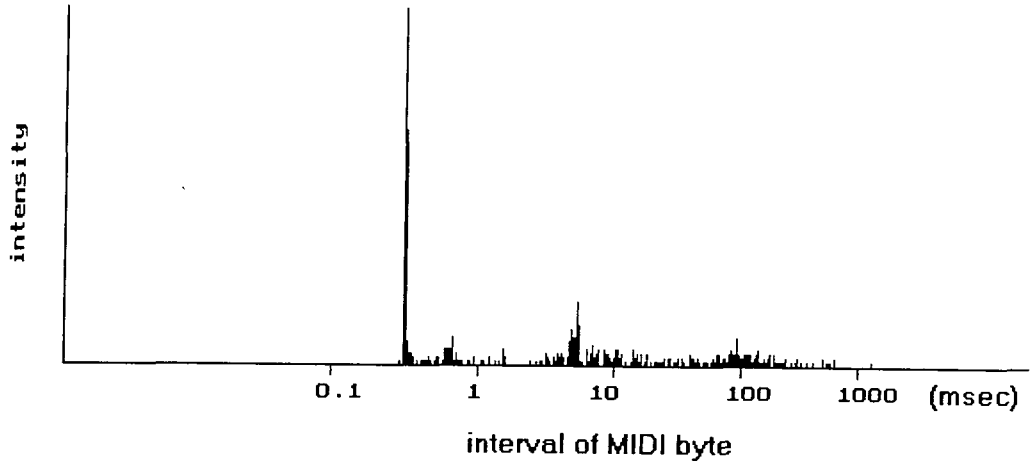


Figure 3.2.9.: Traffic analysis (tune 1).

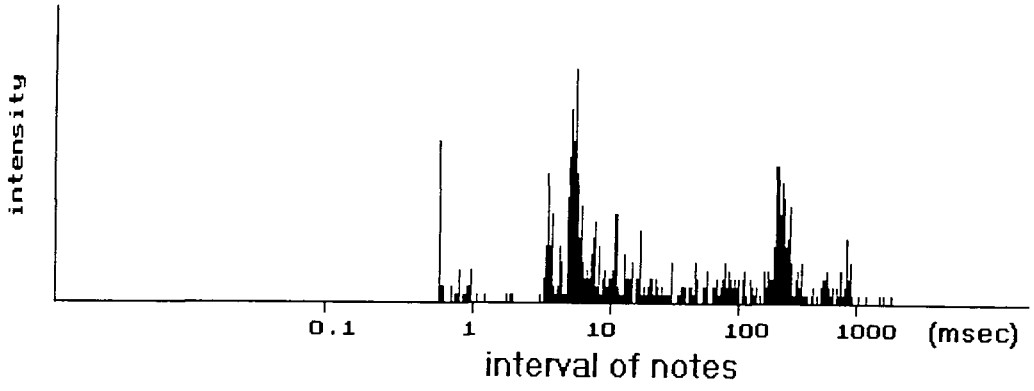


Figure 3.2.10.: Note Interval (tune 1).

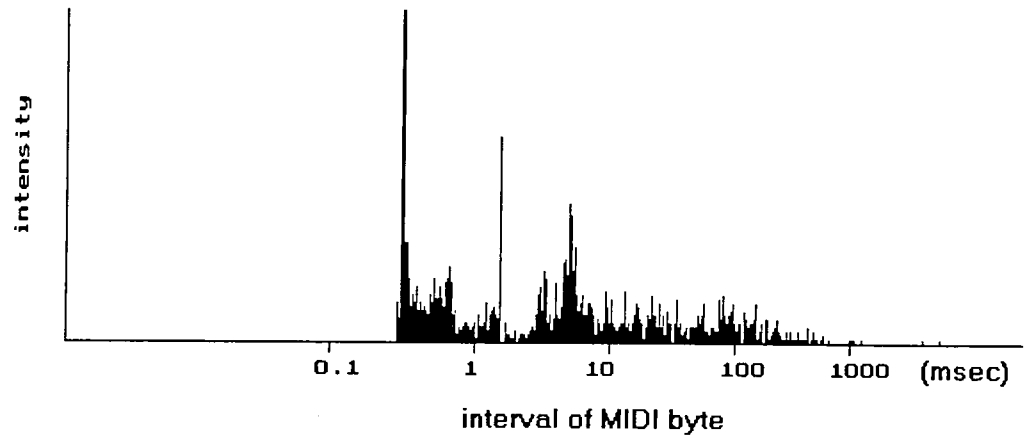


Figure 3.2.11.: Traffic analysis (tune 2).

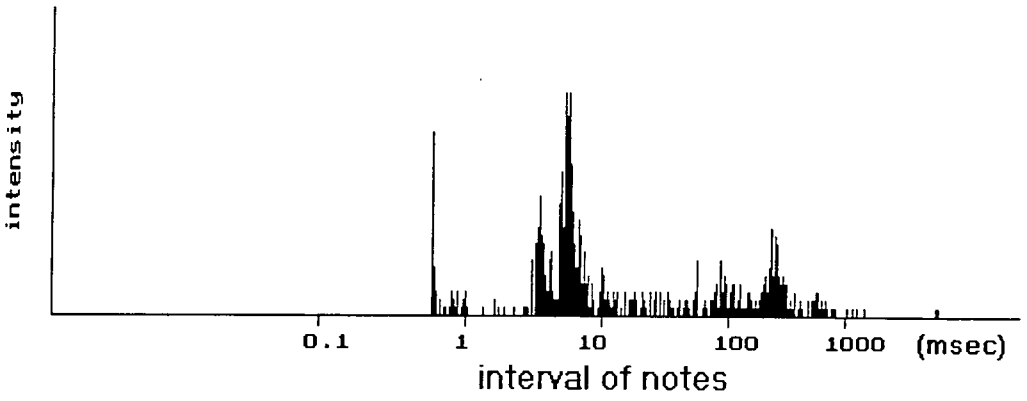


Figure 3.2.12.: Note Interval (tune 2).

As shown above, on each performance there is a strong peak at about 320 μsec, the shortest interval of the MIDI specification. Most of the intervals, however, were not concentrated near the strongest peak. This means that the MIDI connection is not always busy, with significant spare capacity.

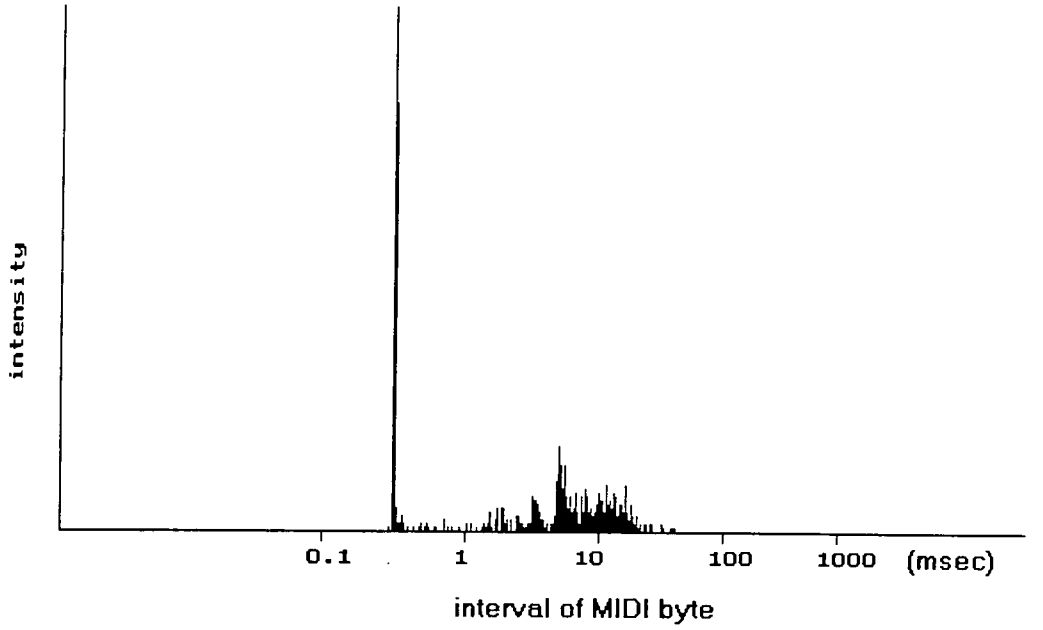


Figure 3.2.13.: Traffic analysis (glissando).

In the case of the busiest performance, indeed, the MIDI line had some spare capacity to carry more information. As a result, I can confirm that the MIDI specification is fast enough to transmit multiple keyboard performances on different multiplexed channels, despite the fact that the standard is based on fifteen-year-old technology.

3.3. Conclusion

In time resolution wise, the minimum time interval for a piano performance and the human auditory system is about ten milli-seconds, whereas the MIDI standard can manage to send a note event packet of information in less than one milli-second; about ten times faster than this requirement. In terms of transmission capacity, it seems to be difficult to saturate the MIDI transmission in terms of conventional keyboard performances without using other non-event-based controllers. From these experiments, the MIDI standard is shown to be fast and reliable enough for the recording and transmission of a single multi-keyboard performance when simple event-based instruments and controllers are used as the standard originally anticipated.

It would seem clear, however, that problems will arise when a number of different multi-keyboard performances are multiplexed together in a single stream of MIDI data to be distributed to a number of MIDI instruments. Hardware solutions to the problem described above have been developed in the form of sophisticated electronic hubs for the interconnection of multiple devices, using more than two MIDI data streams called "parallel MIDI", which provide networked solutions to such communication difficulties, and also solve another limitation; the maximum of sixteen channels per MIDI data stream.

3.4. *Aprés* MIDI

In 1994, about ten years after the publication of the MIDI specification, ZIPI (McMillen 1994) was presented as a replacement to the MIDI, a new language for describing music. There are some technical improvements over the MIDI standard that have been proposed, especially catching up with the latest computer and signal processing technology. When the MIDI specification was originally announced, eight-bit computers, such as Z80 or 8086, were common. Now a 32-bit PC has become the norm and a 64-bit PC is soon to be realised, the software should match with the new standards, in terms of word-length and speed.

Unlike the MIDI standard, ZIPI was led by academics without any strong backup from the electronic instrument manufacturers. This could be one of the reasons why a ZIPI equipped instrument has not yet materialised from the commercial sector. Or, the users of MIDI might be satisfied with this fifteen-year-old technology with some ad-hoc solutions, such as the usage of multiple MIDI cables controlled by a high-power PC. As I mentioned earlier in this chapter, in some extreme circumstances, especially in non-event-oriented cases, MIDI users may remain dissatisfied with its response characteristics. In general, however, it would appear that the MIDI specification is still acceptable to the majority of its users.

Chapter 4. Implementation of Real-Time Additive Synthesis on the Network

4.1. Sine Oscillation Method

4.1.1. Wavetable (Table Look-up)

This method has been reviewed in Chapter 1.3.3. Its implementation in Occam is as follows:

```
fq:         output frequency
sf:         sampling frequency   [constant]
tb:         table size           [constant]
amp:        amplitude
incl:       inclement
ag:         angle
tp:         temporary storage [INT]
out:        output channel
table[]:    wavetable

-- set up
   incl := (fq * tb) / sf
   ag := 0.0 (REAL32)

-- cycle
   ag:= ag + incl
   IF
      (ag > tb)
        ag := ag - tb
      else
        SKIP
   tp := INT ROUND ag
   out ! (table[tp] * amp)
```

List 4.1.1.: Sine Oscillator by Wavetable Method.

In the above implementation, the initial frequency set up needs 28 clock cycles [1.4 µsec at a 20 MHz clock], and each oscillator cycle does 32.4 clock cycles [1.62 µsec] whereas Occam's sine function requires about 17 µsec.

For an implementation of this method over the Transputer Network, however, there is a fundamental problem concerning the memory size, since each T800 transputer has only a 4k-byte on-chip memory. For a real-time implementation, the memory usage should be limited to less than 80% of the capacity, including the program code, to maintain a smooth computation. When the program code is 1k-byte, about a 2.2k-byte of memory space can be used for the wavetable, resulting in an 1100 sample-long table [about 34 msec at 32 kHz sampling rate] in 16-bit integer format or 550 sample-long [about 17 msec] in 32-bit format.

The size of the wavetable directly affects the quality of synthesised sound. The worst case signal-to-error noise ratio is given as 6 (k-1) dB, where the table size is 2^k sample-long (Moore 1977). In the case of the 32-bit wavetable above, the estimated worst case signal-to-error noise ratio is about -48 dB [k=9, 512-sample-long] representing in an unacceptable level of distortion. For the reasons above, this method is not suitable for an implementation over the Transputer Network.

4.1.2. Taylor Series Expansion

The Taylor series expansion method is a simple and elegant method of evaluating most functions.

$$\sin(x) = \sum_{m=1}^{\infty} \frac{-1^{(m-1)}}{(2m-1)!} x^{(2m-1)}$$

```
fq:     output frequency
sf:     sampling frequency  [constant]
pi:     2π                  [constant]
amp:    amplitude
incl:   inclement
ag:     angle
t0:     temporary storage (0)
t1:     temporary storage (1)
t2:     temporary storage (2)
t3:     temporary storage (3)  [INT]
t4:     temporary storage (4)
t5:     temporary storage (5)
out:    output channel

-- set up
   incl := (fq * pi) / sf
   ag := 0.0 (REAL32)

-- cycle
t1 := 0.0 (REAL32)
SEQ i=1 FOR 10
   SEQ
      t0:=REAL32 ROUND (i-1)
      t2:=POWER(-1.0(REAL32), t0)
      t3 := 1
      SEQ m=1 FOR (2*i)-1
         t3 := t3 * m
      t5 := REAL32 ROUND t3
      t4 := REAL32 ROUND ((2*i)-1)
      t1:=t1+(t2/t5)*POWER(ag, t4)
      out ! (t1 * amp)
      ag := ag + incl
```

List 4.1.2.: Sine Oscillator by Taylor Series Method.

The initialisation takes 28 clock cycles [1.4 µsec], and each cycle with up to 10 components 18849.3 clock cycles [942.5 µsec].

4.1.3. Polynomial Approximation

A minimal polynomial approximation presented by Hart et al. (1968) can also deliver a high quality sine and cosine function.

$$|x| = n\pi + f \qquad \text{where } |f| \leq \pi/2$$

$$\sin(x) = \text{sign}\,(x) \times \sin(f)\,(-1)\,n^2$$

$$\cos(x) = \sin(x + \pi/2)$$

The computation of sine or cosine involves three numerically distinct steps: the reduction of the given argument x to a related argument f, the evaluation of $\sin(f)$ over a small interval symmetrical about the origin, and the reconstruction of the desired function value from these results. The accuracy of the function values depends critically upon the accuracy of the argument reduction (Cordy 1980).

```
fq:              output frequency
sf:              sampling frequency   [constant]
pi:              2π                   [constant]
amp:             amplitude
ag:              angle
tp:              temporary storage    [INT]
Xwork, Rwork:    temporary storage
out:             output channel
VAL R IS [2.601903036E-6(REAL32),
          -1.980741872E-4(REAL32),
           8.333025139E-3(REAL32),
          -1.666665668E-1(REAL32)] :

-- set up
  ag := (pi * fq) / sf
  tp := INT ROUND (ag / (pi / 2.0 (REAL32)))
  XWork := ag - ((REAL32 ROUND tp) * pi)
  incl := ag
```

List 4.1.3a.: Polynomial Approximation.

The argument *x* has to be reduced to |π/2| which requires one "IF" trap with a few type conversions. This reduction process requires a large number of time slots, and therefore it is not suitable for a real-time implementation.

```
-- cycle
  tp := INT ROUND (ag / (pi / 2.0(REAL32))
  IF
    (tp > 2) OR (tp < -2)
      tp := tp MOD 2
    ELSE
      SKIP
  XWork := ag - ((REAL32 ROUND tp) * pi)
  IF
    ABS(XWork) > (pi /4.0(REAL32))
      IF
        XWork < 0.0(REAL32)
          RWork := - 1.0(REAL32)
        TRUE
          RWork := 1.0(REAL32)
    TRUE
      G := XWork * XWork :
      RWork := XWork +
(((((((R[0]*G)+R[1])*G)+R[2])*G)+R[3])*G)*XW
ork)
      IF
        (tp /\ 1) = 1
          RWork := - RWork
        TRUE
          SKIP
  out ! (RWork * amp)
  ag := XWork + incl
```

List 4.1.3b.: Polynomial Approximation.

4.1.4. CORDIC

The CORDIC [COdinate Rotation DIgital Computer] is an iterative arithmetic algorithm introduced by Volder (1959). With a coordinate rotation computation scheme, the CORDIC algorithm is a very efficient method for computing many elementary functions. During the 1970's, Hewlett-Packard incorporated a hardware implementation of thus algorithm for their desktop calculators, subsequently in the 1980's Intel began to use it for the computation of sine, cosine and other transcendental functions on their numeric processors. In our Music Technology Group, a CORDIC based digital sine generator was implemented in a 0.7-μm double metal CMOS process (Itagaki et al. 1996) and (Spanir 1998).

For the calculation of a sine or cosine of an angle θ, successive rotations of a radius vector are required on the unit circle starting at x=1, y=0 and ending at x= cos θ, y= sin θ. Rotating a two dimensional vector *[x,y]* by an angle ϕ, [counter-clockwise when $\phi > 0$, clockwise when $\phi < 0$] may be achieved by multiplying it by a matrix $\mathbf{R}_\phi$:

$$[x', y'] = \mathbf{R}_\phi[x,y]$$

$$R_\phi = \cos\phi \begin{vmatrix} 1 & -\tan\phi \\ \tan\phi & 1 \end{vmatrix} \qquad [1]$$

Rotating by angles ϕ then ω is equivalent to a rotation by angle of $\phi+\omega$: $\mathbf{R}_{\phi+\omega} = \mathbf{R}_\phi\mathbf{R}_\omega$. The radius vector is rotated by a series of angles, where the absolute values of which are $\phi_0, \phi_1, \ldots, \phi_n$ where $\phi_i = \tan^{-1} 2^{-i}$. The first three terms of this monotonically decreasing series are 0.785, 0.464

and 0.245. This choice of angle simplifies the arithmetic of multiplying by the corresponding rotation matrices. Their terms $\tan(\phi_i)$ become divisible by powers of two. Since, $\cos(\phi_i) = \cos(-\phi_i)$, the term:

$$K = \cos(\phi_0)\cos(\phi_1)...\cos(\phi_N)$$

can be grouped for all ϕ_i, and used as a constant multiplying scalar.

The popularity of the CORDIC algorithm was primarily due to its straightforward implementation on a fixed point device using only the arithmetic operations of addition, subtraction and binary right shift [division by powers of two]. In case of the T800 transputer, as a processor with a floating-point processing unit, there is no provision for "fixed" point operations. This means that for the implementation of the method on a T800, some of the fixed-point operations have to be substituted by less efficient floating-point operations. [For example, a bit-shift has to be replaced with a multiplication by 2.0.]

In addition, the CORDIC algorithm will only work in the first quadrant; between 0 and $\pi/2$ radian in the case of a sine oscillator. To implement this method as a continuous oscillator, it is necessary to restrict the angle within that range and this requires a few "IF" traps, as shown in the polynomial method where the working angle is between $-\pi/2$ and $+\pi/2$. This is another disadvantage for implementation on a T800.

4.1.5. Summation Recursion

This method, summation recursion in coupled form or a two-dimensional vector rotation, uses two basic trigonometric identities, for the sine and cosine of the sum of two angles:

$$\sin(\alpha+\beta) = \cos(\alpha)\sin(\beta) + \sin(\alpha)\cos(\beta)$$
$$\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$$

This method could be very effective for a real-time implementation, as the set-up [68 clock cycles] and the continuation costs [62 clock cycles each] are not so high, and the memory demand is low. Due to the dependency on both the sine and cosine components, however, this method is not so effective for applications that require only one component.

```
fq:     output frequency
sf:     sampling frequency      [constant]
pi:     2π                      [constant]
amp:    amplitude
incl:   inclement
c:      cosine value
c0:     cosine(θ)
c1:     cosine(θ-1)
s:      sine value
s0:     sine(θ)
s1:     sine(θ-1)
out:    output channel

-- set up
incl := (fq * pi) / sf
c0 := 1.0 (REAL32)
s0 := 0.0 (REAL32)
c1 := COS (-incl)
s1 := SIN (-incl)
```

List 4.1.4a: Sine Oscillator by Summation Recursion Method.

```
-- cycle
s := (c0 * s1) + (s0 * c1)
c := (c0 * c1) - (s0 * s1)
out ! (s * amp)
c1 := c0
s1 := s0
c0 := c
s0 := s
```

List 4.1.4b: Sine Oscillator by Summation Recursion Method.

4.1.6. Chebyshev Recursion

Considering the second-order linear difference equation and applying the *z*-transform:

$$y_{(n)} = \alpha y_{(n-1)} + \beta y_{(n-2)} + x_{(n)}$$

$$H_{(z)} = \frac{Y_{(z)}}{X_{(z)}} = \frac{1}{1 - \alpha z^{-1} - \beta z^{-2}}$$

Solving for the roots of the denominator leads to two cases. In the case where $\alpha^2+4\beta \leq 0$ the poles of $H_{(z)}$ are complex conjugates. They appear in the z-plane at $z = Re^{j\theta_c}$ and $z = Re^{-j\theta_c}$.

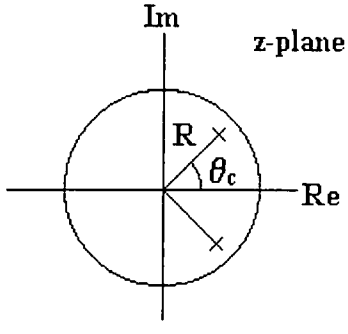


Figure 4.1.1.: Second-order Resonator Poles.

Here,

$$\theta_c = 2\pi \times \frac{f_q}{f_s} = \omega T$$

where $f_s = T^{-1}$: sample frequency

f_q: frequency of a tone

R: the radial distance of the poles

θ_c : the angle off the real axis

The equation can be rewritten as:

$$H_{(z)} = \frac{1}{(1 - Re^{j\theta_c} z^{-1})(1 - Re^{-j\theta_c} z^{-1})}$$

$$H_{(z)} = \frac{1}{1 - 2R\cos\theta_c z^{-1} + R^2 z^{-2}}$$

Since the dual-output of sine and cosine is not needed, a simpler sine wave can be implemented. For *R*=1, with no zeros and the dumping set to zero, the bi-quad transfer function becomes:

$$H_{(z)} = \frac{1}{1 - 2\cos\theta_c z^{-1} + z^{-2}}$$

A simple oscillator may be computed by solving the corresponding difference equation:

$$y_{(n)} = 2\cos\theta_c y_{(n-1)} - y_{(n-2)}$$

To generate an oscillator of amplitude A, the difference equation is started from

$$y_{(n-1)} = 0$$

The amplitude is set by seeding the correct $y_{(n-2)}$ value:

$$y_{(n-2)} = A\sin\theta_c$$

```
fq:     output frequency
sf:     sampling frequency   [constant]
pi:     2π                   [constant]
amp:    amplitude
cont:   constant
s:      sine(θ+1)
s0:     sine(θ)
s1:     sine(θ-1)
t1:     temporary storage
out:    output channel

-- set up
t1 := (fq * pi) / sf
cont := 2.0 (REAL32) * COS (t1)
s0 := 0.0 (REAL32)
s1 := SIN (-t1) * amp

-- cycle
s := (cont * s0) - s1
out ! (INT32 ROUND s)
s1 := s0
s0 := s
```

List 4.1.5.: Sine Oscillator by Chebyshev Recursion Method.

The initialisation requires 101 clock cycles [5.05 µsec], and each cycle does 18 clock cycles [0.9 µsec].

This implementation typically gives between -80 and -120 dB signal-to-noise ratio on a full 16-bit range [2^{16}=65536] sine wave calculated in 32-bit floating point format as implemented, comparable with a sine wave calculated by the Taylor series expansion [see Chapter 4.1.2.] up to 100 components in 64-bit floating point format to achieve higher precision when subsequently converted into 32-bit format for a quantitative comparison.

The measurements were done starting after the generation of 32,000 cycles, about one second after the beginning of a sine wave. In the case of the saw tooth waves consisting of a number of sine waves in different frequencies, however, some phase drifting was observed after generation for about ten minutes. Since the oscillators are operating independently, without synchronisation in their phase except at the initial set-up, the phase of the sine waves in different frequencies, but as a part of harmonic series, may not be aligned at each corresponding cycle, due to the precision of the calculation. For example, the third cycle of an 880 Hz sine wave may not begin at exactly the same time as the second cycle of a 440 Hz sine wave.

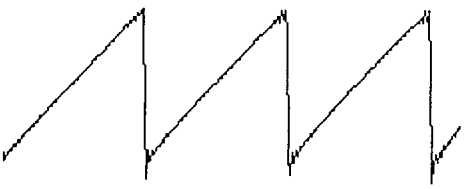


Figure 4.1.2a.: Saw-Tooth Wave by 8-bit integer.

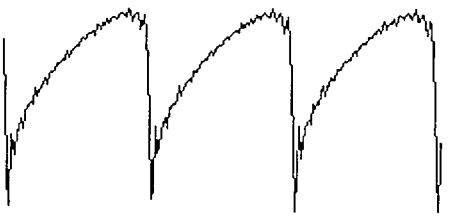


Figure 4.1.2b.: Saw-Tooth Wave by 8-bit integer after 10000 cycles.

This phase distortion problem is marginal, as most musical notes are not as long as ten minutes, and individual sine waves are accurate enough to keep the original pitch.

4.2. 81-Fixed-Voice Implementation

4.2.1. Synthesis Method

To comply with the hardware limitations of the 160 Transputer Network, in particular, the availability of only 4k-byte of local memory on each networked transputer, a Chebyshev recursion method was chosen for the sine wave generation. The recursive method requires a minimal size of memory for multiple oscillators implemented in parallel [see Chapter 4.1. and Gordon and Smith 1985], and generates high resolution sine waves by computing the projection of a rotating vector on the *x*- and *y*-axes. Advantages of the method are:

a) Few stored data components are required.

b) Each value needs few computations.

c) The waveform has a low distortion factor, since the difference equation simulates a physical system whose solution is a perfect sine.

Recursion requires the handling of limit cycles over the long-term, but they can be made to lock on to periodic values.

	calculation cost (clock cycle)	
method	**set up**	**oscillation**
Wavetable	28.0	32.4
Taylor Series	28.0	18849.3
Polynomial	61.0	246.6
Summation	68.0	62.0
Recursion	101.0	18.0

Table 4.2.1.: Computation Time Required in Sine Generation Methods.

For the preliminary implementation, the seeding method described in Chapter 4.1.6. was applied; starting from phase 0. Due to lack of an amplitude envelope, a click noise can be heard at the end of each tone. Using this method, it is also difficult to change the frequency of a sine wave without an extra calculation to keep the output continuos. An initialisation of an oscillator, however, requires only a few operations; about 5.05 μsec in a 20 MHz clocked T800 transputer. In this implementation, a 32-bit integer format is used internally for data communication, although the DAC has only a 16-bit bandwidth, since this achieves optimal performance from the transputer software [see Chapter 2] and also ensures that changes in amplitude level do not result in a loss of quantisation accuracy.

To improve the tumbrel quality, the initial phase information was later set using the SHARC Timbre Database (Sandell 1994). This means, however, that the initial value of the synthesised sound is not always zero, and this necessitates the introduction of an anti-clicking process, using a short amplitude envelope, at the beginning and the end of a tone. Despite the introduction of phase information the quality of synthesised sound was not improved, due to the lack of a long amplitude envelope.

4.2.2. Prototype Programme

A transputer at the top of a single element works as a mixer and the other three members of the group work as a unit of oscillators. At a 32 kHz sampling frequency, each oscillator unit placed in a transputer is able to contain eight recursive sine oscillators that can be controlled

independently in both amplitude and frequency. At 44.1 kHz, representing CD quality, five sine oscillators can be situated in a transputer.

This combination, consisting of one mixer and three oscillator units, is expanded on to a larger network, recursively. As a prototype configuration, an 81 note-organ was implemented and tested. In this configuration, the network is capable of accommodating 81 oscillator groups [one group per transputer] that provide 648 recursive sine oscillators in total, at a 32 kHz sampling rate or 405 oscillators at a 44.1 kHz sampling rate. Each oscillator group corresponds with a fixed MIDI key.

Calculations for the oscillators are executed in 32-bit floating point format. When a sound sample exits from an oscillator unit placed on a transputer, it is converted into an integer number in 32-bit format that is then accumulated with other synchronous samples throughout the network. This bottom-up sample accumulation on a tree structure provides equal path length to each oscillator group, thus causing no phase delay.

The figures below show the configuration of the 81-voice organ. To present the processes implemented in parallel but in the opposite signal direction, the map is drawn as the revised ternary tree with its mirror image.

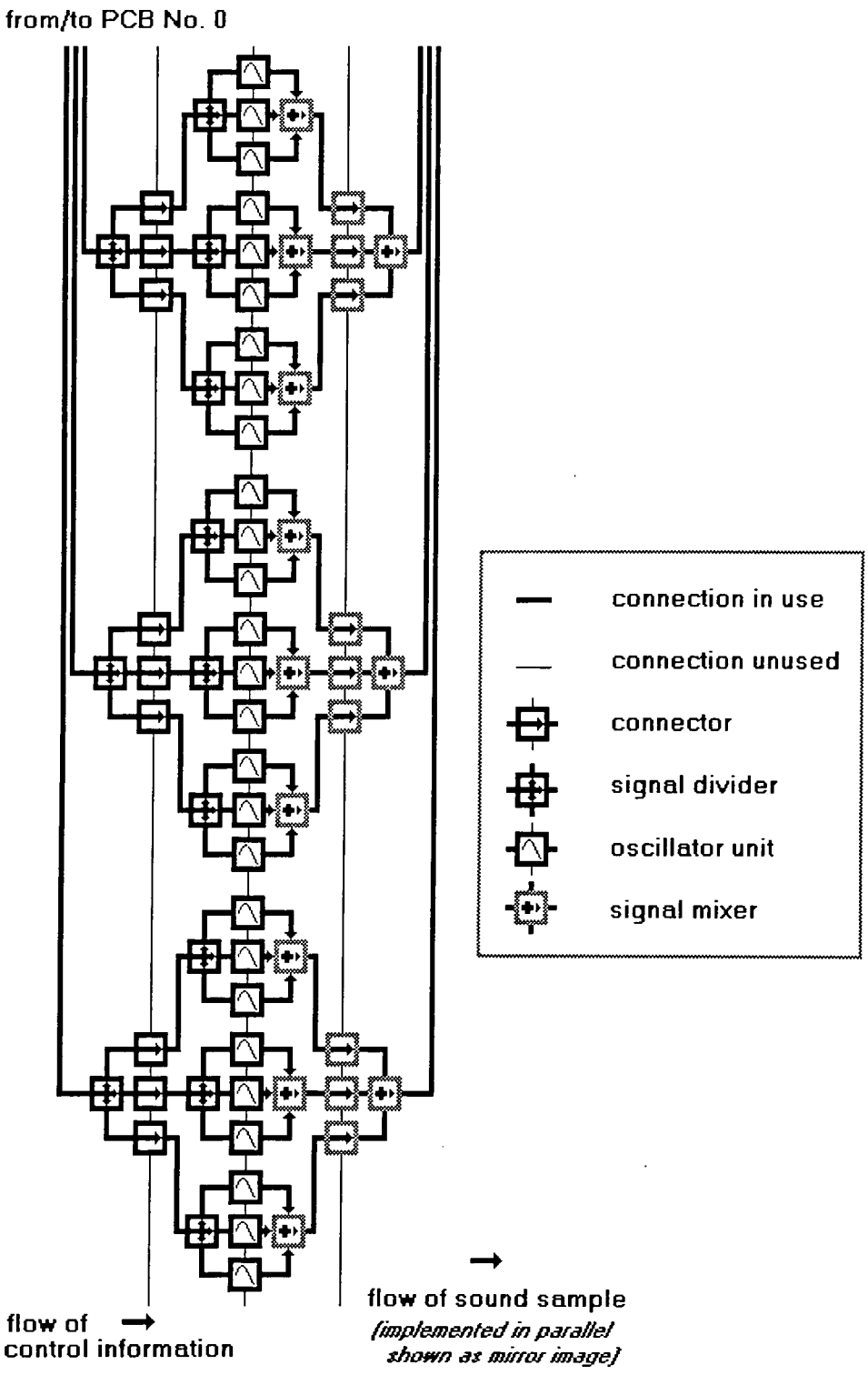


Figure 4.2.1a.: Configuration Map (Left).

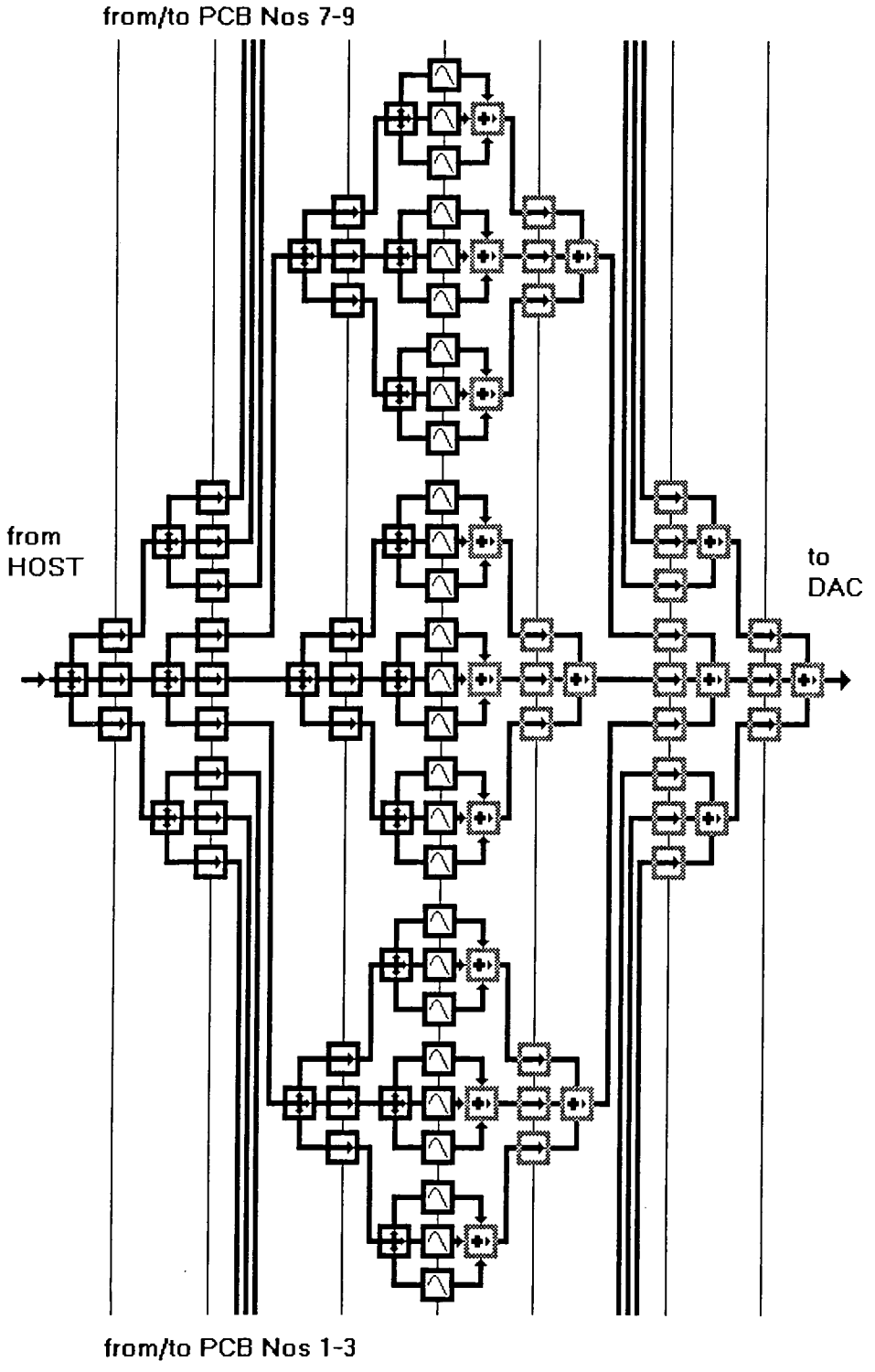
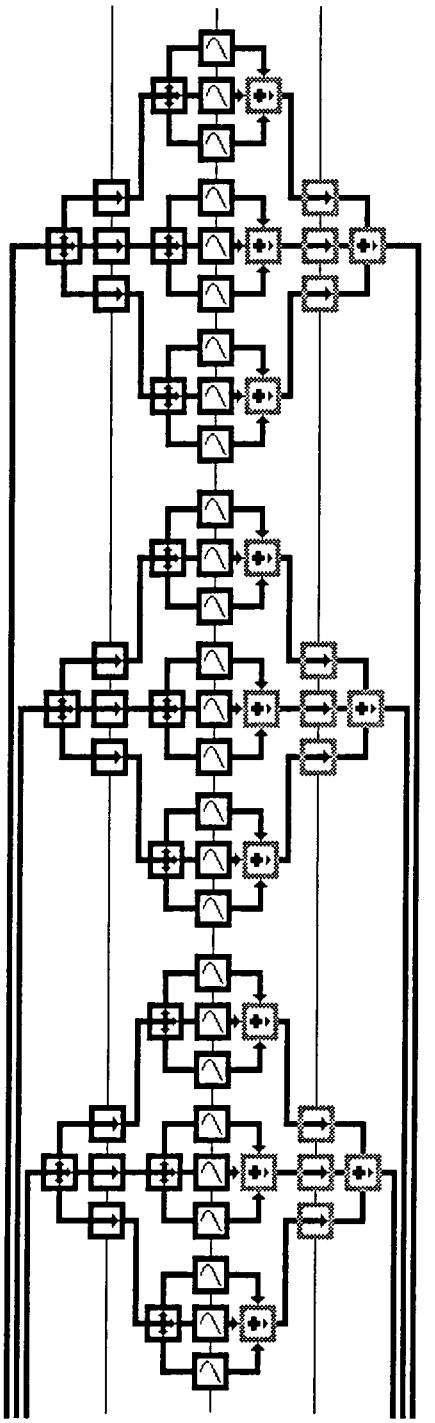


Figure 4.2.1b.: Configuration Map (Centre).



from/to PCB No. 0

Figure 4.2.1c.: Configuration Map (Right).

As described earlier, a standard MIDI keyboard has been used as the primary control device for the synthesis engine, and a custom-designed MIDI-to-transputer interface board [see Appendix 3] provides the communication link between the MIDI keyboard and the network. For the purposes of initial development, MIDI commands were restricted to primitives such as "note on" and "note off", assigned to the audio transputers via the MIDI controller unit. The "raw" MIDI information is filtered to the above primitives and then transformed to a single packet of transputer control data, consisting of the key number and its associated velocity [amplitude].

These messages are led to the top of the network and passed down the branches to the bottom of the tree, where the synthesis instructions are allocated to individual oscillators. On the way to the bottom the message is sorted at signal routers according to the key number. This top-down control method enables groups of oscillators implemented in parallel to work entirely independently.

Over the network the flow of control information and the flow of the sound output are both implemented in parallel, but in opposite directions. The latter is a constant continuous flow at a 32 kHz bandwidth. The former is handled on an on-demand basis, but could involve a data rate of up to 150 packets per second.

If the control signal is periodic, the processes in parallel could be controlled cycle by cycle. Or, if the control signal is at a low frequency, such as a few packets a second, the processes could be handled in "ALT"

structure using a channel guard that costs 24 clock-cycles [1.2 μsec on a 20 MHz clocked transputer] per execution where at 31.25 μsec is available for processing a sound sample at a 32 kHz sampling rate. This leads to both of the processes having to be placed in a high-prioritised simple parallel structure. When the sound sample stream is placed in higher priority and the control signal is in normal [low] priority, the latter may not obtain a time slot, since the high-demand higher priority process occupies most of the processing time.

Due to the nature of the transputer, a time-multiplex parallel processor, it is necessary to incorporate a sound buffer to ensure a constant flow of data to the sound output. A balance has to be struck between a long buffer, which will result in a noticeable performance lag, and a short buffer, which will not allow the reliable accumulation of events for a steady output data stream.

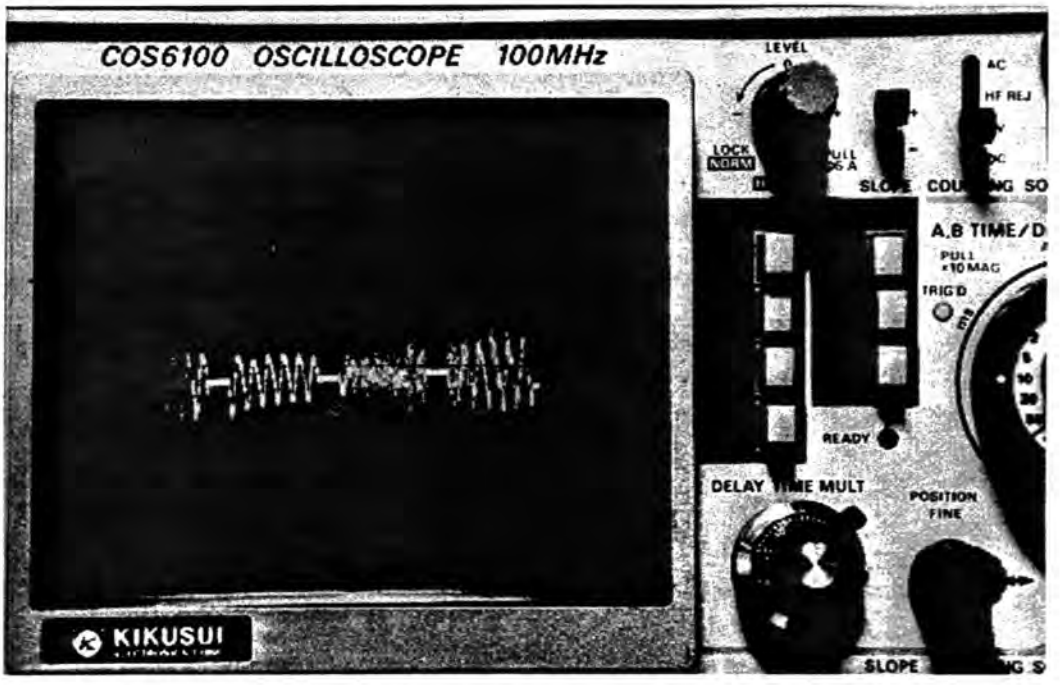


Figure 4.2.2.: Result of Insufficient Buffer Size.

If the output sound sample stream is not constant, but the disruption is short enough to sustain the clock operation, as shown in the above figure, the DAC produces a corrupted output; usually a lower pitch as a result of old samples being retained in the cache.

With a 16 msec-long sound buffer situated between the network and the DAC the system performs reliably, providing the input event control rate is less than about 150 keystrokes per second. In real-time processing, late data almost always result in computing errors. Viewed from the human perception side, it has been shown that the introduction of a very small response delay to reduce this possibility can generally be tolerated by a performer, providing it is kept constant.

Since the raw MIDI data from a keyboard are filtered to essential note/event commands, ignoring other higher density information such as that associated with continuous controllers and system exclusive messages, the peak data rate through the network should not normally exceed the 150 MIDI-event per second boundary, although this may depend to some degree upon the characteristics of the key sensor and other components integral to specific designs of MIDI keyboards.

Subjective tests have confirmed that the 16-msec-long buffer is acceptable to most performers, and given the conflicting considerations identified above perhaps it is the best compromise that can be achieved in this particular context. [See Chapter 3]

When a higher rate of MIDI information is supplied, the system will temporarily halt until the control information has been distributed to its target processors (Itagaki, Purvis and Manning 1994). This is an interesting example of the transputer's fault tolerant behaviour in certain circumstances, if correctly programmed; the result of pseudo-parallel processes with a standard communication protocol that is based on "message" and "acknowledgement". This means that if the receiver side is busy the sender process is frozen until an acknowledgement comes in from the receiver.

Providing all subsequent processes can be sequentially halted in a similar manner without loss of data, the deadlock is not fatal either locally to the processor concerned or globally in terms of processes that are in parallel. Under these conditions the system will always recover at the earliest opportunity without further corrective action.

The set-up latency of the system, the timing between "key press" and "sound out", is about 20 μsec prior to the buffer at the DAC, that is short enough to satisfy the conditions for real-time synthesis at a sampling rate of 32 kHz where, as already noted, a maximum interval of 31.25 μsec is available for computation between successive samples.

Because of the fixed hardware architecture, however, some of the transputers have to work as connectors. To improve the efficiency of the network, a monitoring programme is introduced; a simple process situated in each transputer as a lower priority parallel process.

4.3. Performance Monitoring

The monitoring routine measures the execution time of a simple process and returns the figure to the host transputer through the network, together with the address of the transputer that is assigned by the host. Due to the hardware limitations of the tree architecture, this monitoring information should be sent through a single output, sharing a channel with the sound output. The software language imposes another limitation; one process can only have an access to a single channel. To avoid this violation, it is necessary to attach an extra process to control the communication channel, using an "ALT" structure.

If such a constraint is used to manipulate the sound output and the monitoring information for each oscillator unit, in turn, the effect is to reduce the capacity of each oscillator unit by 50%; from eight oscillators to just four. Accordingly, the configuration at the bottom of the tree has to be changed; one mixer, two oscillator units and a dummy oscillator. The monitoring information from an oscillator unit is then diverted to an adjacent "dummy" oscillator that works as a mixer of monitoring information.

As a result, it was deduced that some transputers at higher stages in the tree-structure assigned as "signal routers" are under-utilised, with spare processing time that may be used for additional synthesis operations.

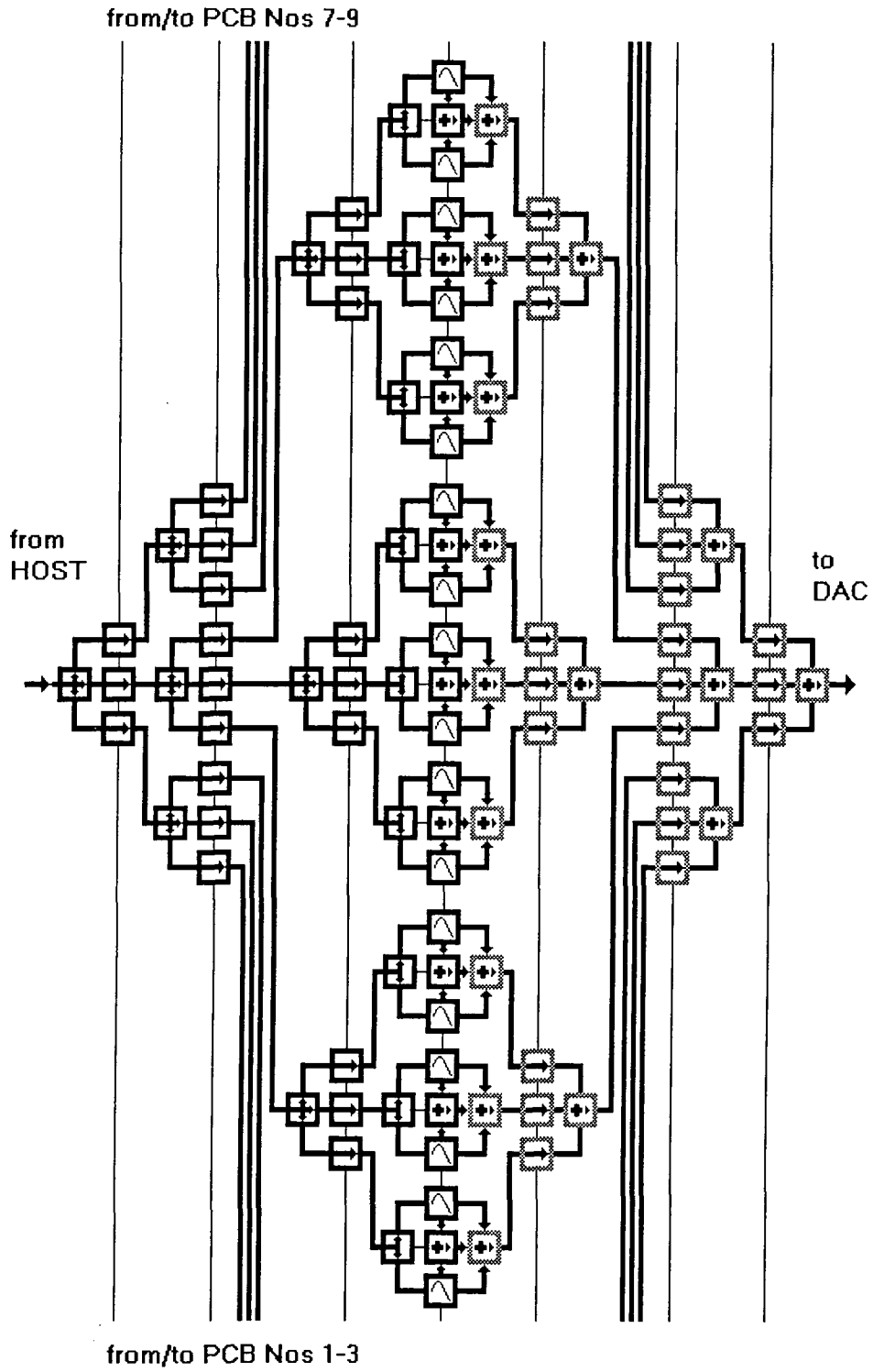


Figure 4.3.1.: Configuration of a Monitoring Program (part).

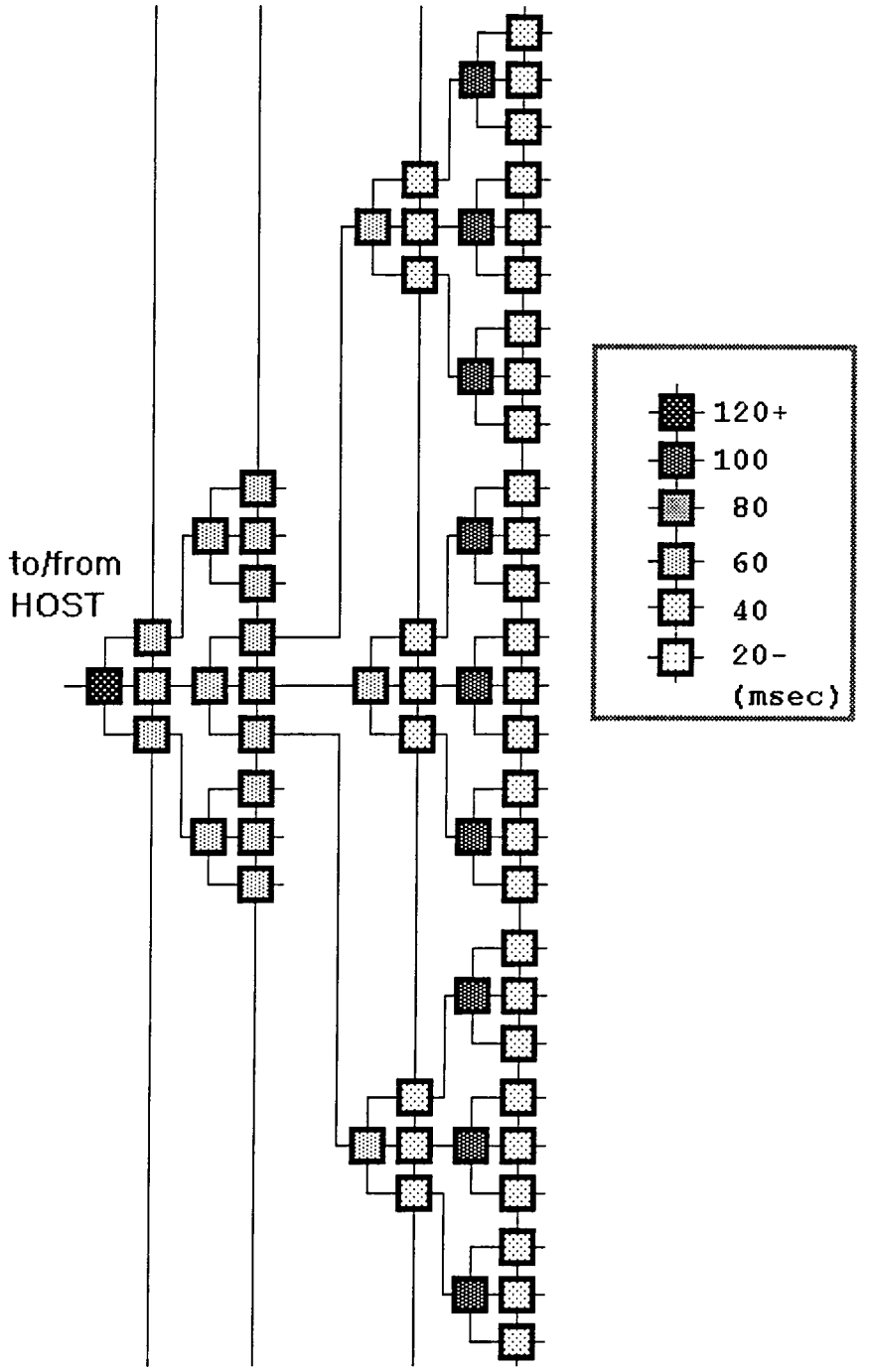


Figure 4.3.2.: Monitoring Result (part).

A signal router, or connector, is re-designed so that the information in both directions is diverted to an adjacent transputer on the same level, using link number one or link number three. The spare processing capacity thus released may then be used for an additional small oscillator unit, half of the standard provision, as a process in parallel with the connector.

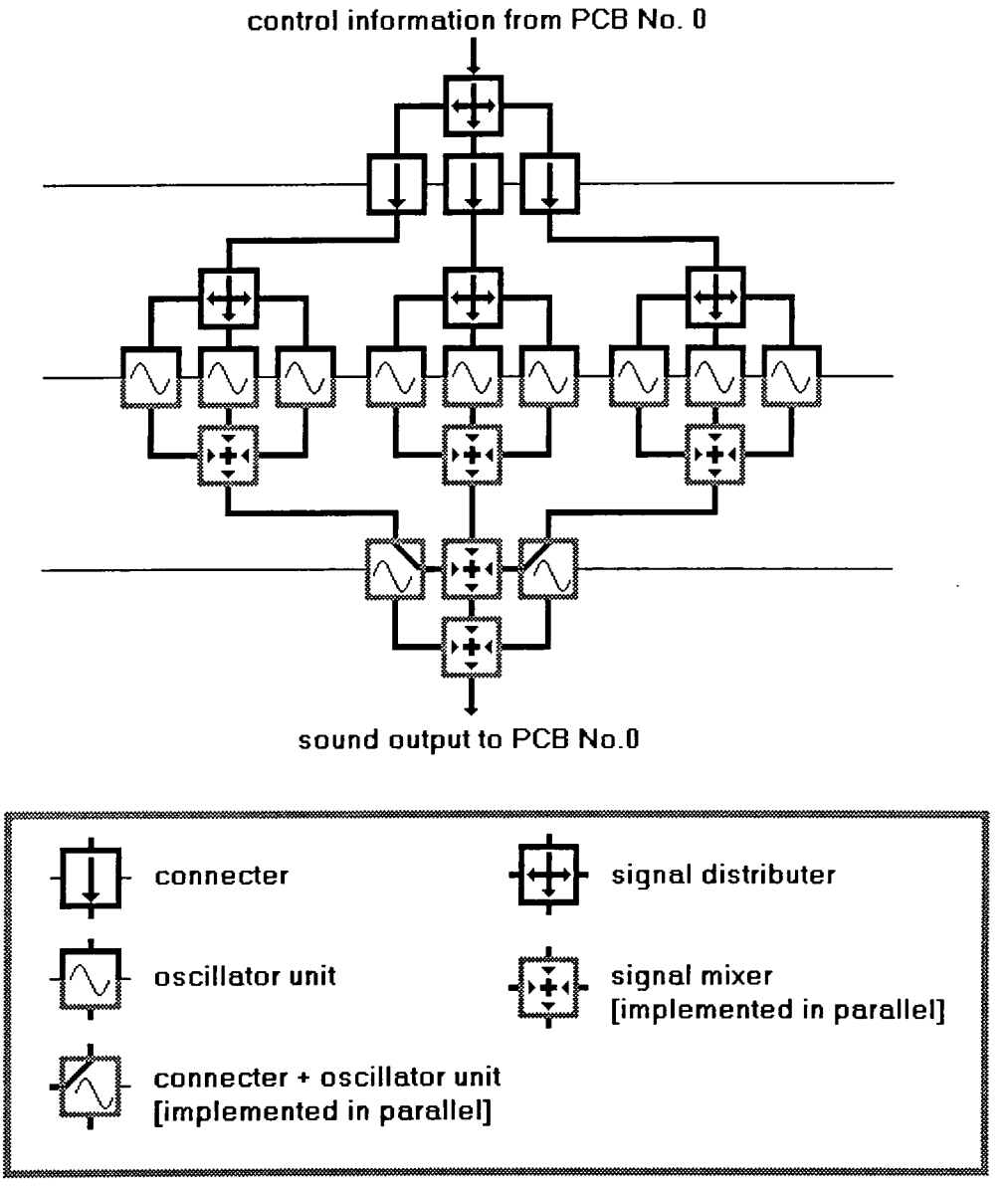


Figure 4.3.3.: Revised Implementation.

4.4. 88-Fixed-Voice Implementation

Since the processes of synthesis to be employed are entirely additive, working from basic sinusoids, the generation of interesting timbres becomes a function of how many individual sine wave oscillators are assigned to each note, and how they are regulated in terms of both frequency and amplitude.

When further consideration was given to the harmonic content of the notes to be synthesised, especially those in higher octaves, and the limitation of the DAC in terms of the Nyquist frequency was also taken into account, it became clear that it is not always necessary to assign all the component oscillators available at each transputer to the synthesis of the harmonic content of a single note.

For example, in the case of the highest sounding A in an acoustic piano; a fundamental frequency of 3,520 Hz, its fifth harmonic [17,600 Hz] is already higher then the Nyquist rate of a 32 kHz DAC [16,000 Hz]. This means that four oscillators are quite sufficient to reproduce such a high order note with acceptable fidelity, releasing spare oscillators to enrich the spectra of notes with lower fundamental frequencies, and the addition of some extra notes at the lower end to create a standard acoustic piano range.

In the light of the above assessment, the program was optimised to accommodate a range of 88 notes over the network, serviced by a total of 752 oscillators at a 32 kHz sampling rate [sound sample 4.3.]. The allocations of oscillators range from sixteen per note at the lower end of

the range to four per note at the top. The overall availability of harmonic components for each note, however, still falls significantly short of the minimum criteria stated earlier for the approximation of most instrumental sounds other than those associated with electronic organs except, given the Nyquist considerations, at the very top of the range.

The oscillator units, additionally implemented to the revised configuration, are situated at a middle level of the tree structured network, where the signal routers were located in the original 81-voice model. Since most of the oscillator units are at the bottom of the tree structure, each additional oscillator unit has a slightly shorter path to the tree-top and this conceivably may cause some latency and phase differences. As mentioned in the 81-voice-configuration, the start-up latency for a note synthesised with the oscillator units at the bottom of the tree is about 20 μsec, where that of the oscillators at the middle level is about 18 μsec. The difference, about 2 μsec, is far shorter than the sample period, 31.25 μsec, and thus can be considered as marginal.

4.5. Improvements on the Network

After implementing these programs, the network was physically modified to allow connections from the left side, the right side, and the bottom of the tree to external resources, such as other transputers with larger on-board memory capacity. These modifications were made partly in response to a need to allow the functionality of the audio processor to be expanded. It is worth noting that in setting out to design such an audio processor, the research group had a unique opportunity to build a massively parallel architecture essentially from first principles. Expandability has, nevertheless, been a paramount consideration from the outset, and the relative ease with which modifications such as the above could be made demonstrates the versatility of the transputer and its serial link system of communications.

The only significant engineering problem so far encountered has been heat emission from the transputers which are placed densely on the PCBs. According to the data book published by the manufacturer (INMOS 1989), the T800 transputer civilian models should work reliably under conditions of temperature between 0 and 70 °C with transverse air flow of about 1 m/s. For a real-time processing application like an 88-note organ, however, the condition seems to be lowered to about 35 °C: the network ceases to work after 20 minutes of processing when the processors are hotter than an average human body temperature. This heat emission problem necessitates some forced air-flows through the network, and providing adequate cooling for such a densely accommodated multi-processor system proved a major design challenge which could only be partially solved in the time available. In summer time, when the room

temperature is about 25 °C or more, the system is still unable to perform a real-time job reliably if left switched on for more than about thirty minutes. Solutions to the problem would be a significant reduction of the processor density on a PCB and a wider distance between the PCBs; about 25 mm between the PCBs on the current system [less than 20 mm between the mounted transputers and the adjacent PCB].

4.6. Conclusion

The real-time additive synthesis applications, 81-fixed-voice and 88-fixed-voice, were implemented on the 160 transputer network. In the 88-voice model, 752 real-time recursive sine oscillators are available over the network, at a sample rate of 32 kHz. These configurations reliably operate up to 150 keystrokes per second.

The set-up latency of about 20 μsec is at an acceptable level for a hard real-time system, as the interval of consecutive sound samples should be produced within a sampling period of 31.25 μsec. This demonstrates the potential of the 160 transputer network as a self-contained real-time audio processor, in particular when configured as an additive synthesis engine. To establish fully its credentials as a self-contained audio synthesiser, however, further research is required into the dynamic control of amplitude.

Due to the hardware limitations of the network, it has not proved possible to implement independently controlled amplitude envelope generators over the entire 81- or the 88-fixed-voice models, and it may be necessary to reconsider the network architecture in order to achieve such an objective, possibly at the expense of the total number of oscillators.

Chapter 5. Optimisation of Real-Time Additive Synthesis on the Network

5.1. "Pipe Organ Style" Borrowing

Having revised the implementation of the 88-voice organ, the immediate priority was to improve the tumbrel quality, since only 16 oscillators per key maximum and about eight per key average are available. Although the priority might have been given at this stage to the requirement of the envelope generation facilities, it was felt that improving the specification of the tone generation facilities was a more logical first step, since this directly tested and sought improvements in the basic distributed processing architecture which paves the way to the efficient operation of the network as a whole.

There are, however, some redundancies over the network, since several oscillators operate at the same frequency. For example, the eighth harmonic of the 110 Hz note is 880 Hz, that is also the fourth harmonic of the octave at 220 Hz, the second harmonic of 440 Hz and the fundamental of 880 Hz. Fortunately, there is a way to reduce the number of oscillators required for the configuration of fixed-note allocations by using amplitude information.

A suitable technique has been applied in a conventional pipe organ design: high pitch stop-ended pipes are used to boost the high harmonic components of a low pitch open-ended pipe (Audsley 1905). The even number harmonic components may thus be "borrowed" from other tones. [Organ builders prefer to use the word "duplication" instead of

"borrowing".] For the odd numbered components, however, some compromise must be made.

An acoustic piano and other fixed-pitch modern instruments are tuned to equal temperament, in which each semitone is made an equal interval. In a twelve-tone system, commonly used in modern instruments, the interval is $2^{\frac{1}{12}}$ of a base key frequency. This leads to another type of "pipe organ style" borrowing. For example, the ninth harmonic of A 110, 990 Hz, may be replaced with the sixth harmonic of E above A 110 [$110 \times 2^{\frac{7}{12}} = 164.81$], 988.88 Hz, with a 0.1128% difference in frequency.

A subjective feasibility test was conducted; a comparison between a note with true harmonics and that of some borrowed harmonics. A "true" note, a triangle wave of A 110, was synthesised with up to the sixteenth harmonic components [sound sample 5.1.1.].

Harmonic	Frequency (Hz)			Origin
	real (A)	borrowed	+ / - (%)	
base	110.00			
2nd	220.00			
3rd	330.00			
4th	440.00			
5th	550.00			
6th	660.00			
7th	770.00			
8th	880.00			
9th	990.00	988.88	-0.113	E 6th
10th	1100.00	1108.73	0.794	$C^{\#}$ 8th
11th	1210.00	1222.30	1.016	F 7th
12th	1320.00	1318.51	-0.113	E 8th
13th	1430.00	1453.57	1.648	$G^{\#}$ 7th
14th	1540.00	1567.98	1.817	G 8th
15th	1650.00	1661.22	0.680	$G^{\#}$ 8th

Table 5.1.1.: Comparison between "true" and "borrowed" Harmonics.

A "borrowed" note was then synthesised with the front eight harmonics using components of "true" tones and ninth to sixteenth harmonics using "borrowed" tones [sound sample 5.1.2.].

The differences, in terms of frequency, are less then a few cents. For my ears, however, the notes have slight but recognisable differences in their tumbrel features, especially in the brightness of the sound, the slight mistunings leading to a blurring of the overall definition quite different to any conventional chorusing characteristics. As Audsley concluded in his research on borrowing [of pipes] and duplication [of upper partial tones]:

> In conclusion, we may say that beyond the formation of an expressive auxiliary Pedal Organ, as above alluded to, we strongly condemn the practice of borrowing and duplication as unscientific, inartistic and fatal to a perfect tonal appointment: its absurd side is eloquently set forth in the "Nouveau Système" of M Léonard Dryvers.
>
> "Borrowing and Duplication."
>
> in *The Art of Pipe Organ Building* (Audsley 1905)

For the reasons above, the full implementation of the "pipe organ style borrowing" was abandoned.

5.2. Dynamic Allocation of Notes

In the configuration of the network as an 88-voice organ described above, the situation arises in "normal" performance, or even in the case of a simple duet, where a static allocation of oscillators to specific notes frequently involves a high degree of redundancy. Therefore, there are significant advantages to be gained from a dynamic note allocation algorithm that allows optimal deployment of oscillators, thus increasing the range of timbres that can be generated.

In the revised configuration, the system is programmed to accommodate 27 simultaneous notes with up to 24 oscillators per note, or 9 notes with up to 72 oscillators each; a total of 648 oscillators over the network.

voices	oscillators per voice
81 (fixed)	8
88 (fixed)	16, 12, 8, 4
27	24
13	***48***
9	72

N.B. The program in ***italic*** has not been implemented.

Table 5.2.1.: Allocation of Oscillators.

The voices are controlled by a voice allocation unit that sends control signals to the top of the network. At this stage, some latency for the voice selection, 150 nsec per voice, has to be expected in addition to the set-up latency of about 20 µsec.

In the worst case, the 27-voice model operating at full capacity, whereupon a 28th note is activated, forcing release of the longest-

sounding note, the maximum latency at voice selection is estimated to be about 8 µsec. A control signal packet now has to contain an additional byte for voice information, thus the stream of sound output is interrupted more frequently, in turn requiring a longer sound buffer.

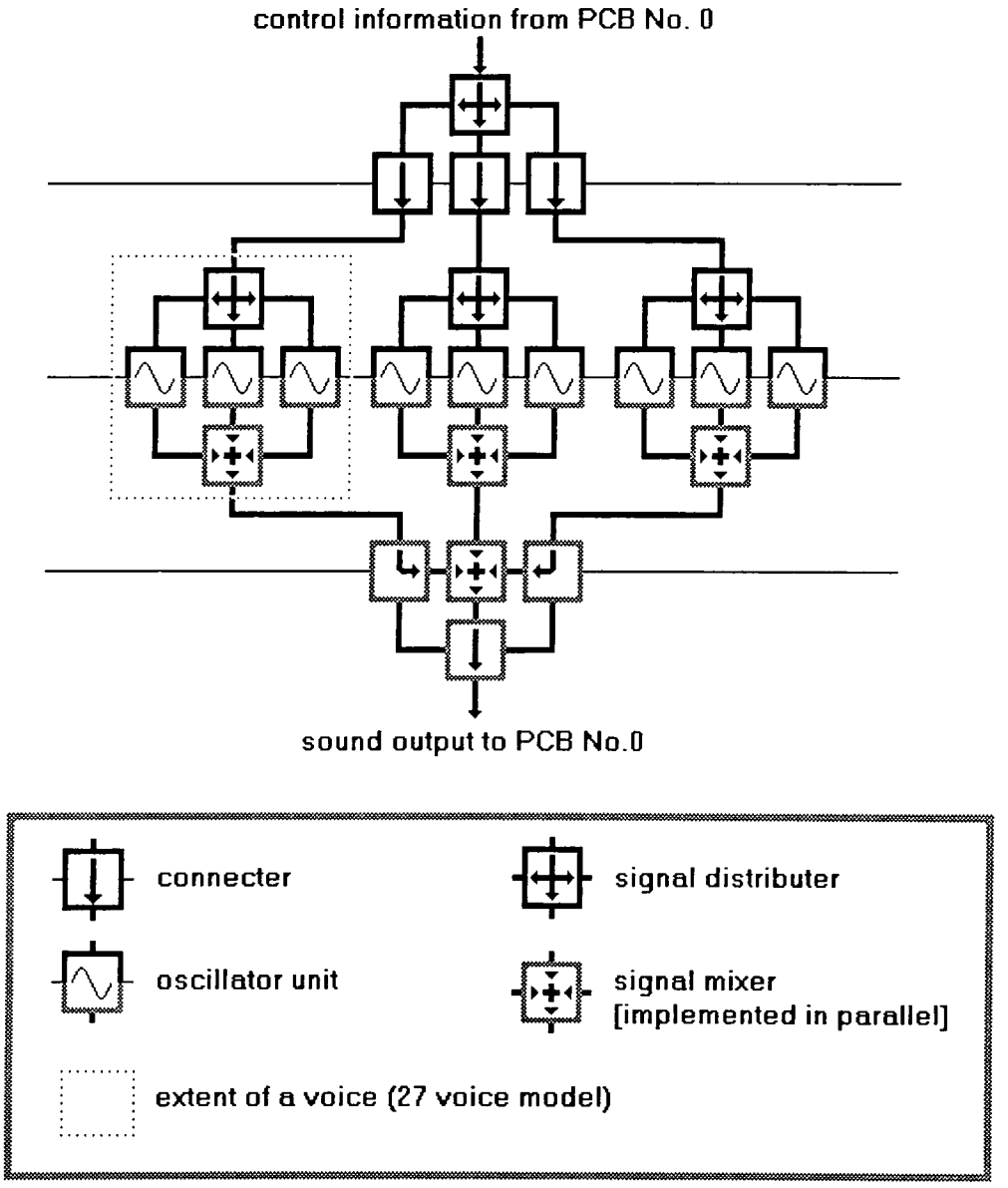


Figure 5.2.1.: Configuration Map 27-voice Model (part).

As an initial step to overcome these problems, the size of buffer was doubled to 32 msec. Despite this significant increase in its length, however, the longer buffer still failed to reliably return an acknowledgement signal to the network, resulting in a non-recoverable time-out condition with loss of data in the DAC driver. This problem was solved by means of a double buffering arrangement: whereby a very small [less than 1 msec] second buffer is added between the network and the main buffer to ensure quick acknowledgements to the network. To many performers, however, the resulting response delay of almost 33 msec is uncomfortably close to the maximum that can be reasonably tolerated. By shortening the main buffer length to overcome this objection, this itself, unfortunately, reintroduces the risk of time-out errors.

After the implementation above, three extra functions were introduced; harmonics, phase and amplitude envelope. A few sets of simple amplitude information were given to the oscillator units that create saw-tooth, square and triangle waves [sound sample 5.2.1.]. Thence, using Sandell's "SHARC Timbre Database" (Sandell 1994), the amplitude and the initial phase for each harmonic content were supplied: the former was distributed to each oscillator unit and the latter was stored over the network, residing in the on-chip memory area. Despite an effort to keep the memory load to less than 75%, this extra memory requirement introduced a significant overload into the sound processing itself. The problem was solved by means of reducing phase information by way of a compromise. Due to the lack of an amplitude envelope, however, the raw synthesised sound was still very raw and unshaped.

A few types of simple independently controlled amplitude envelopes were then implemented at the top of the routing for the each voice where a transputer works as a connector. Since the available time slots only permit a few operations per sound sample, and the memory capacity for the envelope are limited, less than 3.2k-bytes, only basic functions were chosen and tested; half-sine rise/decay, parabolic curb and hyperbolic [sound sample 5.2.2.]. A combination of the hyperbolic decay curb and a saw-tooth wave creates a passable string instrument-like sound, but, it remained evident that more sophisticated envelope shaping facilities are necessary, if the synthesis engine is to reduce its full potential. Most strongly, it is apparent that the potential trade-off of oscillator resources against envelope shaping routines will be even more severe than in the case of the simpler implementations of discussed in the previous chapter, and these need further investigation.

5.3. Multi-Rate Approach

Additive synthesis using a large set of digital sine oscillators is a fine-grain parallel algorithm and particularly compatible with the architecture of the T800 tree. The high control bandwidth required, however, has meant that its computation in real-time has only become economic in recent years, due to advances in VLSI technology. This has led to a re-awakening of interest in the research community for methods to optimise computation; such as multi-rate DSP techniques that integrate well with the traditional oscillator-set model of additive synthesis and thus with its implementation on the 160 Transputer Network.

A multi-rate optimisation is based on the idea of eliminating redundant computation. For instance, a sine oscillator producing a tone of 1 kHz requires a minimum of 2 kHz sampling frequency. On the other hand, preferred sample rates for audio output are 32 kHz [NICAM], 44.1 kHz [CD] and 48 kHz [DAT], some 16 to 24 times higher than the minimum requirement. Researchers in the area, however, suggest that a geometric progression of sample rates across the audio spectrum generally satisfies the greatest number of demands on a note-based synthesis engine (Phillips, Purvis and Johnson 1994). Theoretical research on the subject using complex oscillators has also been conducted; (Phillips, Purvis and Johnson 1996) and (Phillips 1997).

In the initial implementation, three sampling rates were used; 8 kHz, 16 kHz and 32 kHz [sound sample 5.3.]. Two FIR filters are placed at the top of each branch of the lower sampling rate groups to convert the output to a standardised sampling rate of 32 kHz: the technique applied here is

known as "over-sampling" the procedure common to many designs of compact disc players.

The constraints of real-time sound processing restrict the permitted length of the interpolation filter; a long filter performs a suitably sharp response, but results in a long delay that should be avoided for the real-time operation. This leads to some degree of compromise, involving a combination of a short FIR filter with an optimal grouping of the note and the oscillators. The filters have to be distributed over a part of the network, since a transputer has infinite capability in calculation speed. The smallest dividable unit, a leaf of the network, consists of four transputers, therefore, the number of transputers for a filter has to be a multiple of four.

For a fixed 88-voice application, up to sixteen oscillators are assigned for a note, except for the twelve notes in the lowest octave where the assignment is increased to 24. Notes are grouped according to their fundamental frequency and their harmonic contents. In this configuration, 1,296 oscillators are available over the network, in comparison with the single-rate model with 752 oscillators.

sampling rate	oscillators per group (transputer)	groups on network	sub total
8 kHz (1/4)	32	21	672
16 kHz (1/2)	16	18	288
32 kHz	8	36	336
TOTAL		**75**	**1296**

Table 5.3.1.: Allocation of Oscillators (1).

An alternative 44.1 kHz based configuration, which has not been implemented, would show a similar result.

sampling rate	oscillators per group (transputer)	groups on network	sub total
7.35 kHz [1/6]	30	15	450
22.05 kHz [1/2]	10	27	270
44.1 kHz	5	33	165
TOTAL		75	885

Table 5.3.2.: Allocation of Oscillators (2).

In the case of a multi-rate application using dynamically allocated notes (Itagaki, et al. 1995), the effects, measured by the number of oscillators allocated to a voice, are estimated as shown in the table below.

sampling rate [kHz] [production-output ration]	number of voices	oscillators per voice	total oscillators
10.7 [1/3], 32 [1/1]	22	36	792
4 [1/8], 8 [1/4], 32 [1/1]	15	80	1200

Table 5.3.3: Effect of Dynamic Allocation + Multi-Rate.

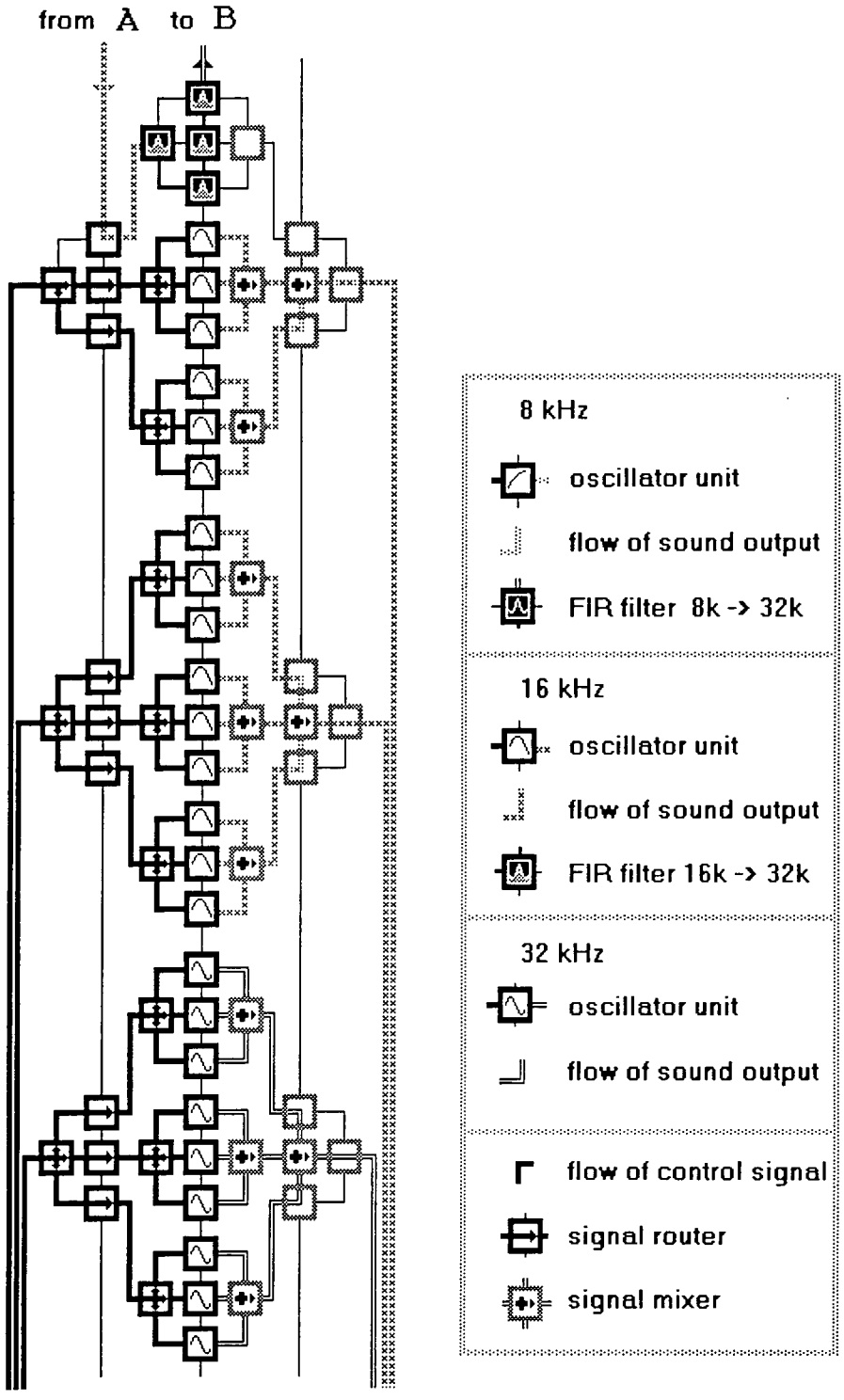


Figure 5.3.1a.: Configuration Map of Multi-rate Application (left).

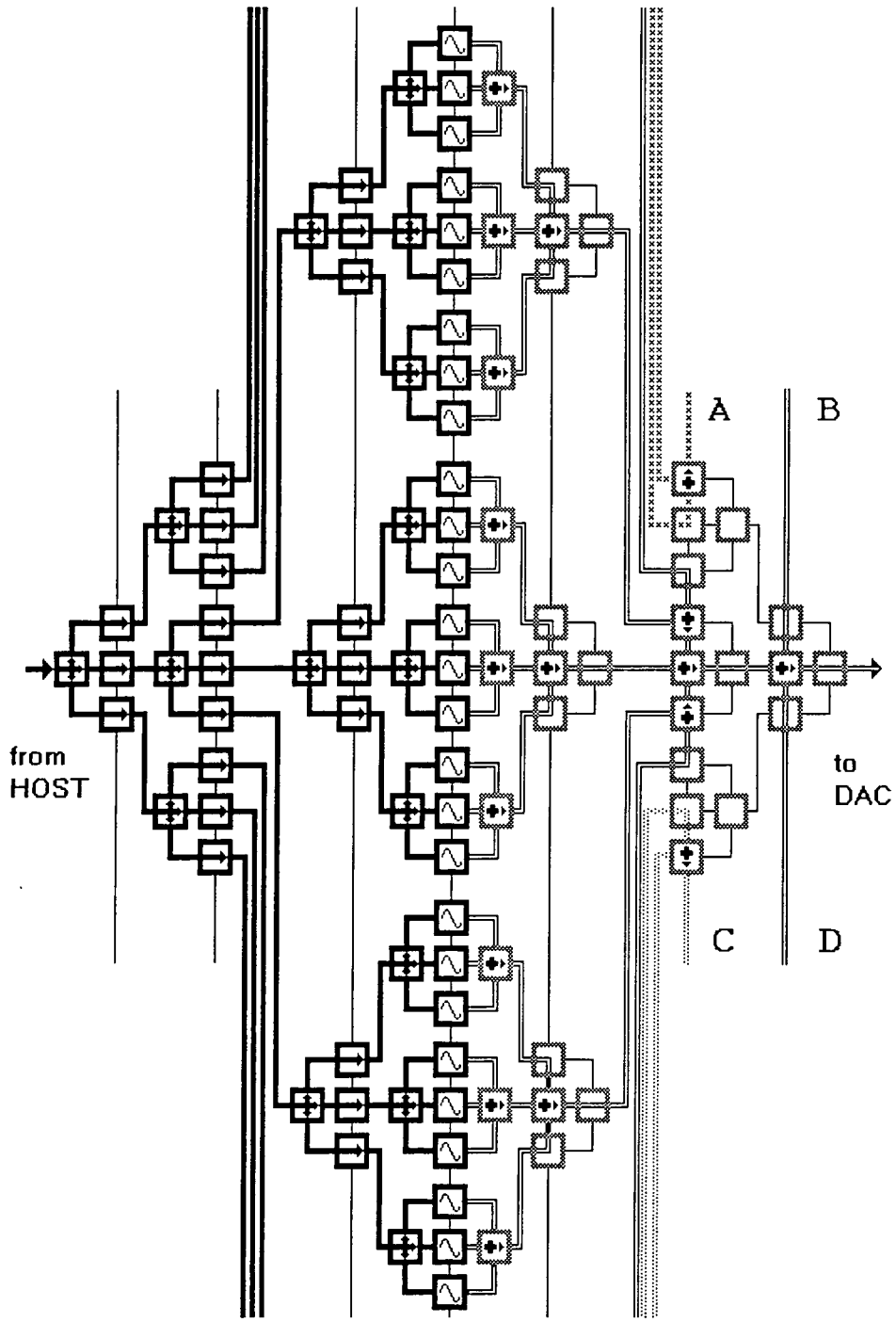


Figure 5.3.1b.: Configuration Map of Multi-rate Application (centre).

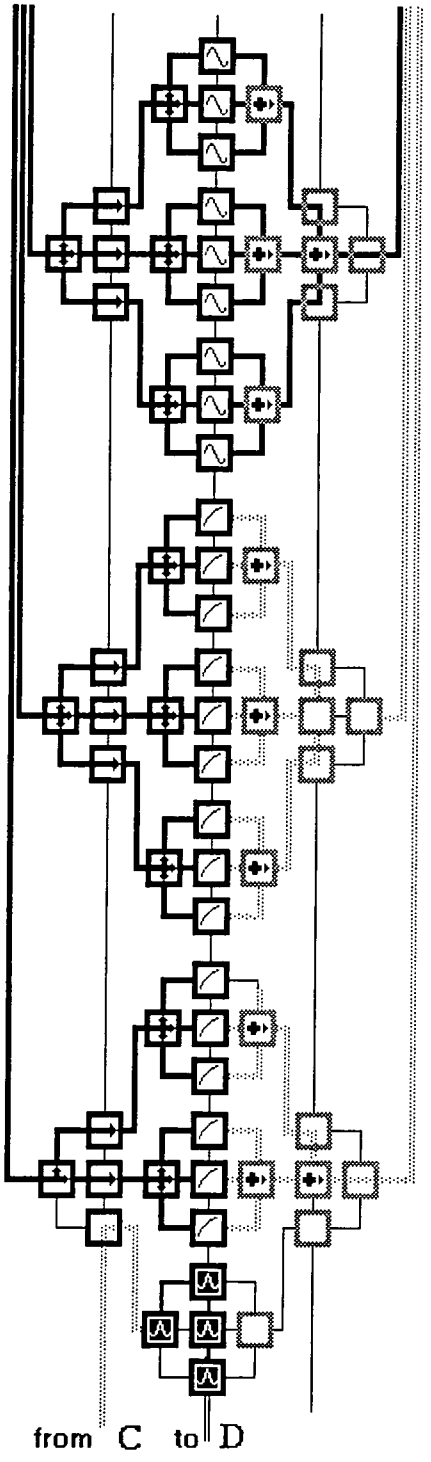


Figure 5.3.1c.: Configuration Map of Multi-rate Application (right).

5.4. Conclusion

The optimised methods, resource management by the dynamic allocation of notes and the multi-rate approach usefully increase the number of harmonics per note, representing a significant improvement in the quality of timbrel detail. The multi-rate optimisation alone has been shown to be effective for accelerating the process of dynamic tone generation, provided the notes are efficiently allocated to optimal sampling rates, and the combination of the sampling rates applied is suitable.

Due to the hardware restrictions on the network, especially in memory capacity and structure, there are limits when seeking to improve the tumbrel quality. As already noted, a new network structure, therefore, should be considered for future investigations, such as an asymmetrical tree or non-homogeneous tree in terms of memory capacity and calculation power.

Chapter 6. Granular Synthesis and Sound Granulation

6.1. History of Granular Synthesis

6.1.1. Acoustical Quanta: the theory behind the granulation

Granular synthesis or sound granulation is a means of constructing complex sounds from grains originally proposed as a representation of "acoustic quanta" by a British engineer Denis Gabor (Gabor 1946), who also invented the idea of the hologram. The basic idea of sound granulation for narrow bandwidth transmission / reproductive purposes is quite similar to the quantum-wave theory in physics: sound may be described as a sequence of elementary acoustic elements. It can be seen in cinema and video images where a rapid sequence of static images gives the impression of moving objects.

Gabor firstly presented his method of sound analysis in a three-part article (Gabor 1946). He expanded the uncertainty theorem of quantum mechanics to sound signals, by using a complex representation of the signal, then proposed an "elementary signal"; a short term sound or pulse with an amplitude envelope; equivalent to a quanta in the physical theory. [A year later, this part was modified and appeared in another article (Gabor 1947) as a proposal of "acoustic quanta".] In part two, he conducted some experiments to illuminate the limitations and the range of human perception in listening for determining the size of grain. In the final part, he described the principles of frequency compression and expansion using his theories of "elementary signal" and "sliding window", and then their practical application; the frequency converter. This part became the

basis of asynchronous sound granulation, especially in its application to time-stretch and time-compression.

He concluded that the minimum duration of the "acoustic quanta" should be 10 msec with an amplitude envelope generated by the Gaussian method;

$$s(t) = \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt$$

A variation of this is a quasi-Gaussian curve that has a flat top between the rise and the delay of the original Gaussian curve (Roads 1978). In the case of analysis-synthesis, a Hanning window may be more convenient, hence its usage in the fast Fourier transform. Gabor also suggested line-segment envelopes for practical reasons in the original article (Gabor 1946). The shape of grain is discussed later in this thesis.

Gabor applied his theory of "elementary signal" for frequency conversion by means of a kinematical method. Using as an example the sound track of a film, he explained the relation between the sound frequency at the original film speed and at a faster one. [At this time, 1946, a sound track on a film was optically recorded and reproduced using a photocell behind the moving film, unlike the method which replaced it; a magnetic strip.]

The film moves across the fixed window, and the moving slit is effective while it traverses the window. To obtain a continuous sound, a second slit should appear at or just before the instant at which the first slit moves out

of the window, after that the third slit would appear, and so on. Assuming the window has a continuously graded transmission; full in the middle and fading out at both sides, abrupt clicks thus can be avoided. This is the basis of the kinematic frequency converter and can be applied for sound in any application, as Gabor described in his theory on "acoustic quanta". These hypothesis were verified mathematically by Bastiaans (Bastiaans 1980, 1985).

Gabor presented the proposal as a means of sound granulation for the purposes of signal transmission and signal conversion by "windowing". There are many other analysis-transformation techniques developed in the field of digital signal processing that can also be used for granular synthesis. For example analysis and synthesis systems such as wavelets or the short-time Fourier transform [STFT or SFT] supply a local representation of the signal, by means of grains or wavelets multiplied by coefficients, that provides another theoretical foundation for granular synthesis such as pitch synchronous granular synthesis.

6.1.2. Past Implementations of Granular Synthesis/Sound Granulation

Granular synthesis has proved quite attractive to a number of composers, in particular for the power and flexibility it offers for the tumbrel transformation of sampled sounds, and also its conceptual simplicity: small fragments of sounds are superimposed to construct more complex sound material. Various composers have explored this technique since the early 1970's, inspired by researchers such as Barry Truax and Curtis Roads.

Xenakis developed the technique initially in an analogue electronic domain. He considered the grains as "music quanta", and suggested a method of composition based on the organisation of these elements in terms of graphic projections of the key parameters of frequency and amplitude (Xenakis 1971).

Roads implemented granular synthesis using a computer that allowed a greater accuracy of control over the organisation of grains (Roads 1978). He suggested a high-level organisation of grains based on the concepts of tendency masks, or "clouds" in Xenakis' definition, implemented in the time-frequency domain.

Truax also applied the technique as a development of his own work with tendency masks applied to random-generated spectra, in due course extending usage of "acoustic quanta" for the granulation of sampled sound in real-time (Truax 1988). The latter involves a process of stretching and compressing the sound in a manner identified as variable-rate time shifting.

There are several software programs now available for non-real-time granular synthesis, such as CSOUND (Lee 1995). Due to the nature of the synthesis method, however, composers generally prefer to interact with the control parameters, thus requiring a real-time implementation. Despite the conceptual simplicity, the signal processing requirements of this application, especially in real-time, set tough demands for both hardware and software.

Most of the established working systems which are not only real-time, but also fully interactive in terms of user control, have been developed using custom-designed hardware (Truax 1994) or a purpose-built "music workstation"; such as IRIS-MARS (de Tintis 1995) and IRCAM Signal Processing Workstation (Lippe 1993).

As a part of our Music Technology Group's on-going investigations into real-time audio synthesis using a multi-processor network, granular techniques have provided an interesting challenge in terms of devising and mapping suitable algorithms onto a parallel architecture.

6.1.3. Terminology for Granular Synthesis / Sound Granulation

There are a number of different streams of research in this field, and each of them uses slightly different terminology. To clarify the problem, I define the following terms to be used in this thesis.

"Granular Synthesis" is a sound synthesis technique using Gabor's acoustic quanta theory. In the narrower definition, mainly used by Xenakis and Roads, the technique is referred to as sound modification using short grain models that change the texture of the source sound. "Sound Granulation" may be included in "Granular Synthesis" in the wider context. In the narrower interpretation, however, the method could be referred to as employing long grain models to maintain the source sound.

The main differences are;

	Granular Synthesis	Sound Granulation
Source Sound	synthetic sound	synthetic and natural sound
Grain Model	short less than 10 msec	long more than 10 msec
Effects	extra harmonics (side bands) caused by the granulation pitch change caused by the granulation and its parameters	time-stretch/compress; preserving the original timbre

Table 6.1.1.: Differences of "Granular Synthesis" and "Sound Granulation".

There are many similar techniques and effects in both methods, such as echo effects, enriching sound by overlapping sound streams, and sound spatialisation.

There are a few ways to describe the granulation parameters; some of them are dependent on individual systems and on the particular philosophy employed.

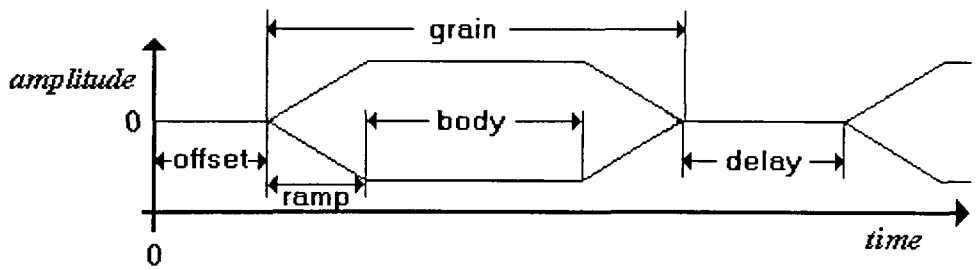


Figure 6.1.1.: Terminology for Grain.

delay	interval between the consecutive grains (interval, gap)
grain body	sustain part, between the ramps, of grain
grain density	number of grains per voice per second (gps)
grain ramp	rising and decaying part of grain
grain speed	= grain density
(initial) offset	timing until the first grain
voice / channel	a stream of grain

Table 6.1.2.: Terminology for Grain.

6.2. Related Applications

6.2.1. Wavelet Transform

Wavelet Transform was originally developed for applications in physics and acoustics (Dutilleux et al. 1988). A wavelet is a signal that forms a sinusoid with a smooth attack and decay. From a musical perspective, the wavelet transform can be considered as a special case of the constant *Q* filter paradigm. Wavelets inject the notion of a "short-time" or "granular" representation into the constant *Q* filter model. The transform represents and manipulates sounds mapped onto a time-frequency "plane" or "grid" that is also implicit in constant *Q* methods.

In the wavelet theory, every input signal is expressed as a sum of wavelets with a precise starting time, duration, frequency and initial phase. The peculiar aspect of the wavelet is that no matter what frequency it contains, it always encapsulates a constant number of cycles. The size of the wavelet window, therefore, can be expanded or compressed, according to the frequency being analysed.

Dilations and translations of the "Mother function" Φ(x), define an orthogonal basis:

$$\Phi_{(s,l)}(x) = 2^{-\frac{s}{2}} \Phi(2^{-s}x - l)$$

The variables *s* and *l* are integers that scale and dilate the mother function Φ to generate the wavelet. The scale index *s* indicates the wavelet's width, and the location index *l* gives the position. The mother functions

are re-scaled, or "dilated", by a power of two, and translated by an integer. The scaling function *W(x)* for the mother function Φ is given as;

$$W(x) = \sum_{k=-1}^{N-2} (-1)^k c_{k+1} \Phi(2x+k)$$

where c_k are the wavelet coefficients that must satisfy linear and quadratic constraints;

$$\sum_{k=0}^{N-1} c_k = 2 \qquad \sum_{k=0}^{N-1} c_k c_l = 2\delta_{l,0}$$

where δ is the delta function and *l* is the location index. It is helpful to think of the coefficients $\{c_0, c_1, ..., c_n\}$ as a filter that is placed in a transformation matrix. The coefficients are ordered using two dominant patterns; one works as a smoothing filter, and the other brings out the "detail" information from the data. These two orderings of the coefficients are called a "quadrate mirror filter pair".

One of the major dissimilarities between Fourier transform and Wavelet transform is that individual wavelet functions are localised in space whereas that of Fourier sine and cosine functions are not. This localisation feature along with a wavelet's localisation of frequency make many functions and operators using wavelets "sparse" when transformed into the wavelet domain. This sparseness, in turn, results in a number of useful applications such as data compression.

6.2.2. Pitch Synchronous Granulation

Pitch-synchronous granular synthesis is an analysis-synthesis technique that is designed for granulation of pitched sounds with one or more formant regions in their spectra. The technique starts from a spectrum analysis that is divided into significant time-frequency areas, each of them corresponding to a grain. The pitch detection is followed by the re-synthesis that consists of a train of pulses at the detected pitch. At each grain time frame, the system emits a waveform that is overlapped with the previous grain to create a smoothly varying signal. An implementation of the technique features several transformations that can create variations of the original sound (de Poli and Piccialli 1988).

6.2.3. Quasi-Synchronous Granulation

Quasi-Synchronous Granulation is a technique that creates sounds by means of generating one or more streams of grain, one grain following another with a variable interval. The technique is called "quasi-synchronous", since the grains follow each other approximately at equal interval.

If the gaps between successive grains are constant, the overall envelope of a stream of grains forms a periodic function. Since the envelope is periodic, the generated sound can be analysed as a case of Amplitude Modulation [AM] that occurs when the shape of one signal [modulator] determines the amplitude of another [carrier]. The result created by the modulation effect of the periodic envelope is that of formant surrounding the carrier frequency. The quasi-synchronous granulation, in this sense,

is similar to the formant synthesis method; for example, formant-wave-function synthesis known as FOF (Rodet 1984).

If the gaps between successive grains are irregular, perfect grain synchronisation is foregone along with its predictable side effects. Sounds are still created by one or more streams of grains with randomised onset timing to each grain that leads to a controllable thickening of the sound texture through a "blurring" of the formant structure (Truax 1988).

6.2.4. Asynchronous Granulation

In asynchronous granulation, the concept of linear streams of grain is abandoned, and the grains are scattered over specified regions inscribed on the frequency-time plane in the manner of a "cloud" (Xenakis 1971). A "cloud" is specified by the following parameters; start time and duration of the cloud, grain duration, density of grain per second, frequency band of the cloud, amplitude envelope of the cloud, waveform(s) in the grains, and spatial dispersion of the cloud. Later in this chapter and the following chapters, the effects of some of the parameters above are examined.

6.2.5. FOF

A "formant" is a peak of energy in the frequency domain that can include both harmonic and inharmonic partials. Formant regions serve as a kind of tumbrel cue to the source of many sounds. Formant wave-function synthesis [or in French; *Fonction d'onde formantique;* FOF] is the

basis of the CHANT sound synthesis system. CHANT has been implemented on various platforms; for example (Asta et al. 1980), and later, FOF generators have also been implemented in CSOUND (Clarke 1990). The basic sound model in CHANT is the human voice, as understanding the formant nature of human speech has long been a scientific goal.

FOF, the core of the CHANT system, originated from formant synthesis methods based on a traditional subtractive approach; a source signal with a broad bandwidth filtered by a complicated filter bank to a few resonant peaks. The complicated filters banks used in subtractive synthesis systems [see Chapter 1.2.] can be broken down to an equivalent set of parallel bandpass filters excited by pulses (Rodet et al. 1984). A filter such as *F* can be represented by its *z*-transfer functions:

$$H(z)=\frac{\beta_0+\beta_1 z^{-1}+\ldots+\beta_q z^{-q}}{1+\alpha_1 z^{-1}+\ldots+\alpha_p z^{-p}}$$

that include *p* poles and *q* zeros: this is linear prediction (Moorer 1977). This can be rewritten in another form;

$$H(z)=\sum_{i=1}^{j} c_i \frac{1+d_i z^{-1}}{1+a_i z^{-1}+b_i z^{-2}}.$$

The above form describes the *H* filter as a set of parallel *j* cells, each of these composed of a first-order filter and of a section of the second-order

series, with a gain c_i. The parameters *a* and *b* determine the centre frequency of a pass band and its local form, and d_i signifies the slope of the envelope.

When the excitation is a series of impulses;

$$E(k) = \sum_{-\infty}^{+\infty} e_n(k)$$

where *n* indexes the impulses in turn. The response *S* from the previous filter can be calculated as the sum of the responses $s_n(k)$ shifted from a period of the fundamental *T=1/F0*, where *F0* is the fundamental frequency of the excitation and the response. A response $s_n(k)$ is the sum of the *J* responses;

$$s(k) = \sum_{i=1}^{J} s_{(n,i)}(k)$$

where the $s_{(n,i)}(k)$ are called formant wave-functions [FOF], because they correspond with the formant or main modes of the response of the system. By changing the durations of the fundamental periods *T=1/F0*, the beginning of successive FOFs, variations of fundamental frequency can be realised. The characteristic of each FOF determines the envelope of the spectrum.

An FOF cell [or grain] is a damped sine wave with an attack and a quasi-exponential decay. The response to a unitary impulse of a cell;

$$c_i \frac{1+d_i z^{-1}}{1+a_i z^{-1}+b_i z^{-2}} = c_i \frac{1+d_i z^{-1}}{(1-r_i z^{-1})(1-r_i z^{-1})}$$

is the FOF

$$s_i(k) = G \times e^{-\alpha k} \sin(\omega k + \Phi)$$

with

$$\alpha = -\frac{1}{2}\log(b_i),$$

$$\omega = \mathrm{Arg}(r_i),$$

$$\Phi = \arctan\left\{\frac{\sin(\omega \cdot e^{-\alpha})}{d_i - a_i - \cos(\omega \cdot e^{-\alpha})}\right\},$$

$$G = \frac{c_i}{\sin(\Phi)}$$

where the *G* is the gain, the Φ is the initial phase of the formant. The local envelop is formally defined as follows;

$$\begin{aligned} k < 0 \qquad & s(k) = 0 \\ 0 \le k \le \frac{\pi}{\beta} \qquad & s(k) = \frac{1}{2}[1 - \cos(\beta k) \cdot e^{-\alpha k}] \\ k > \frac{\pi}{\beta} \qquad & s(k) = e^{-\alpha k} \end{aligned}$$

The range [$0<k<\pi/\beta$] is the attack, followed by the decay [$k>\pi/\beta$]. The ω is the central frequency of the response. The $\alpha\pi$ signifies the pass band width at -6 dB, and the length of the decay part. The β governs the pass

band width [or skirt width] at -40 dB, and the length of the attack part. Since the duration of each FOF cell usually lasts just a few milliseconds, the envelopes create audible side-bands around the source signal [the contents of the FOF cell; sine wave], creating a formant.

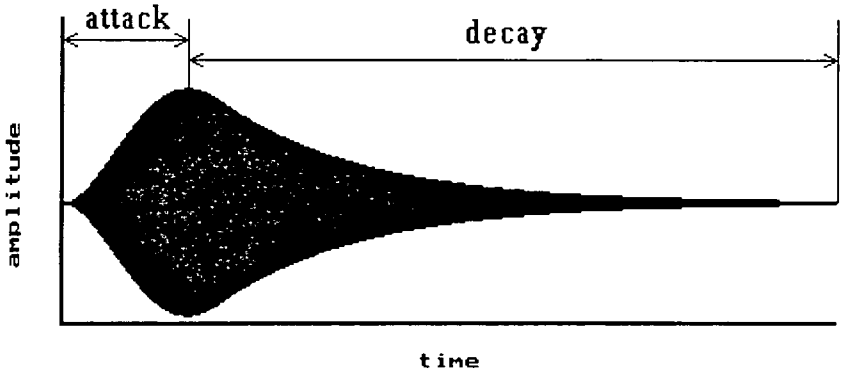


Figure 6.2.1a.: FOF Envelope.

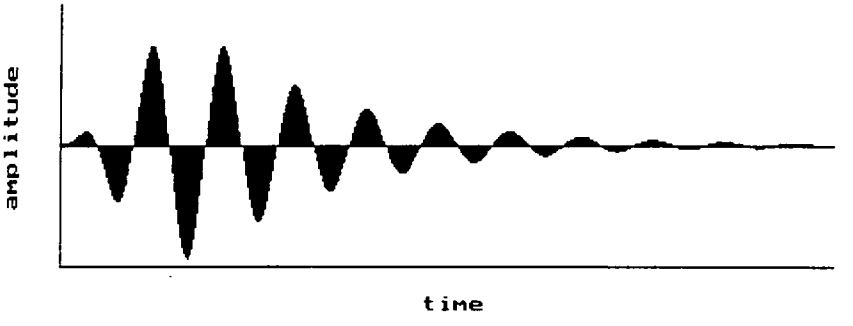


Figure 6.2.1b.: FOF Cell.

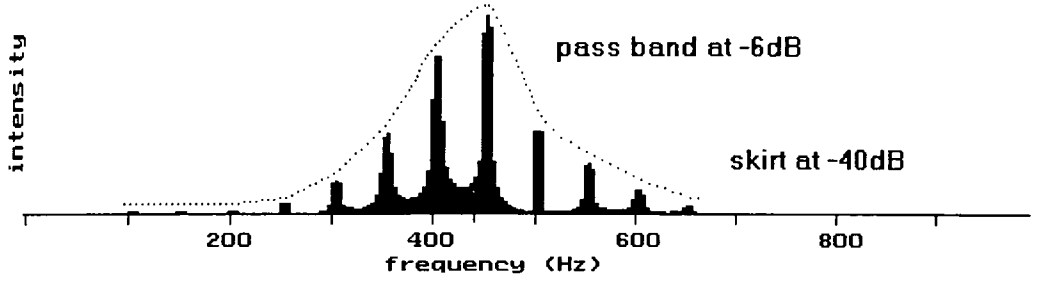


Figure 6.2.1c.: Frequency Response of FOF Cell.

Due to its time-domain nature, the FOF technique could be referred to as a pitch-synchronous granular synthesis technique.

An extended FOF generator is FOG. This technique is somewhat similar to some aspects of the asynchronous granular synthesis (Eckel et al. 1995). The main differences between the original FOF and the FOG are as follows;

	FOF	FOG
source sound	sine wave	arbitrary sound
trigger of formant	periodically	any

Table 6.2.1.: Major Differences between FOF and FOG.

6.3. Related Researches in Sound Granulation/Granular Synthesis

6.3.1. Granulation Systems in Simon Fraser University, Canada

The original granulation system, called PODX, was implemented on a DMX-1000 signal processor (Truax 1988). Sound samples are stored on a hard disk, then loaded into a 4k-byte window on the DMX. Unlike the TDS, the DMX has a high-speed hard disk and interface to support the real-time sound granulation and time-shifting which acts as a cache memory for the source sound communications with the DMX in real-time. All the granulation parameters [centre frequency, frequency range, offset, offset range, grain duration, duration range, and delay or density of grain] are controlled from the console keyboard of the host computer for the DMX, a PDP11/23. A variety of grain envelopes are available, by changing the length of ramp. For real-time user controls, a new value may be typed in for any parameter through the computer keyboard at any point.

The new system for granulation and time-shifting (Bartoo et al. 1994) is implemented over quad DSP56002 DSP chips [a total of 80 MIPS at 40 MHz] with a 68EC020 micro-processor hosted by an Apple Macintosh. One of the new features is that the sound sample which is being processed can be utilised as a signal processing effect in a mix-down environment, having a 256k to 16M-word [16-bit] audio sample memory at its disposal.

The system has eight channels of analogue input and output, digital control interfaces; SCSI and MIDI [IN and OUT]. The analogue input and the large audio sample memory make real-time feeding and processing of

a live source sound possible, whereas the PODX system and my transputer based system only accept digitised and pre-stored sound.

6.3.2. Granular Sampling on ISPW, IRCAM

The IRCAM Musical Workstation [IMW] was designed to facilitate real-time sound processing and interactive musical composition, and was based on one or more NeXT computers with between 2 and 24 Intel i860 processors [80 MFLOPS each at 40 MHz] as co-processors for sound controlling and one Motorola's DSP56001 chip for I/O processing; a serial port for digital audio at a 44.1 kHz sampling rate [CD quality], an RS-442 serial port for MIDI, another controller interface, and a DMA transfer between other NeXT boards through a NeXTBus (Puckette 1991, Lindemann, Starkuer and Dechelle 1990).

The same architecture called the IRCAM Signal Processing Workstation [ISPW] is described in an article by Lindemann (Lindemann et al. 1990). [In this thesis, I use the term ISPW to refer to the architecture, since most of the papers published after 1991 adopt this labelling.]

Lippe used the term "Granular Sampling" in his paper on the subject (Lippe 1993), however, the application he outlined is more appropriately described as "sound granulation". His system is based on Xenakis' description of the granular synthesis, using control valuables of pitch, envelope description, maximum amplitude, grain duration, rate of grain production [grain density], overlap of grain [delay] and spatial location of grain [pan].

The algorithm was implemented on an ISPW running under MAX, a sound processing language. MAX supports various inputs from MIDI instruments, including sliders and pedals, and the ISPW with dual i860 co-processors is able to handle a high rate of control information. Lippe suggested the usage of algorithmic mapping for the parameters. However, none of the details on controlling mechanisms nor their method are included in the paper.

6.3.3. Granulation System on IRIS-MARS Workstation

"Grains" is a system for quasi-synchronous granular synthesis implemented on the IRIS-MARS workstation (de Tintis 1995). Musical Audio Research Station, MARS, developed by IRIS s.r.l. [Italy] is a programmable and open system for real-time digital signal processing. MARS is based on a sound generation board SM1000; two IRIS X20 DSP chips [25.6 MIPS each at 25 MHz] for sound processing controlled by a Motorola 68302, and an integrated package EDIT20, supported by an Atari or a Macintosh, for graphical environment (Andrenacci et al. 1992). The former has a pair of MIDI ports [IN / OUT], an RS-232 interface, a parallel port, and a serial IIS port for communication.

There are four voices that are generated sequentially and controlled by five granulation parameters; frequency, grain length, waveform [source sound] and amplitude. Another parameter, horizontal density [grain density/speed → delay], determines when another grain should be activated. Since the end of one voice triggers another with or without

delay, it would appear that there is little scope for overlapping of two or more grains. Also, a Gaussian random number generator initiates the firing process, thus providing micro-modulation on the output sound.

Three of the parameters; grain length, amplitude and grain density, are generated with normal distribution in a range controlled independently through three MIDI sliders. The system can also be controlled by using a mouse through an Atari platform. In this case, only one parameter can be changed at a time, due to hardware restrictions. There are some features for a graphic interface to provide information on wave forms and envelopes, but these are not described in any detail. The system works in real-time at an unusual sampling rate of 39 kHz, slightly less than an audio standard of 44.1 kHz [CD quality].

6.3.4. Granular Synthesis on CSOUND

Granular synthesis using CSOUND could be achieved by applying conditional statements or a combination of existing unit generators, if a complicated and large "score file" and an associated "orchestra file" are provided. Recently, two new unit generators were developed to provide different levels of control specifically for granular synthesis (Lee 1995), and these can be integrated into normal CSOUND composition. Thus, source sounds can be obtained from existing CSOUND function tables, ranging from simple sine waves to sampled sound.

The control valuables for the grain generation unit are amplitude, grain size, gap [delay], pitch shift, attack [grain rising ramp size] and decay [grain decaying ramp size]. This suggests that grains created by the unit generator are not always symmetrical; the size of rising ramp can be different from the size of decaying one, where the conventional grain is symmetrical. All the parameters, except the amplitude, can be modified with a random number generated by another unit generator.

The other unit generator controls grains in a stream, by controlling the parameters of amplitude, skip [offset], grain size, gap [delay], offset, ratio [time-shift/stretch] and number of voices. This provides a multi-voice granular synthesis and sound granulation on CSOUND. This scheduling method allows grains to be overlapped in a voice by changing the gap parameter, whereas my system does not allow the overlap in a voice; the minimum delay is zero.

These unit generators have been tested on high-performance workstations like Sun and SGI, as well as conventional PCs. Due to the high demand in computation, the process however is quite time consuming, with only limited possibilities for real-time operation.

6.3.5. Granular Synthesis on SoundMaker

"SoundMaker" (Ricci 1997) is an offspring of a shareware sound application for Apple Macintosh; "SoundEffects". "SoundMaker" allows third-party developers to create new sound-manipulation modules in its environment using the plug-in paradigm. A plug-in module for sound

granulation; granular synthesis and time stretching, was published by Norris (1997).

The granulation parameters are; grain size [20 to 60 msec], grain distribution [linear synchronous or asynchronous by randomise], grain density [by percentage] and grain shape. A figure in the article suggests that a selection of grain shapes might be available, and the grain-body ratio seems to be variable. At the time of publication, the program works only on a non-real-time basis.

6.4. Time-Compression and Time-Stretching

Granular representation has made possible another powerful technique in sound processing; time-compression/stretching. This involves the granulation of the time-domain signal and re-synthesis of the grains with new time order. In 1946, Gabor built one of the earliest electro-mechanical time changers; "Kinematical Frequency Converter". The basic idea was time-granulation of recorded sound. A similar method is also used for slow-motion cinema and video images by repeating a frame a few times [see Chapter 6.1.1.].

In my digital implementation, recorded sound is stored on a 256k-byte external memory of a transputer in 16-bit integer format. Gabor's mechanical "sliding window" technique is modified with the wave-table digital synthesis method: a movement of the window turns into a change of read address.

A sine wave [440 Hz] was granulated with the following parameters:

number of channels	9
grain size	640 sample-long (160-320-160)
ramp	160 sample-long simple ramp
grain speed	3.7 grains per second per channel

Table 6.4.1.: Granulation Parameters.

The grain speed is chosen to ensure that a half of each grain, both ramp parts, is partially covered with the ramp segment of adjacent grains: the result being no interval between grains in the overall stream. The initial offset values for each channel were specified to match the overlap

condition above: 480 samples offset per channels with the exception of the first channel. The increment value for the address generator was selected for x2 time compression [sound sample 6.4.1.]; x2 grain size and x2 time stretch [sound sample 6.4.2.]; half the grain size. [All the figures of FFT results in this Chapter, both frequency and intensity axes, are on a linear scale.]

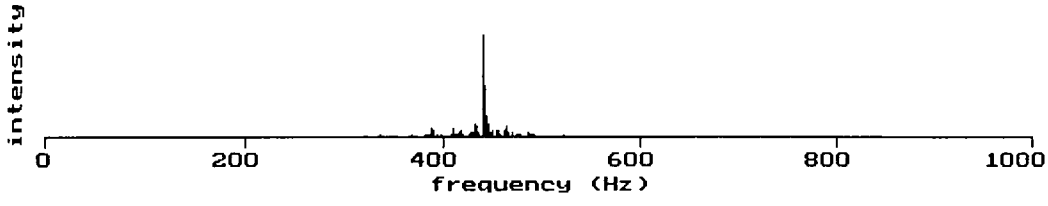


Figure 6.4.1.: Frequency Response of Granulation (x2 compression).

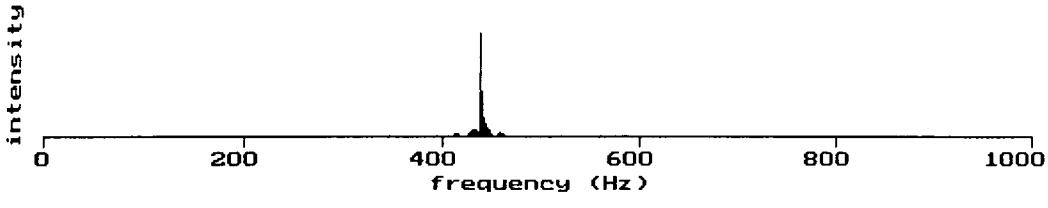


Figure 6.4.2.: Frequency Response of Granulation (x2 stretched).

These FFT results show a strong peak at the source frequency of 440 Hz and some low intensity surroundings. This confirms the theory of time shifting is correct.

The below figure shows the FFT result of the recorded speech element which was used for source sound of the granulation. The FFT was conducted using a 32,768 [$=2^{15}$] point-window, about 1 second long, that is shifted 15 times to cover the granulated sound. [The same sequence will be applied for the later FFTs of granulated natural sound, unless specified.]

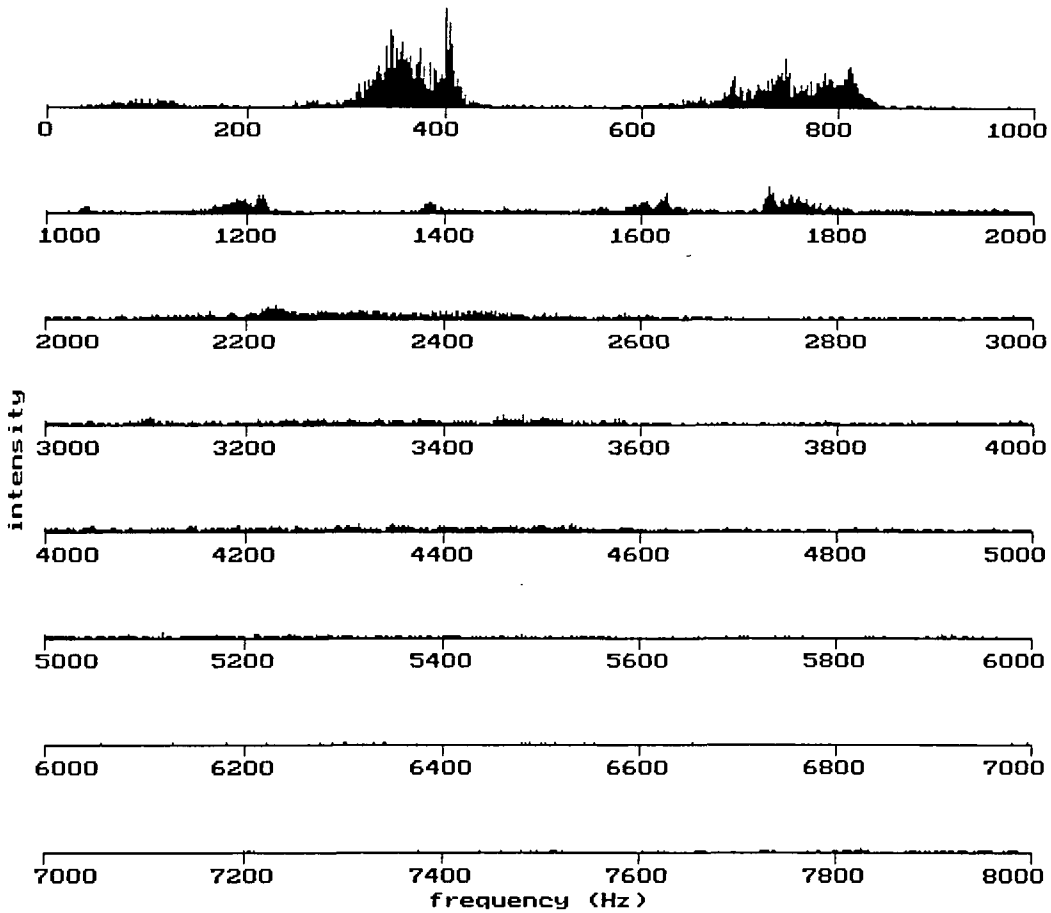


Figure 6.4.3.: FFT Result of Speech (32768 sample-long-window x 15).

Similar results are obtained from a granulation of speech with the same conditions. [sound sample 6.4.3.; original speech, 6.4.4.; compressed, 6.4.5.; stretched] Since the output-length of granulation was fixed for the convenience of the FFT program, there could be slight differences on these results. However, the overall shape of the result is similar to the original sound.

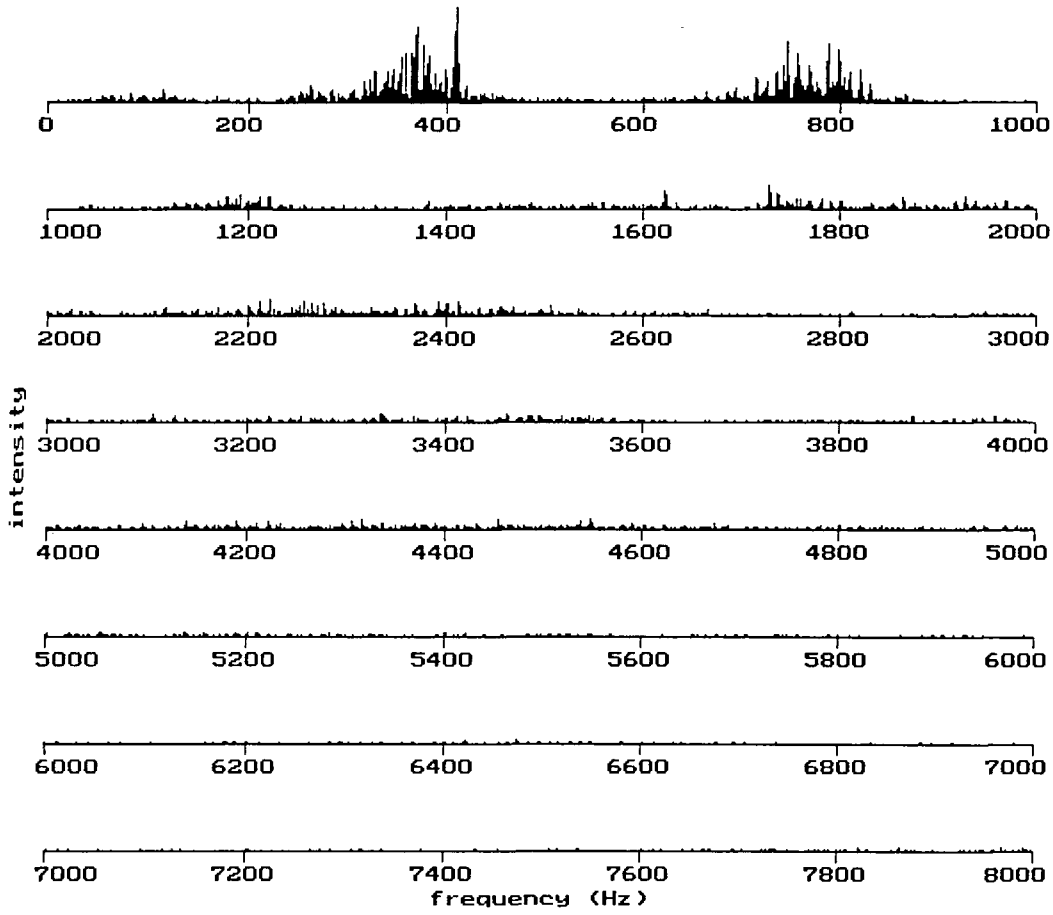


Figure 6.4.4.: Frequency Response of Granulation (x2 compression).

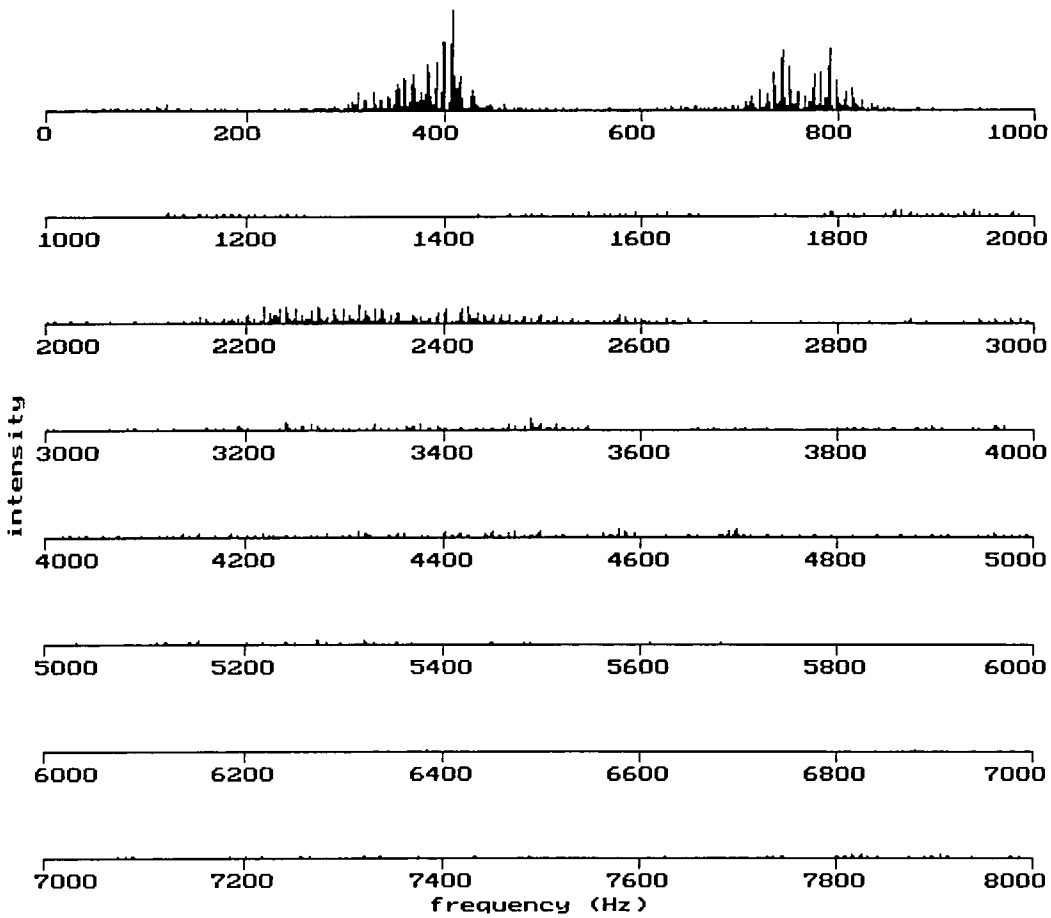


Figure 6.4.5.: Frequency Response of Granulation (x2 stretched).

6.5. Summary

The history of granular synthesis/sound granulation was reviewed together with similar applications, such as FOF and wavelet transformation. A number of implementations of granular synthesis and sound granulation methods were also studied. The technique of time-stretching and time-compression by sound granulation were also examined. In the next Chapter, the effects of the granulation and the granulation parameters will be investigated.

6.6. Further Applications

6.6.1. Granular Morphing and Spatialisation of Sound

Sound granulation could be used for smooth transition of sound, such as fade-in and fade-out, by changing grain density and other granulation parameters. Todoroff implemented the technique on an ISPW under a MAX environment, controlled by MIDI faders (Todoroff 1995). The system is capable of the fade-in/fade-out mode of two synchronised voices from 32 voices that operate at the maximum aggregated grain density of 2,500 gps, and offer voice spatialisation.

Multi-channel spatial distribution enhances the distinctive characteristics of granular synthesis and sound granulation. Granulated textures are articulated by scattering individual grains in different spatial locations. The perception of the spatial position of a stream of grain is determined by the physical properties of the signal and the "localisation blur" introduced by the human auditory system. The spatial distribution of the grains can be specified by distributing grains over the available channels, which are in turn panned across the stereo listening area.

6.6.2. Time Stretching for Language Teaching

Gabor's original idea of the sound granulation was meant for kinematic frequency conversion without changing the pitch (Gabor 1946). This may still have further applications for other subjects, such as language teaching and research. For example, language students are often required to hear recorded speech at a slower tape speed than the original. This results in a lower pitch than the original and distorted

tumbrel contents. Using the sound granulation technique, a recorded sound could be time-stretched without causing such distortions. The technique is also useful for measurements of timing in sound, since the time scale can be enlarged.

Chapter 7. Analysis of Grain (Granulation Parameters and Sound)

7.1. Shape of Grain

Gabor (1946) proposed granulation by a Gaussian amplitude envelope.

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} dx$$

A variation of this is the quasi-Gaussian curve that consists of a half Gaussian rise, sustain part and a half Gaussian decay that was used in Roads' system (Roads 1978). Truax (1994), however, applied a simple straight ramp. [Gabor also suggested it, for practical reasons (Gabor 1946).] Truax selected the ramp for his envelope for efficiency in terms of computation time and memory space: the calculation for a simple ramp requires a few operations per sample, an overhead that is quite important for "real-time" granular synthesis if the envelope is re-calculated for each grain.

An investigation into the effects caused by grain shapes was conducted. Four sine waves [27.5, 55.0, 110 and 440 Hz] were granulated by four different amplitude envelopes; simple-ramp, half-cosine, parabolic and quasi-Gaussian. [Only 27.5 Hz and 440 Hz examples are shown below.] Then the granulated signals were analysed by 65,536 point-FFTs, about two seconds in sound length. The sine waves, the envelopes and other calculations were performed in 32-bit floating point format. These figures show the granulated wave based on 440 Hz sine waves and the frequency response of the granulation [sound samples 7.1.1., 7.1.2., 7.1.3. and 7.1.4.].

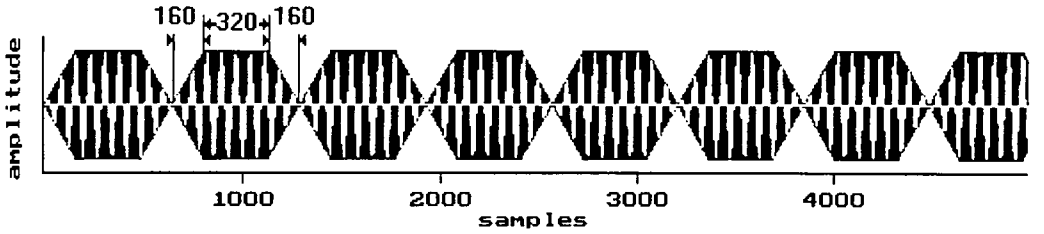


Figure 7.1.1a.: Granulation of 440 Hz Sine Wave by Ramp Envelope.

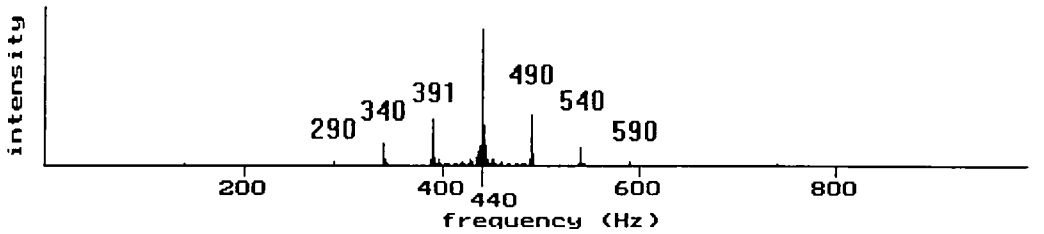


Figure 7.1.1b.: Frequency Response of Grain (ramp, 440 Hz).

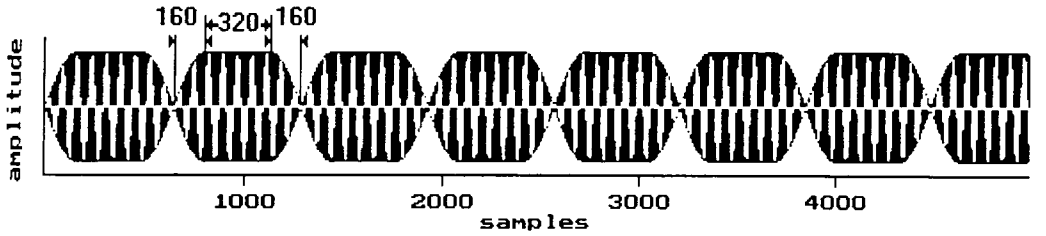


Figure 7.1.2a.: Granulation of 440 Hz Sine Wave by Half-Cosine Envelope.

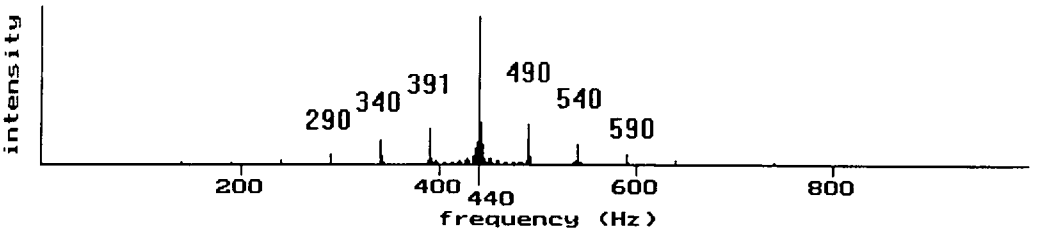


Figure 7.1.2b.: Frequency Response of Grain (half-cosine, 440 Hz).

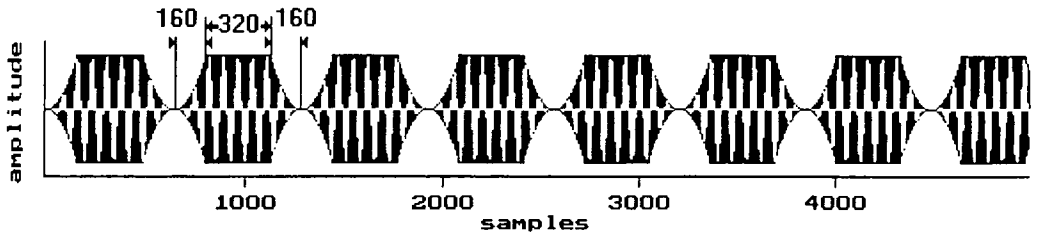


Figure 7.1.3a.: Granulation of 440 Hz Sine Wave by Parabolic Envelope.

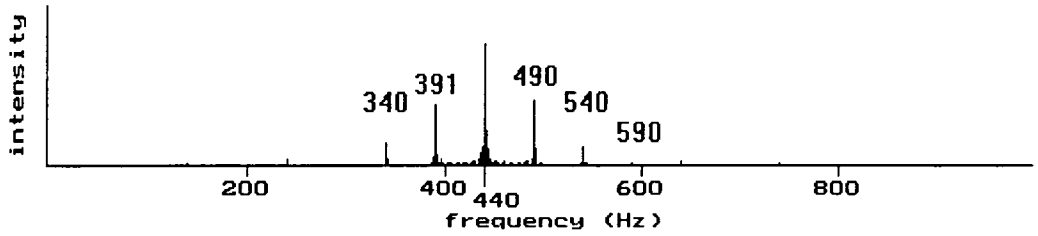


Figure 7.1.3b.: Frequency Response of Grain (Parabolic, 440 Hz).

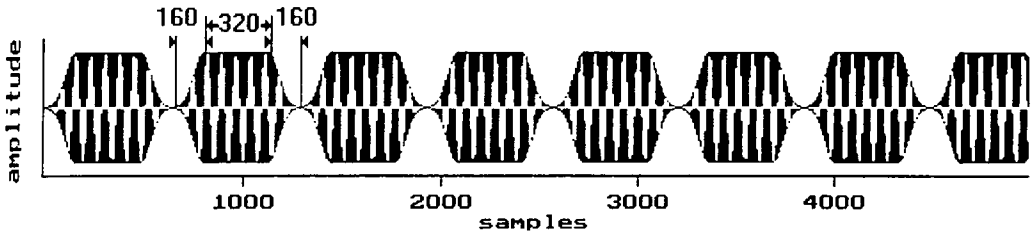


Figure 7.1.4a.: Granulation of 440 Hz Sine Wave
by Quasi-Gaussian Envelope.

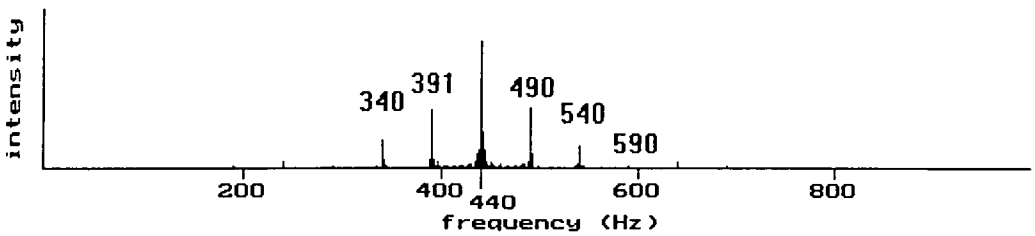


Figure 7.1.4b.: Frequency Response of Grain (Quasi-Gaussian, 440 Hz).

A few extra harmonics appear on all the FFT results, showing the effect of amplitude modulation caused by the granulation; a grain size of 640 samples [50 Hz at 32 kHz sampling rate].

The results below are from the granulation of a 27.5 Hz sine wave.

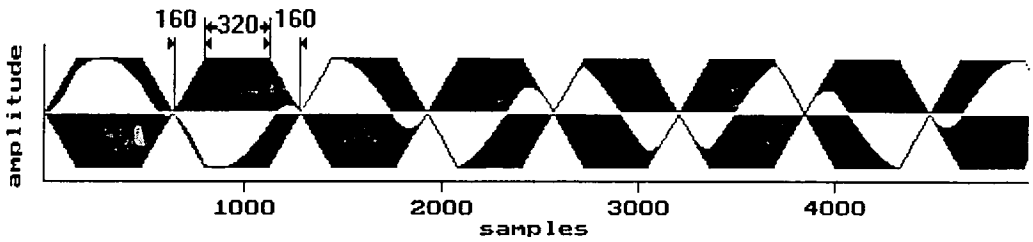


Figure 7.1.5a.: Grain Shape (ramp, 27.5 Hz).

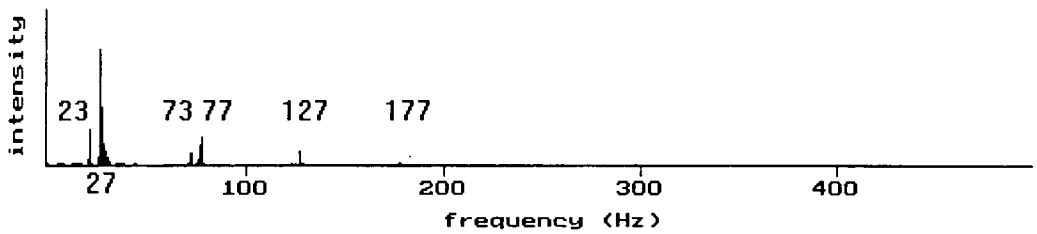


Figure 7.1.5b.: Frequency Response of Grain (ramp, 27.5 Hz).

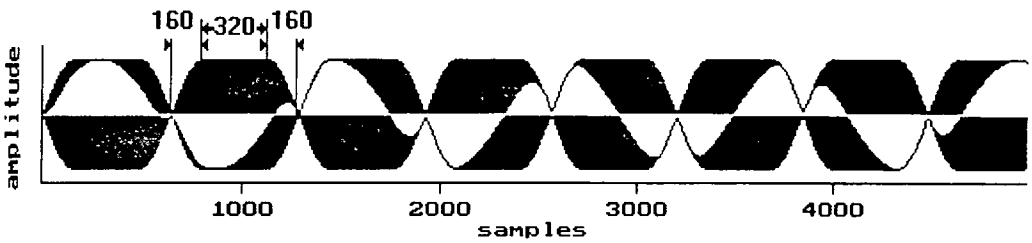


Figure 7.1.6a.: Grain Shape (half-cosine, 27.5 Hz).

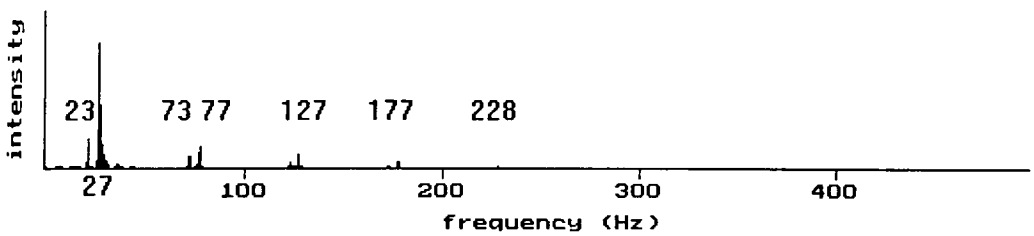


Figure 7.1.6b.: Frequency Response of Grain (half-cosine, 27.5 Hz).

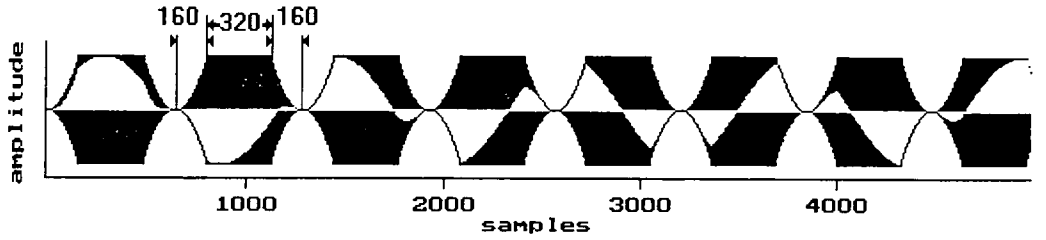


Figure 7.1.7a.: Grain Shape (parabolic, 27.5 Hz).

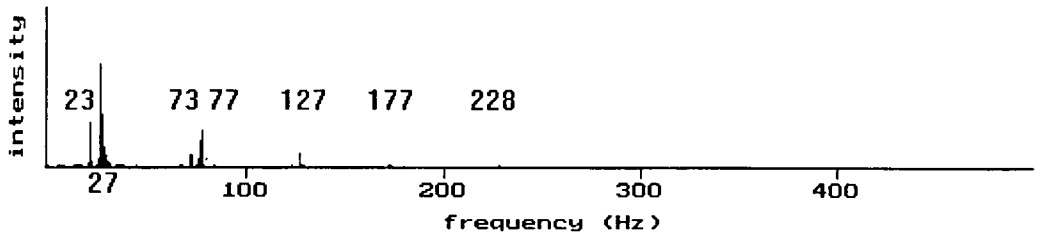


Figure 7.1.7b.: Frequency Response of Grain (parabolic, 27.5 Hz).

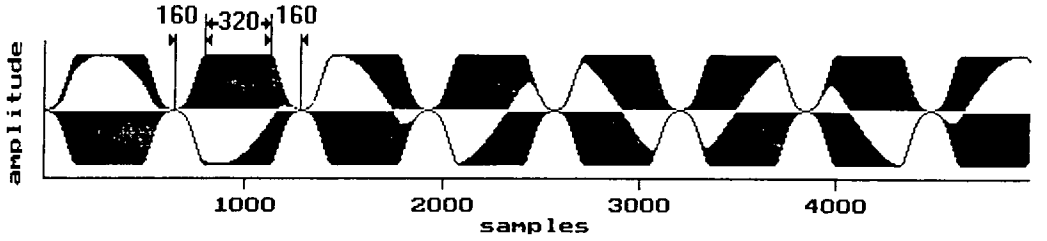


Figure 7.1.8a.: Grain Shape (Gaussian, 27.5 Hz).

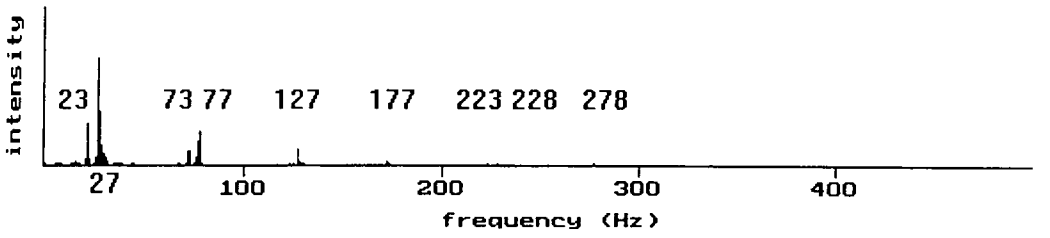


Figure 7.1.8b.: Frequency Response of Grain (Gaussian, 27.5 Hz).

The result for the 27.5 Hz sine wave shows that in the frequency domain, there are some minute differences in high harmonic components and the intensity of the fundamental frequency. Some peaks are supposed to be reflections of negative frequencies: such as 23 Hz [27.5 - 50 = -22.5] and 73 Hz [27.5 - 100 = -72.5] where 50 Hz is the grain frequency [640 sample-long-grain in 32 kHz sampling rate]. Another example using a 440 Hz sine wave shows similar results without the reflections.

The following figures are the frequency responses of the granulation of a 440 Hz sine wave by a 1,280 sample-long model. The grain length was doubled in order to create more extra harmonics [or decrease the group frequency].

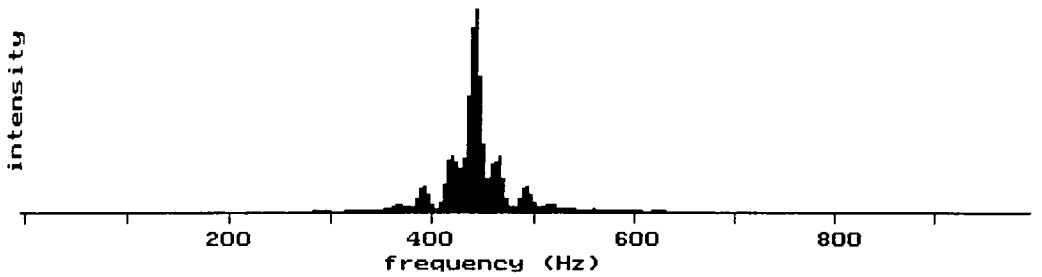


Figure 7.1.9a.: Frequency Response (ramp, long grain).

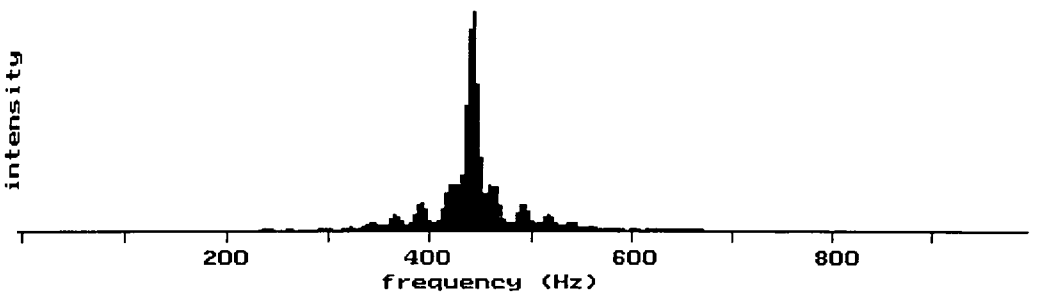


Figure 7.1.9b.: Frequency Response (half-cosine, long grain).

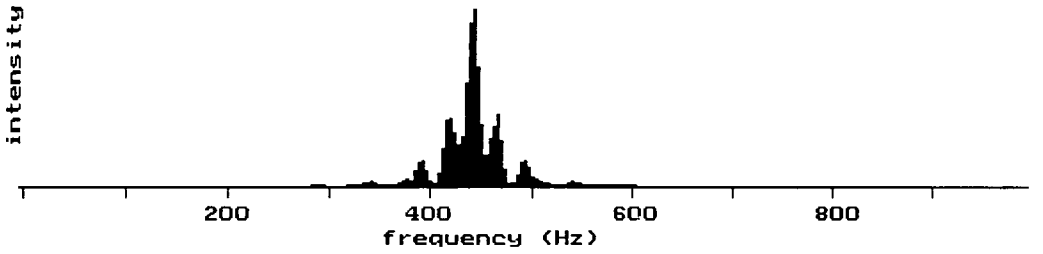


Figure 7.1.9c.: Frequency Response (parabolic, long grain).

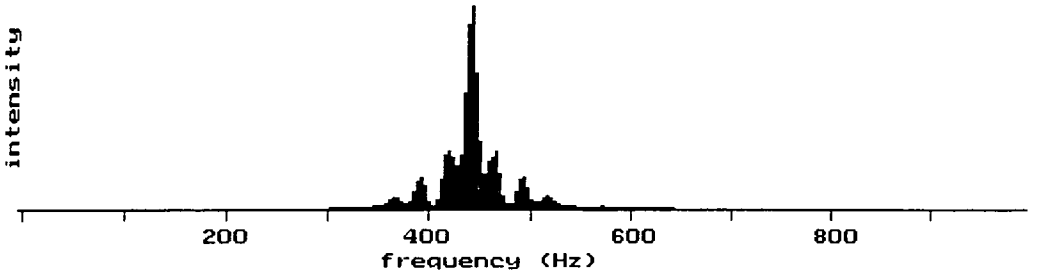


Figure 7.1.9d.: Frequency Response (Gaussian, long grain).

Generally, as Roads claims:

> A musically important grain parameter is the waveform inside the grain.
>
> Roads, C. 1991. "Asynchronous Granular Synthesis."

In the case of a long ramp, this may be expected to cause some differences in frequency characteristics; extra harmonics caused by the granulation will be concentrated around the strong peaks, since the grain frequency is low. It means that the modulation is the main effect and the grain shape is not. In the case of a short ramp, the grain frequency is high enough to create a wide range of harmonics.

A short ramp, however, means few steps between 0 and 1 which makes it hard to differentiate between one type of ramp and another. The shape of grain, therefore, does not have an important effect on the granulation. It is, however, clear that there are some differences in appearance and intensity of extra harmonic components that may significantly affect the timbre of the granulated sound.

7.2. Length of Grain

As an initial step towards the implementation of real-time granular synthesis, an 80-sample-long [2.5 msec at 32 kHz sampling rate] model was investigated. The model consists of a 20-sample-long rising ramp, a 40-sample-long sustain part and a 20-sample-long decaying ramp implemented onto a part of the 160 Transputer Network [see Chapter 8 for details]. A sine wave [440 Hz, "concert A"] was then granulated. The waveform, envelope shape and FFT result are shown below. [All the figures of FFT results in this Chapter, both frequency and intensity axes are in linear scales.]

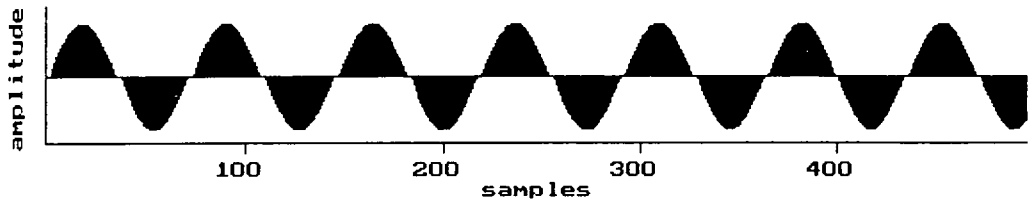


Figure 7.2.1a.: Source Signal (440 Hz Sine Wave).

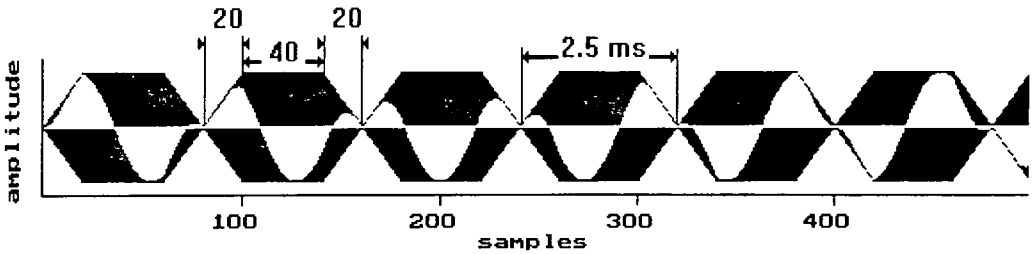


Figure 7.2.1b.: Granulation of 440 Hz Sine Wave by Ramp Envelope (80).

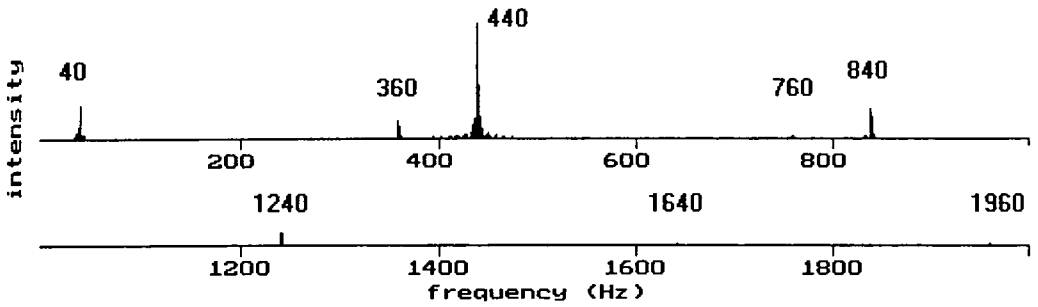


Figure 7.2.1c.: Frequency Response of Short Grain Model (80).

The frequency response of the grain model shows that there are a few extra harmonics created by the granulation. The grain frequency is 400 Hz [at 32 kHz sampling rate] such that modulation causes measurable upper-band-harmonics at 840 [440+400], 1240 [440+800] and 1640 [440+1200] Hz, and lower-band-harmonics at 40 [440-400], 360 [440-800; -360], 760 [440-1200; -760] and 1960 [440-2400; -1960] Hz.

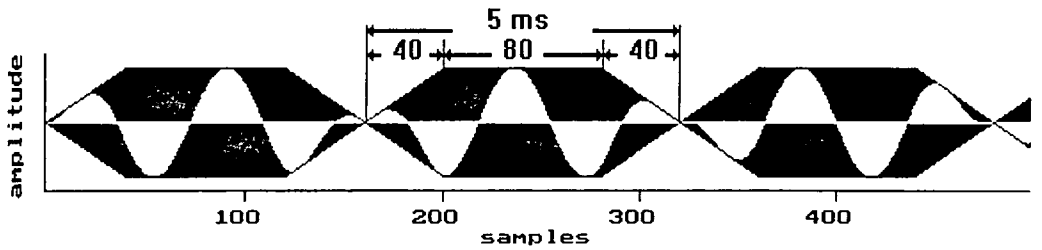


Figure 7.2.2a.: Granulation of 440 Hz Sine Wave by Ramp Envelope.

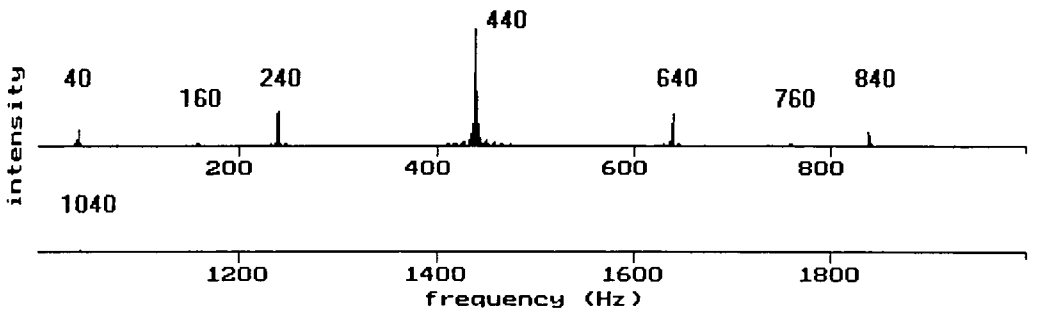


Figure 7.2.2b.: Frequency Response of Short Grain Model (160).

The grain length was doubled to 160-sample-long. This model creates a smoother sound but not smooth enough for sound reproduction, because of the wide gap between the ramp envelope steps. The source sound is heavily distorted by the extra harmonics. The granulated waveform, the envelope shape and an FFT result are shown above. The FFT result is quite similar to the shorter grain model, except the grain frequency is now 200 Hz.

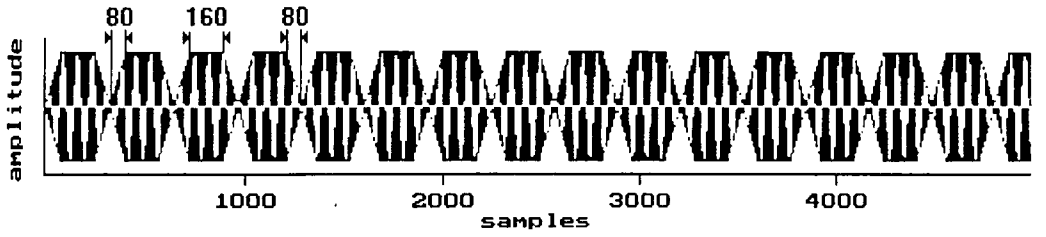


Figure 7.2.3a.: Granulation of 440 Hz Sine Wave by Ramp Envelope (320).

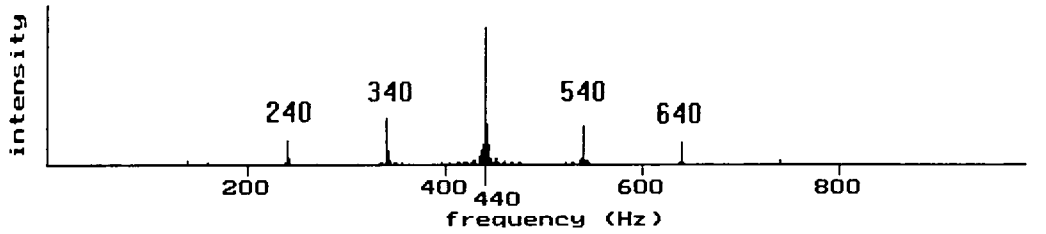


Figure 7.2.3b.: Frequency Response of Short Grain Model (320).

The grain length was again doubled to 320-sample-long [10 msec at 32 kHz sampling rate] and the same tests were conducted [sound sample 7.2.1]. The FFT result shows similar side-band-harmonics with 100 Hz, the grain frequency, as the spanning interval. These are weaker than the shorter models, but are still strong enough to affect the source sound. Gabor (1946) estimated a 10 msec-long grain model is the minimum size for sound granulation. This might be appropriate for a single sine wave granulation, but not for reproduction and time-stretching of natural sounds, since natural sounds have more than one peak in their spectrum characteristics that may interfere with each other's artefacts.

At this point, the 320-sample-long model was implemented onto a part of the 160 Transputer Network linked up to 9 external transputers that were configured for 9 channels of real-time sound granulation for recorded sounds [see Chapter 8 for details]. This enabled the use of recorded natural sounds, such as speech and random noise, for listening tests.

A single-stream granulation of the same recorded speech extract was performed with the 320-sample-long grain model [sound sample 7.2.2.].

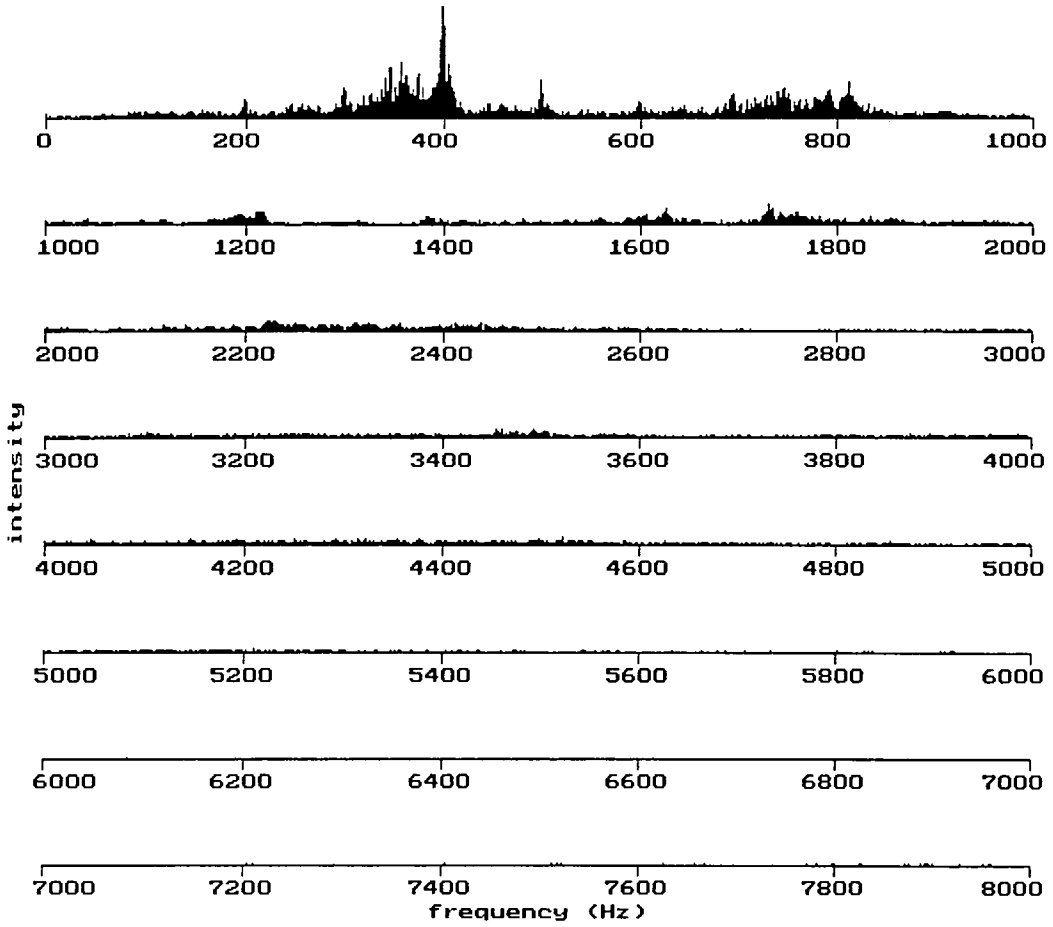


Figure 7.2.4.: FFT Result of Granulated Speech (320-sample-long grain).

The FFT result of the granulated sound above shows the strong peaks around 400 Hz and 800 Hz, but the surrounding frequency characteristics are changed [see Figure 6.4.3. for the source sound]. There are extra harmonics appearing around 200, 300, 500 and 600 Hz; -200, -100, +100 and +200 Hz where the grain frequency is 100 Hz. In the high frequency regions above 2,200 Hz, the peaks are diminished as if the granulation works like a low pass filter.

In terms of hearing, the granulated speech still contains some artefacts; distorted speech, like speaking with a sheet of paper held in front of the mouth. The sound is not so smooth, compared with the granulated A 440 sine wave using the shorter grain model, especially during periods when consonance parts are processed, a result of the short steep ramp envelope against high frequency components.

The grain length was then doubled to 640-sample-long; 20 msec at 32 kHz sampling rate, and granulated A 440 sine wave [sound sample 7.2.3.] and recorded speech [sound sample 7.2.4.] conducted using a single stream of grain envelope.

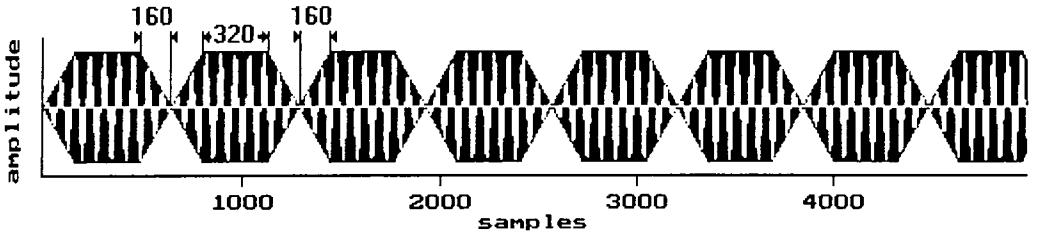


Figure 7.2.5a.: Granulation of 440 Hz Sine Wave by Ramp Envelope (640).

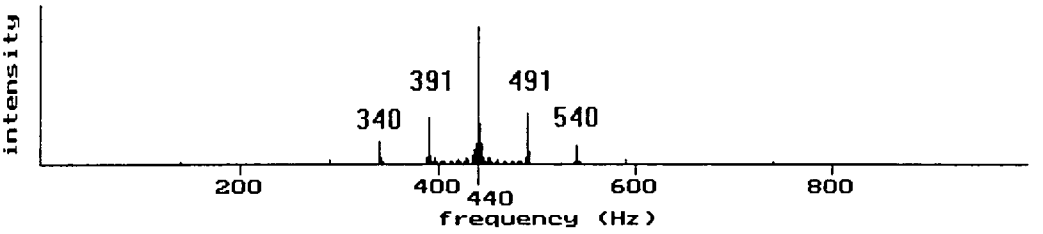


Figure 7.2.5b.: Frequency Response of Grain Model (640).

The result shows the similar effects of amplitude modulation with the grain frequency of 50 Hz. In the case of the granulation of speech, the sound became smoother, and the artefacts were weak enough for reproducing original sound.

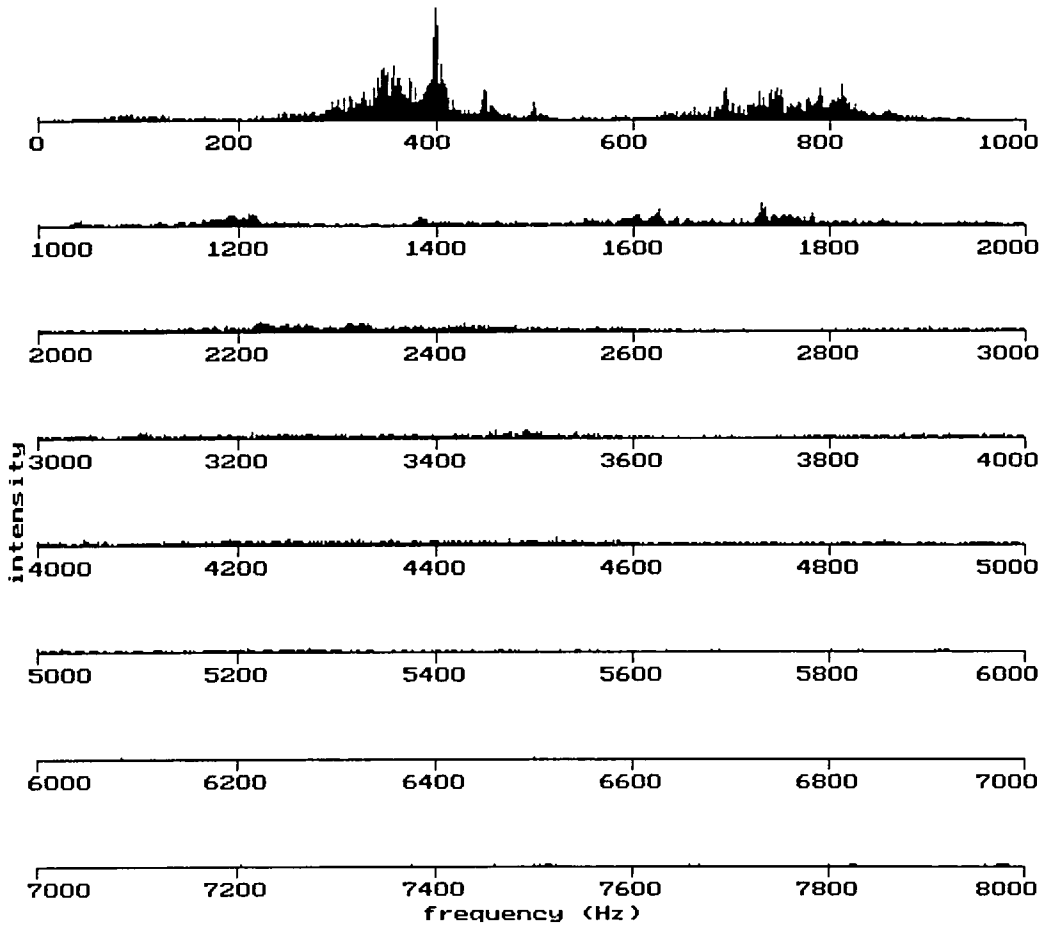


Figure 7.2.6.: FFT Result of Granulated Speech (640-sample-long grain).

The FFT result shown above also looks like the original without granulation. In the light of these assessments, I decided to use the 640-sample-long model as a standard reference and this is applied for the implementation of real-time granulation [see Chapter 8].

For further investigations into the size of grain, a longer grain model was tested: a 2,560-sample-long or 80 msec-long with a sine wave [440 Hz] [sound sample 7.2.5.] and the speech element [sound sample 7.2.6.]. The result of this frequency response analysis shows a high concentration of intensity at and around 440 Hz, the source sound, with amplitude

modulation of the grain where the grain frequency is 12.5 Hz at 32 kHz sampling rate. In terms of hearing, the granulated sound is a source sound with a low frequency amplitude envelope, *tremolo*, rather than a continuous sound.

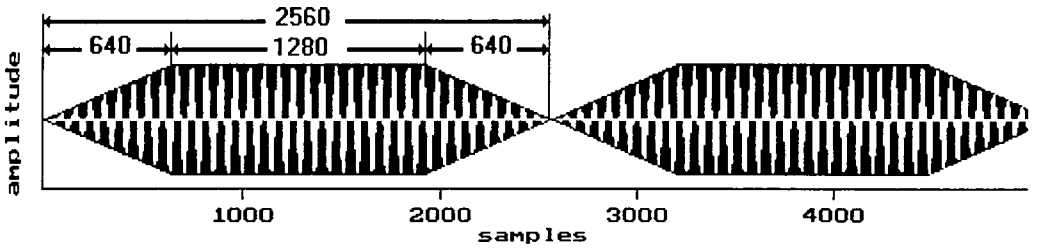


Figure 7.2.7a.: Granulation of 440 Hz Sine Wave by Ramp Envelope (2560).

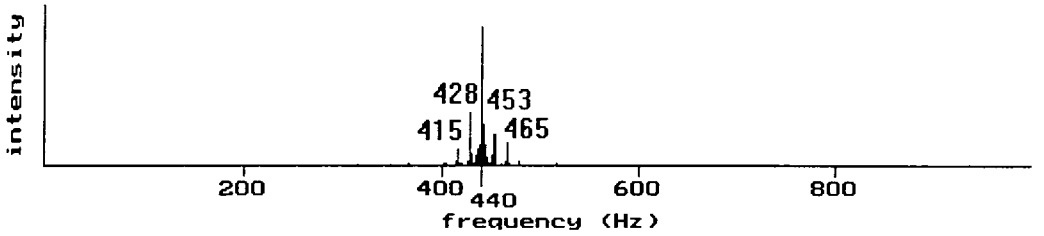


Figure 7.2.7b.: Frequency Response of Grain Model (2560).

These experiments suggest that a longer grain model can produce a smooth sound that closely approximates to the source sound in terms of frequency response. A longer grain model is able to reproduce the source sound better than a shorter one, since it creates narrow side-bands, and the intensity of each artefact is lower. In other words, the size of grain has some effects on the bandwidth of the granulated sound.

For the purposes of time-stretching and sound reproduction, a longer grain model would be preferred, since a longer grain can preserve the original sound texture. For sound modification, or "Granular Synthesis", a shorter grain model would be better, because of the wider side-bands and other

effects of a shorter grain. Gabor's estimation, a minimum of 10 msec-long grain, was thus confirmed. The minimum, however, might not work as well for some natural sounds with rapid change, like speech.

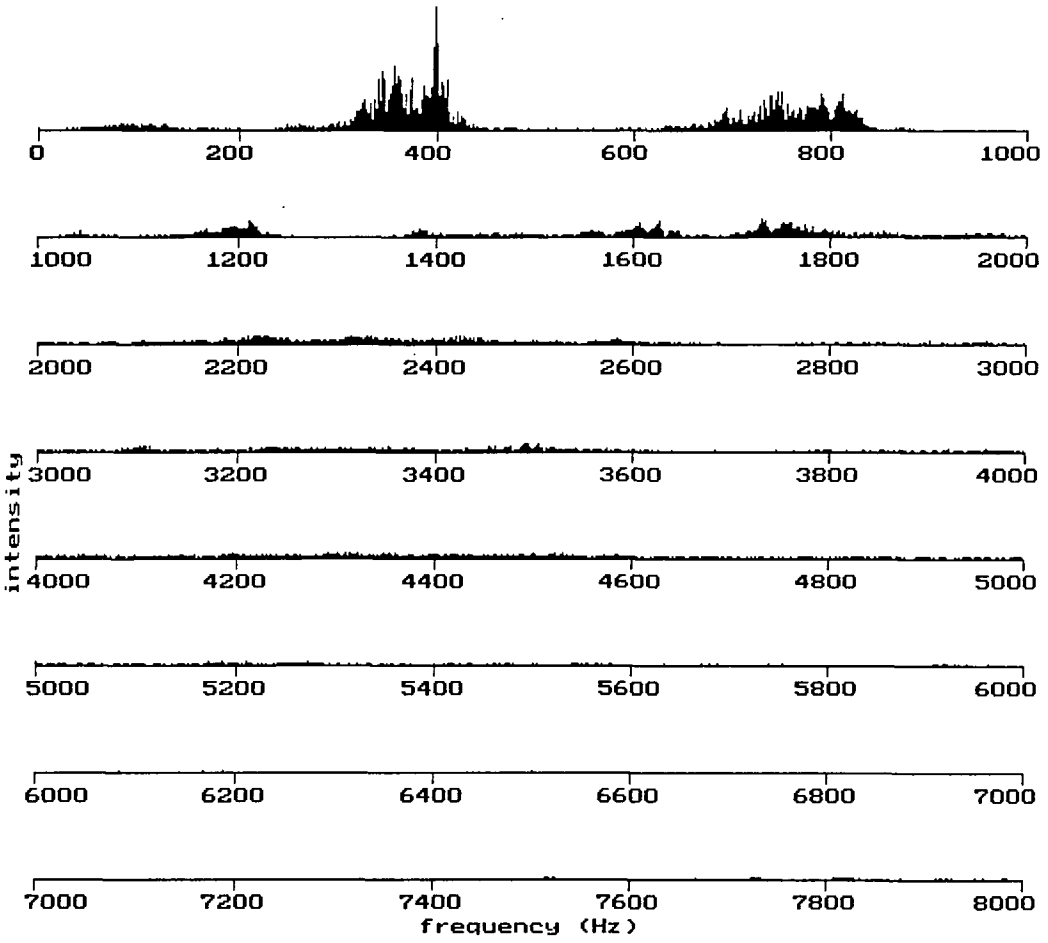


Figure 7.2.8.: FFT Result of Granulated Speech (2560-sample-long grain).

2,560-sample-long [80 msec] and longer grain models, that require a large memory to store the ramp coefficients, might not be suitable for a real-time implementation on a single transputer without external memory; for example a 640 sample-long ramp occupies 2.56k-byte out of the 4k-byte memory available on a networked transputer. In the case of a long grain model, it may be feasible to divide the granulation task implemented on a

transputer to two individual transputers; one generating ramp coefficients, instead of storing them, and the other multiplying the source sound with the coefficients.

7.3. Ramp-Body Ratio

Ramp-body ratio is another parameter for the control of sound granulation. The regular ramp-body ratio is one-to-two. In other words, a grain consists of a quarter-grain length rising ramp, a half-grain sustain part and a quarter-grain length decaying ramp.

For the investigation into the significance of the ratio, firstly, the size of the grain was fixed to about 640 sample-long. The effect of the granulation is the amplitude modulation, where the grain frequency is 50 Hz, and extra harmonic contents should appear every 50 Hz. Secondly, the size of the ramp is fixed to 160 sample-long and the size of the grain body is changed. This should provide results that exclude the effects from the artefact of short ramps. A 440 Hz ["concert" A] sine wave was used as the source sound of granulation in both cases.

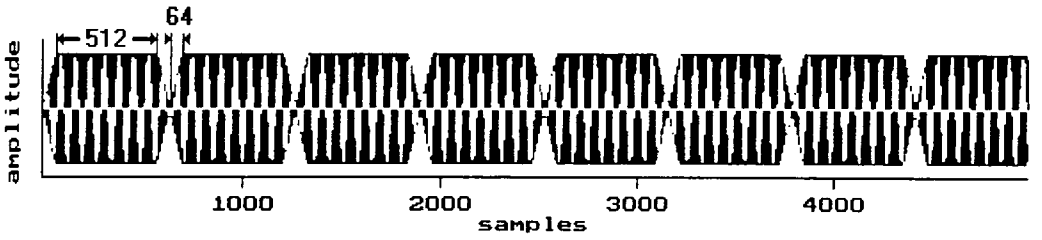


Figure 7.3.1a.: Grain Envelope and Granulated Sound (1:8).

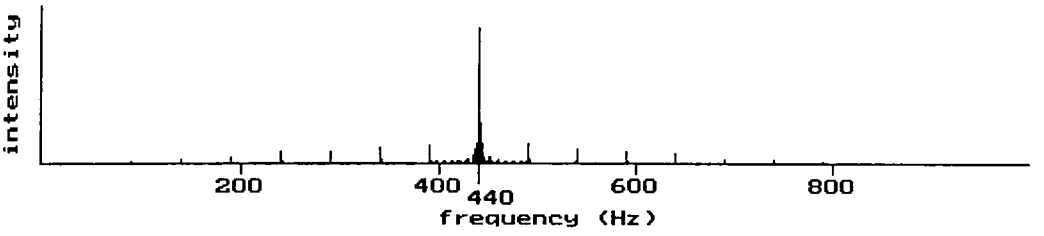


Figure 7.3.1b.: Frequency Response of Granulation (1:8).

The above graphs show the granulated waveform, the amplitude envelope and their frequency response to the 64-512-64 grain model, where ramp-body ratio is 1:8.

Extra harmonics in the upper-band appear every 50 Hz, the grain frequency, up to 740 Hz, and in the lower-band, down to 140 Hz. Their intensities are not so significant; less than 10% of the source frequency 440 Hz. In cases of 1:16, 1:32, 1:64 models, their results are quite similar to the 1:8 model except the deviation of the extra harmonics is wider.

The results of 1:4 and 1:2 models below are quite similar to the 1:8 model, except the deviation of extra harmonics becomes narrower and the intensity of first and second harmonics in both side-bands are higher.

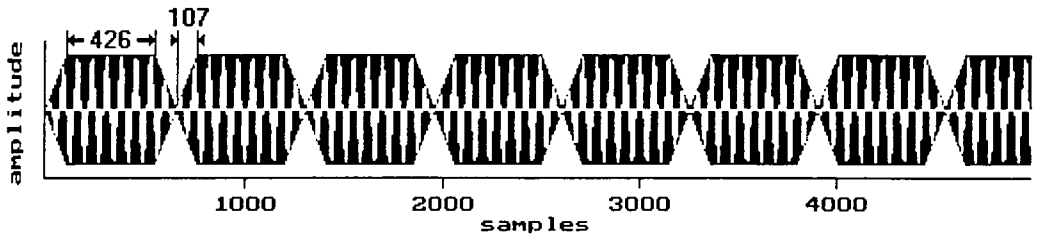


Figure 7.3.2a.: Grain Envelope and Granulated Sound (1:4).

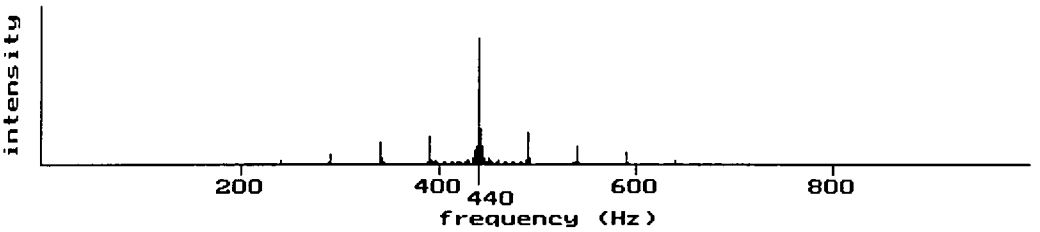


Figure 7.3.2b.: Frequency Response of Granulation (1:4).

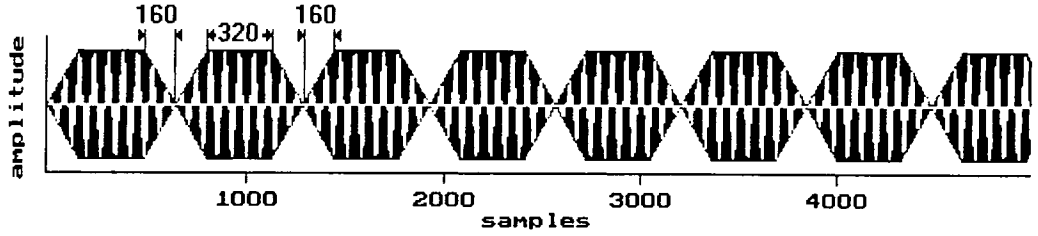


Figure 7.3.3a.: Grain Envelope and Granulated Sound (1:2, regular).

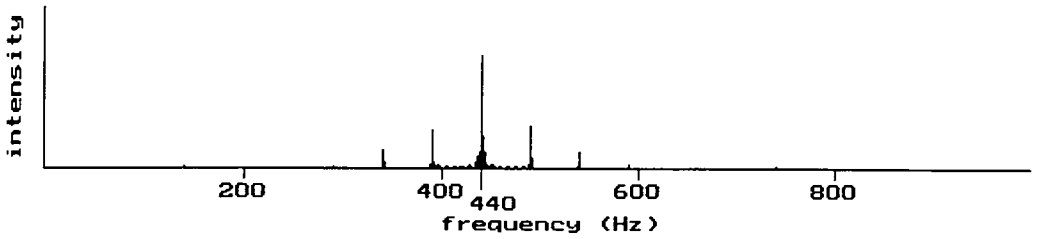


Figure 7.3.3b.: Frequency Response of Granulation (1:2, regular).

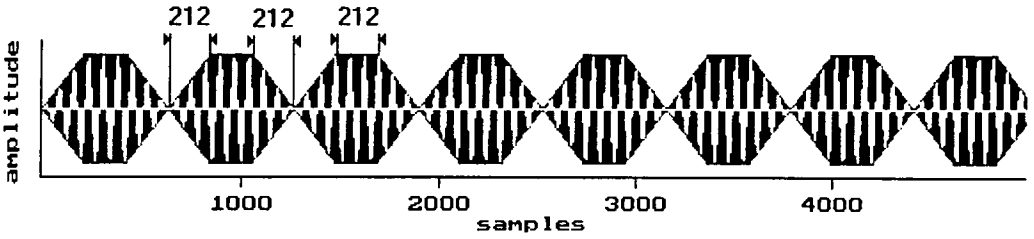


Figure 7.3.4a.: Grain Envelope and Granulated Sound (1:1).

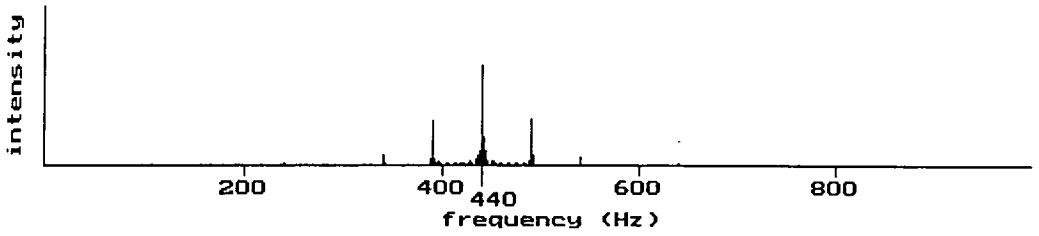


Figure 7.3.4b.: Frequency Response of Granulation (1:1).

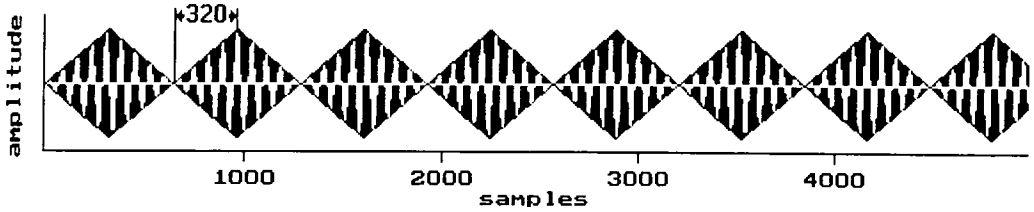


Figure 7.3.5a.: Grain Envelope and Granulated Sound (ramp only).

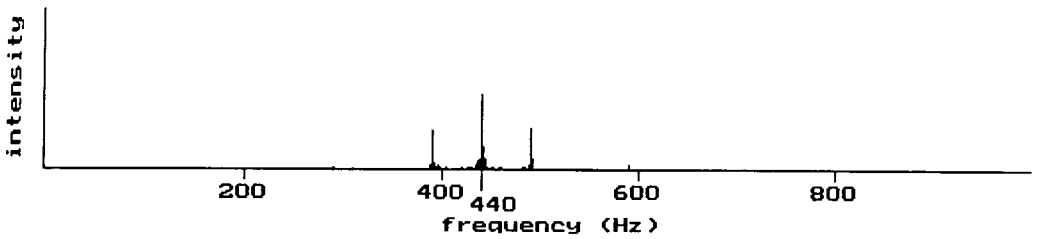


Figure 7.3.5b.: Frequency Response of Granulation (ramp only).

The above results are based on the fixed grain size analysis. To avoid effects from the size of ramp, the following analysis of the ramp-body ratio was conducted using a 160-sample-long fixed-ramp and variable sizes of grain body. It means that the grain size and the grain frequency are not constant.

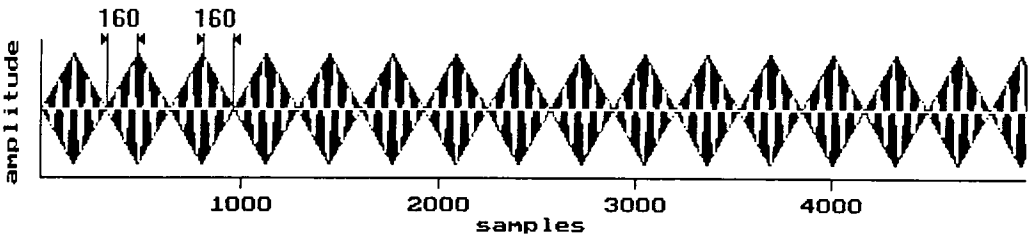


Figure 7.3.6a.: Grain Envelope and Granulated Sound
(fixed ramp, ramp only).

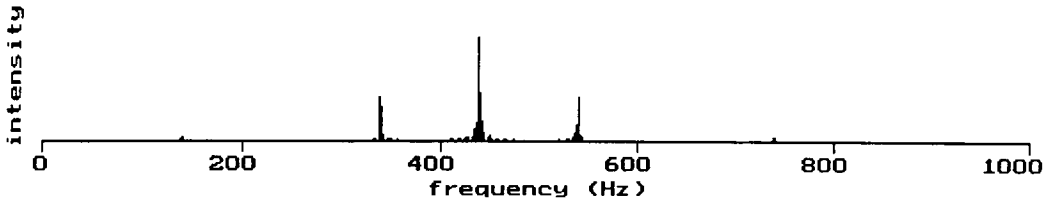


Figure 7.3.6b.: Frequency Response of Granulation
(fixed ramp, ramp only).

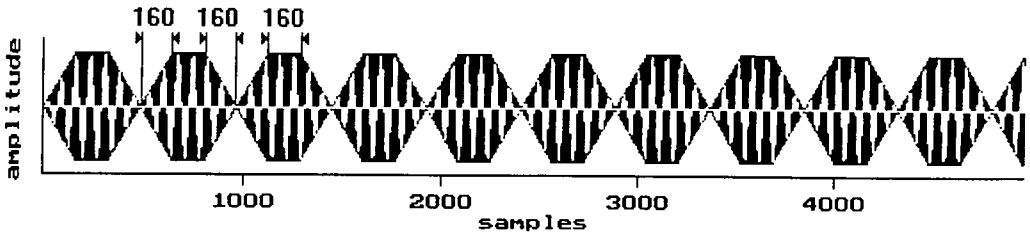


Figure 7.3.7a.: Grain Envelope and Granulated Sound (fixed ramp, 1:1).

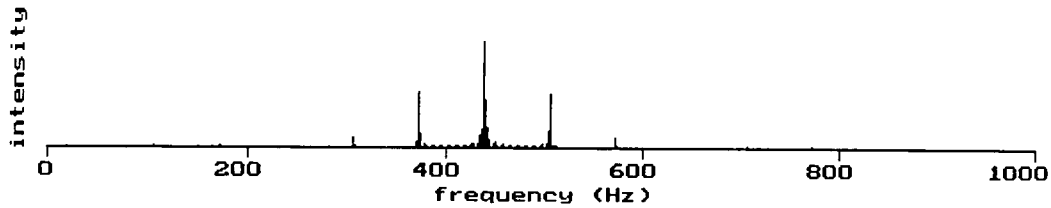


Figure 7.3.7b.: Frequency Response of Granulation (fixed ramp, 1:1).

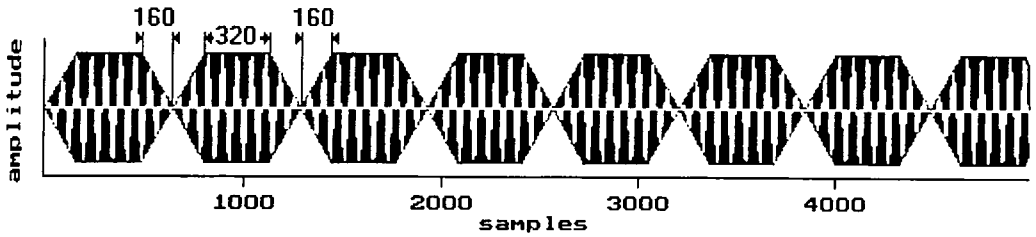


Figure 7.3.8a.: Grain Envelope and Granulated Sound (fixed ramp, 1:2).

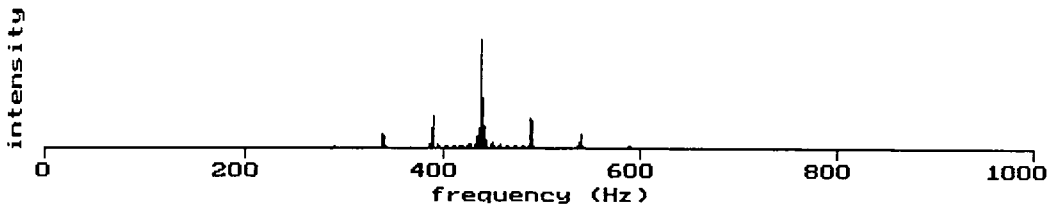


Figure 7.3.8b.: Frequency Response of Granulation (fixed ramp, 1:2).

The above examples show that the results are similar to the fixed grain-size models: a low body-ramp ratio means narrow deviation of extra harmonics and low intensity of the source sound frequency.

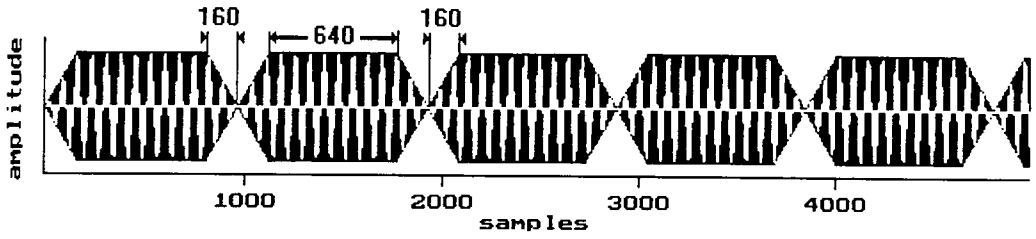


Figure 7.3.9a.: Grain Envelope and Granulated Sound (fixed ramp, 1:4).

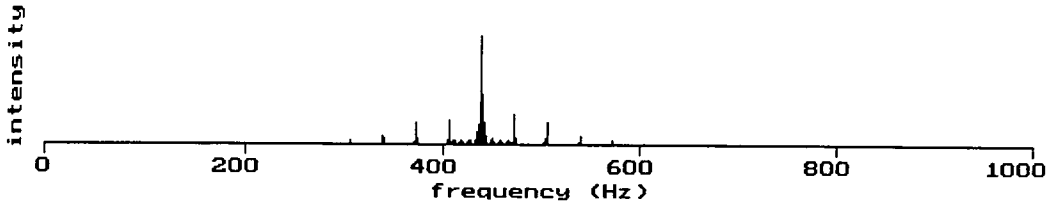


Figure 7.3.9b.: Frequency Response of Granulation (fixed ramp, 1:4).

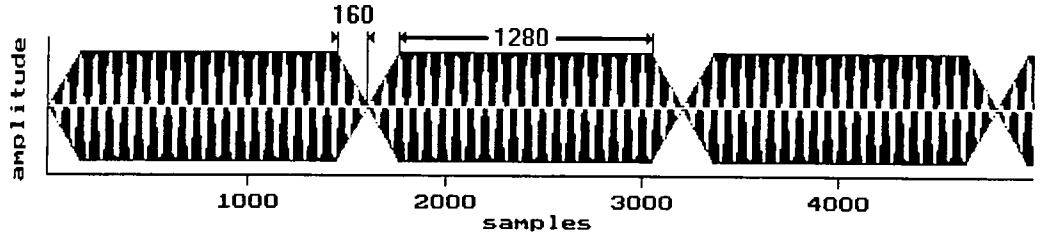


Figure 7.3.10a.: Grain Envelope and Granulated Sound (fixed ramp, 1:8).

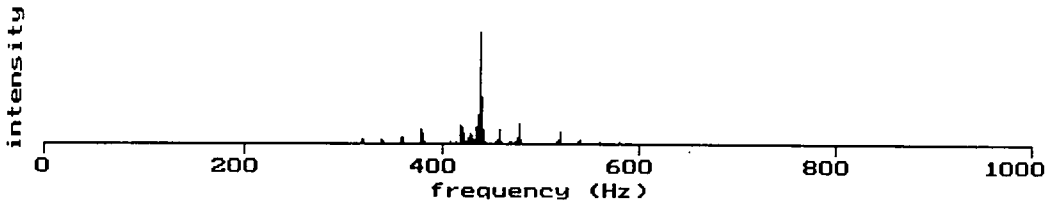


Figure 7.3.10b.: Frequency Response of Granulation (fixed ramp, 1:8).

The results of 1:8 and 1:16 models show a significant concentration of intensity at the source sound frequency; 440 Hz. Total intensities of extra harmonic contents are less than 10% of the main frequency. The 1:32 and the 1:64 model also show similar results. These results suggest that

changes in the ramp-body ratio effect the deviation of the extra harmonics and their intensities.

To confirm the reflection of "negative" frequencies, a 110 Hz sine wave was granulated with a standard 1:2 grain model. The side bands appear at an interval of 50 Hz [grain frequency] and are similar to those of the source frequency at 440 Hz.

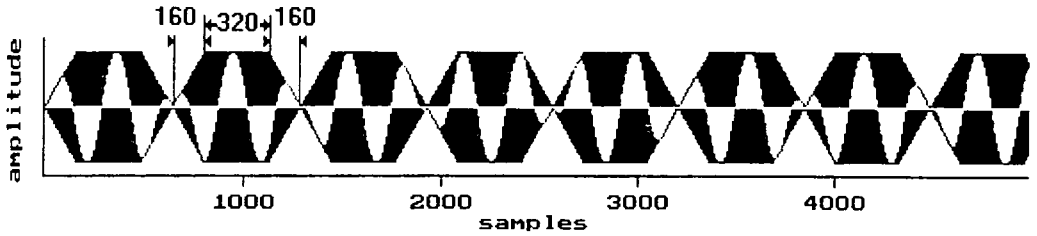


Figure 7.3.11a.: Grain Envelope and Granulated Sound (110 Hz, 1:2).

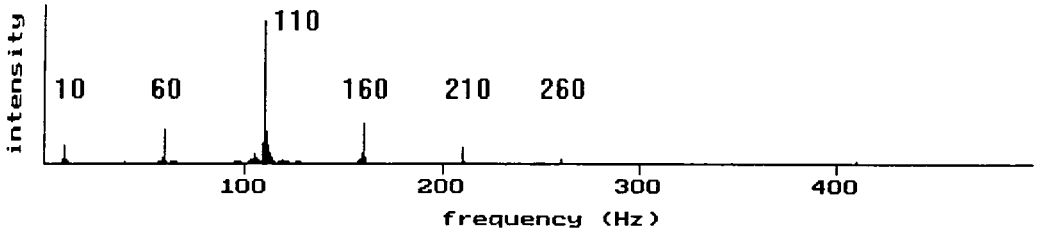


Figure 7.3.11b.: Frequency Response of Granulation (110 Hz, 1:2).

Ramp-body ratio governs the basic profiles of the side-band-width of the granulated sound, regardless of the size of the grain. Since extra harmonics appear at the interval of the grain frequency, which depends upon the size of the grain, the extra harmonics are controlled firstly by the body-ramp ratio and are multiplied by the grain frequency. For the efficiency of grain calculations, the fixed-ramp algorithm is applied for the real-time implementation [see Chapter 8].

7.4. Interval between Grains

Another main parameter for granular synthesis is the interval between the grains that is also called "delay" or "space". The grain-space ratio is usually defined backwards from "grain density" or "grain speed"; grain-per-second [gps]. The experiments in previous sections were conducted with a single stream of grains without interval. In this section, further investigations into the effects of a short interval between grains were conducted.

Using the standard 640-sample-long grain model, a sine wave [440 Hz] was granulated with various sizes of space, then its frequency characteristics were analysed. The intervals between the grains were as follows; 1/4 grain-long [160 sample-long], 1/2, 3/4, 1, 3/2 and 2 grain-long.

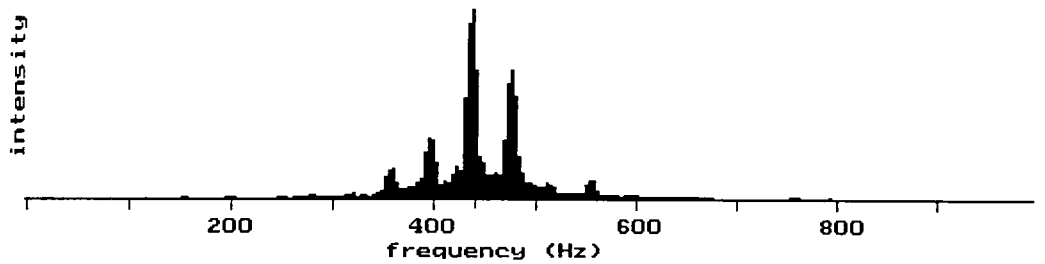


Figure 7.4.1a.: Frequency Response of Grain with Space (1/4).

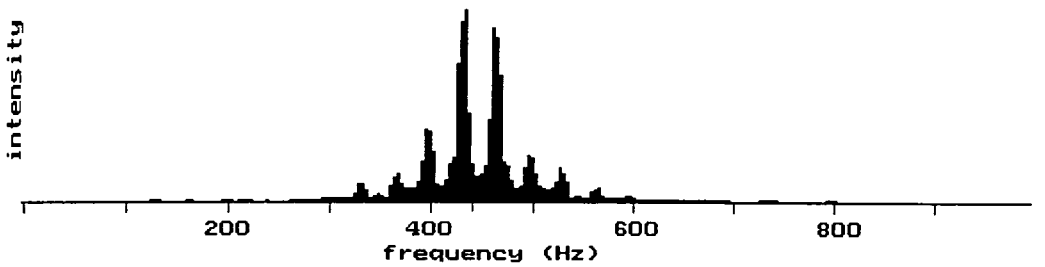


Figure 7.4.1b.: Frequency Response of Grain with Space (1/2).

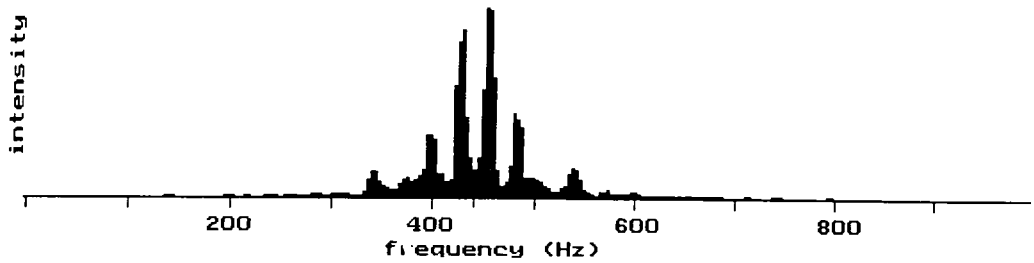


Figure 7.4.1c.: Frequency Response of Grain with Space (3/4).

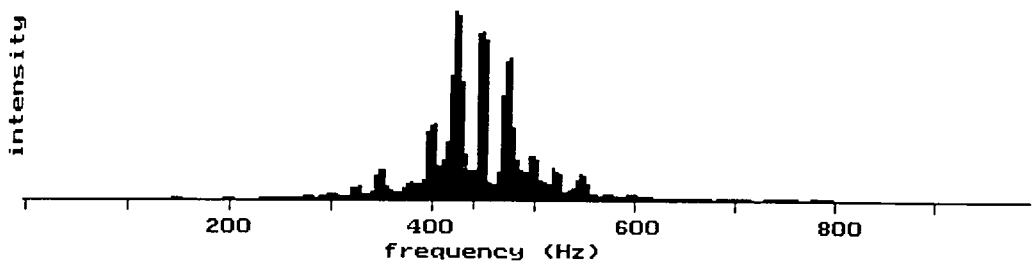


Figure 7.4.1d.: Frequency Response of Grain with Space (1/1).

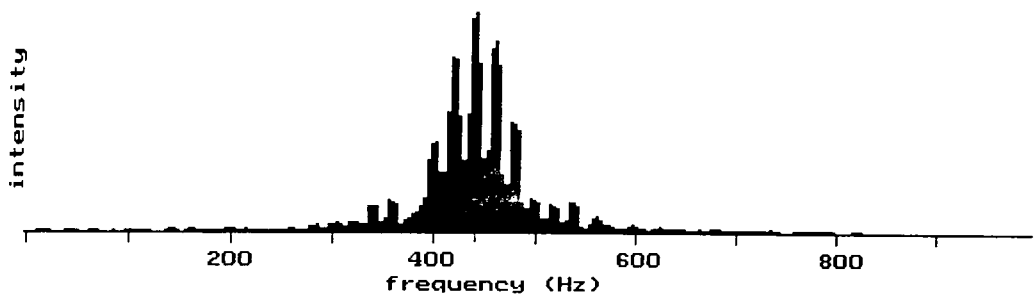


Figure 7.4.1e.: Frequency Response of Grain with Space (3/2).

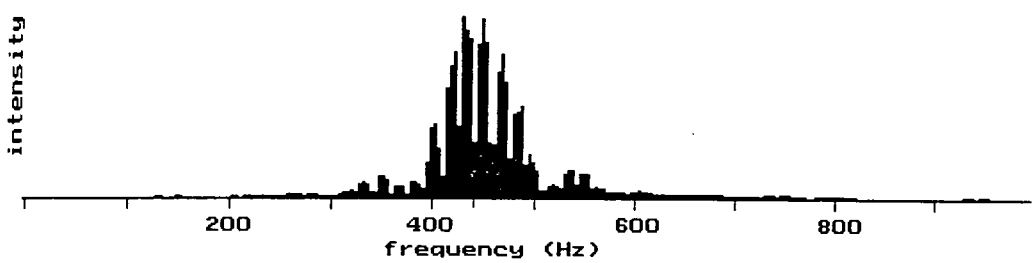


Figure 7.4.1f.: Frequency Response of Grain with Space (2/1).

The FFT results above show some peaks around the source frequency [440 Hz] with an interval of "group frequency"; frequency of grain + interval. The strongest peak, however, is not at the source frequency.

A similar set of analyses was done with a 2,560-sample-long grain model; four times longer than the standard. Their FFT results show the same phenomenon; peaks are distributed around the source frequency, 440 Hz, with the interval of group frequency. The central frequency of the group of peaks seems to be at the source frequency.

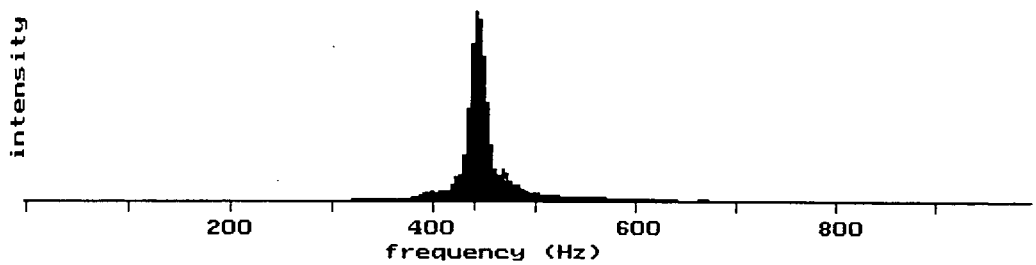


Figure 7.4.2.: Frequency Response of Grain with Space (2560:2/1).

These results suggest that the space between the grains contributes to amplitude modulation at a "group frequency" around the source frequency. In terms of frequency, the strongest peaks on the frequency analyses are not always at the source frequency. The tendency of the peaks, however, shows that the central frequency is at the source frequency.

7.5. Summary

The artefacts of granulation are controlled as follows. The side-bands appear both sides of each strong peak in the source sound. When an extra harmonic component appears in the negative frequency, this will be reflected into the positive side. The intervals of the extra harmonics depend upon the size of the grain and the delay ["group" length]; amplitude modulation by the source frequency with the group frequency. It means that a high grain speed [i.e. low "grain frequency"] makes a wider interval. The number of side-bands is determined by the body-ramp ratio, however, that does not depend on the size of the body. The side-bands are, therefore, firstly controlled by the body-ramp ratio, as a profile of the side-bands, and then are multiplied by the group frequency. In other words, the side-band-width can be changed by altering either the grain-body size or the ramp size. This leads to a real-time implementation of a fixed ramp-size grain model. The details of the implementation are discussed in the next chapter.

The granulation model used in this chapter [160-320-160] could be seen as a special case of FOF cell. In the case of a standard FOF, the shape of the cell is asymmetric without a sustained part, and the size of the cell is not longer than 10 msec whereas the grain model is symmetrical and far longer [20 msec]. The frequency response of a standard FOF cell, however, is similar to that of the granulation model; one strong peak with a few side-bands. In the case of the FOF, the top- and the skirt-pass band width are determined by the length of the attack- and the decay parts whereas the side-band-width of the granulation model is basically dictated by the body-ramp ratio and multiplied by the group frequency. Also, the

behaviour of the granulation model could be described as that of a wavelet with a mother function [or envelope; rise-sustain-decay].

Chapter 8. Implementation of Real-Time Granular Synthesis and Sound Granulation on the Network

8.1. Preliminary Implementation -using only on-chip memory-

At a preliminary stage of the implementation, a short fixed sound sample was used, as an experiment for controllability. A nine-voice system was implemented over 16 transputers, a branch of the 160 Transputer Network, using a 160-sample-long grain model; 40-sample-long rising ramp, 80-sample-long sustained part, 40-sample-long decaying ramp. A 320-sample-long sine wave-table in 440 Hz is the source of the granulation.

The grain parameters; grain-body size, delay, source sound frequency and amplitude are controlled from a PC keyboard that is connected with the root transputer. A set of keys on the PC keyboard are assigned for controlling the parameters; pressing a key causes a parameter to increase or decrease. Since a short-grain-model is employed, the frequency of the granulated sound is determined by the interval of the grain, "delay": the group [grain + delay] frequency is the output frequency.

The source sound frequency is controlled using a wave-table synthesis technique; note **A** in each octave synthesised from the original source 440 Hz. When the required output frequency is not equal to the source frequency, the output should be synthesised by means of changing the length of "delay". When "delay" is changed gradually, a *glissando* effect [gradual change in frequency] is available with this configuration. The

minimum increment of the frequency depends on the size of grain and the source sound frequency.

Since a short grain model, 160-sample-long, is employed, a full cycle for low frequency tones of less than 200 Hz cannot be fitted into a grain. For example, a 55 Hz tone requires at least 582 samples to complete a full cycle, at a 32 kHz sampling rate. This causes a difference in the output sound texture between sounds above 200 Hz, where at least one whole cycle occurs in a grain, and those below, where less than one cycle occurs in a grain. As the grain size becomes larger, this threshold is lowered. If the threshold is 20 Hz, however, the lower limit of the human auditory system, a 1,600-sample-long model is needed, and this may not create enough artefacts of the granulation process unless a steep ramp is applied.

A change of parameters is allowed only at the end of a grain, due to the restrictions on the software: some ALT structures have to be introduced to enable interactions within a grain period that necessitates a long delay. If a change of parameters is made during a grain period, the new information is held firstly by the key buffer of the PC, where no software control is available from TDS, and then at a transputer assigned for the granulation task. The latency of the system against a parameter change is less than a grain period (4 msec). If the configuration is expanded over the 160 Transputer Network, 81 voices become available.

As a first step towards improving control flexibility, two coefficients; amplitude and frequency, can be communicated to the network through

the MIDI-to-Transputer interface. The MIDI inputs, "velocity" and "key number", are converted to the amplitude and the frequency of the output sound. The latter is done using the method described above. Possibilities exist for controlling other key parameters through MIDI using continuous controllers (de Tintis 1995), program change or system exclusive messages, but these have yet to be fully investigated.

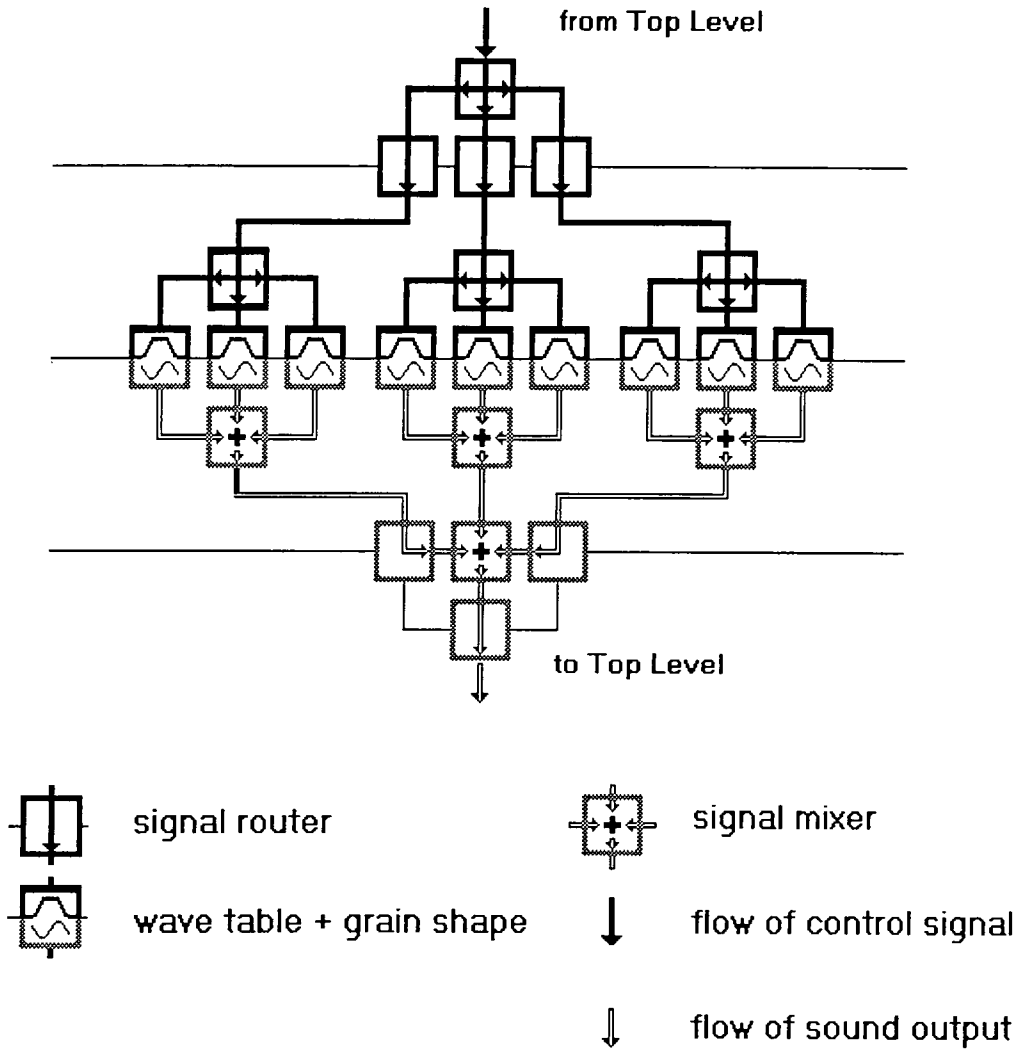


Figure 8.1.1.: Granular Synthesis with Short Wavetable.

Since a networked transputer has a 4k-byte on-chip memory, there are some limitations on programming. For example, to synthesise the lowest **A** on an acoustic piano [fundamental frequency: 27.5 Hz] using a wave-table, at least an 1,164-sample-long table is required for a complete cycle, at a 32 kHz sampling rate. If a 1,200-sample-long wave-table is in 32-bit integer format, it requires 4.8k bytes of memory space, more than the on-chip memory of a transputer. If it is in 16-bit integer format [that of the DAC used in the system] and occupies 2.4k bytes, this unconverted format, nevertheless, requires an additional overhead in calculation, due to the lack of a 16-bit processor on the 32-bit based T800 transputers. This led to the idea of a distributed wave-table. The wave-table for granulation is distributed over three transputers and is controlled by another transputer.

This distributed wave-table provides an 800 sample-long (or 25 msec at a 32 kHz sampling rate) sample storage facility. In a "grain generator", a static 160-sample-long simple ramp is implemented. The ramp could be replaced with that of half-cosine or Gaussian characteristics with a simple change in the initialisation program. The parameters that describe a grain; grain-body size, grain-body range, amplitude and source frequency, are controlled in real-time from a PC keyboard through a host transputer. The size of the ramp can be re-programmed, but altering the grain-body size makes the body-ramp ratio change, so providing a variety of side-band widths [see Chapter 7].

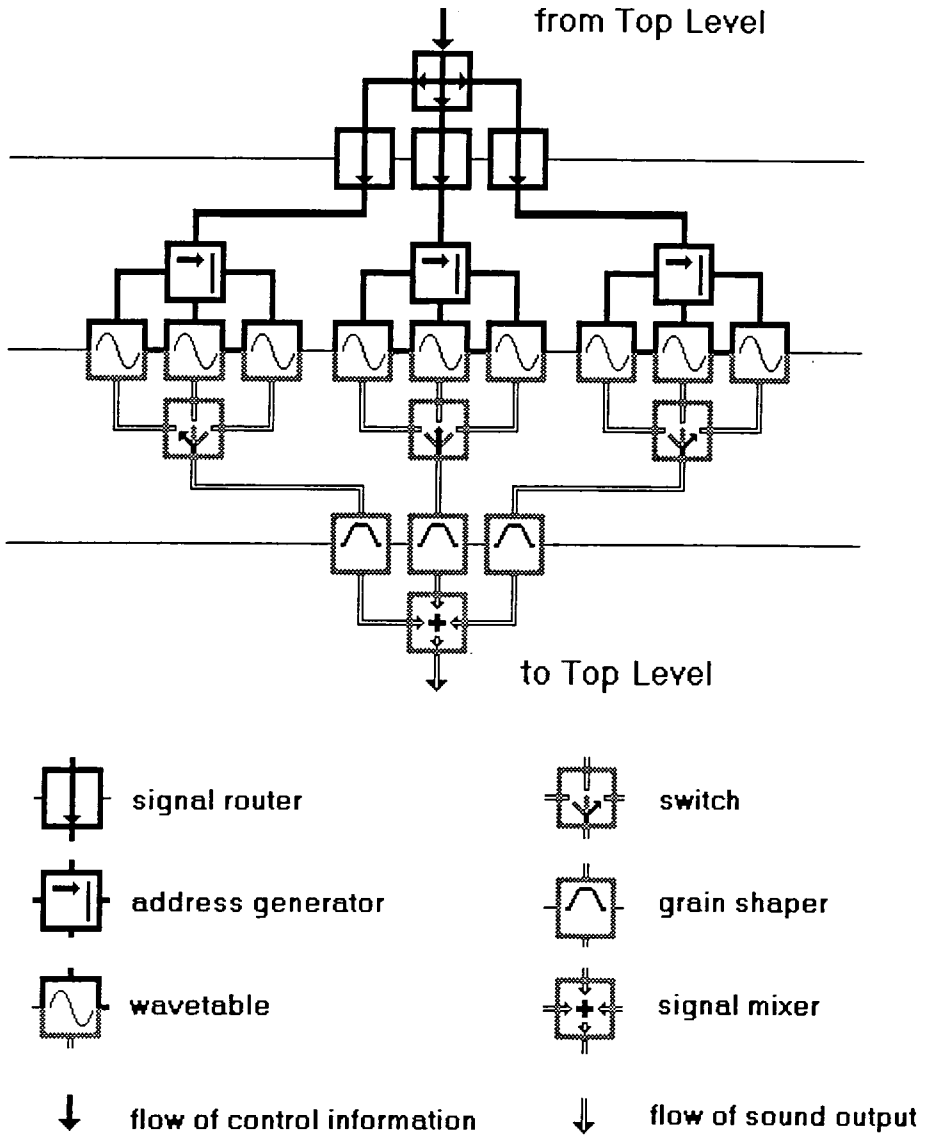


Figure 8.1.2.: Granular Synthesis with Distributed Wavetable.

The system reliably performs real-time granulation. Due to the lack of a long sample storage facility, however, the experiment was limited to granular synthesis using a short sine wave and a similarly brief extent of "synthetic" sounds. From this experiment, I ascertained that it is possible

to implement nine voices of real-time granular synthesis over the 160 Transputer Network with some external sample storage.

8.2. Revised Implementation -using 256k external memory-

In the configuration described above, due to the lack of on-board local memory, some of the transputers had to be assigned solely for the purposes of sound sample storage. This has not proved particularly efficient, since only 800 samples, in 32-bit floating point format, can be accommodated within a transputer's on-chip memory; 4k-byte. This is also against the original rationale of the network architecture, whereby a large memory storage should not be required for real-time systems. It became clear, however, that the minimum memory requirements for a sampled sound granulation are far greater than that available on the core network. The architecture of the main network, nevertheless, makes provisions for the attachment of extra processors and memory on peripheral links; a possibility wisely anticipated at the design stage.

For the revised configuration for asynchronous granular synthesis, a series of transputer cards consisting of a T800 transputer [20 MHz clocked] with 256k bytes of external memory, are connected to the network at the bottom of the tree structure through extension cables. The manufacturer of the transputer recommends that the maximum length of such cable should be less than 0.5 metre and it should be twisted (INMOS 1987). Because of the hardware architecture of the 160 Transputer Network, however, the extension cable had to be longer than the limit. This unfortunately resulted in occasional failures of signal transmission. The problem was eventually solved by shielding the cables with a metallic cover and a grounded copper board.

Currently there are nine such external boards that allow granular synthesis with nine voices. Each voice is distributed between the left and right channels in a fixed ratio, creating a spread of sound images for the final two-channel stereo output. In this configuration, the grain parameters; offset, grain speed, grain speed range [randomise factor], grain size, grain range [randomise factor] and time stretch ratio are controlled from a PC keyboard: a pair of keys are assigned to each parameter. Whenever a key interaction is made, a set of fresh parameters are sent to the tree top of the network .

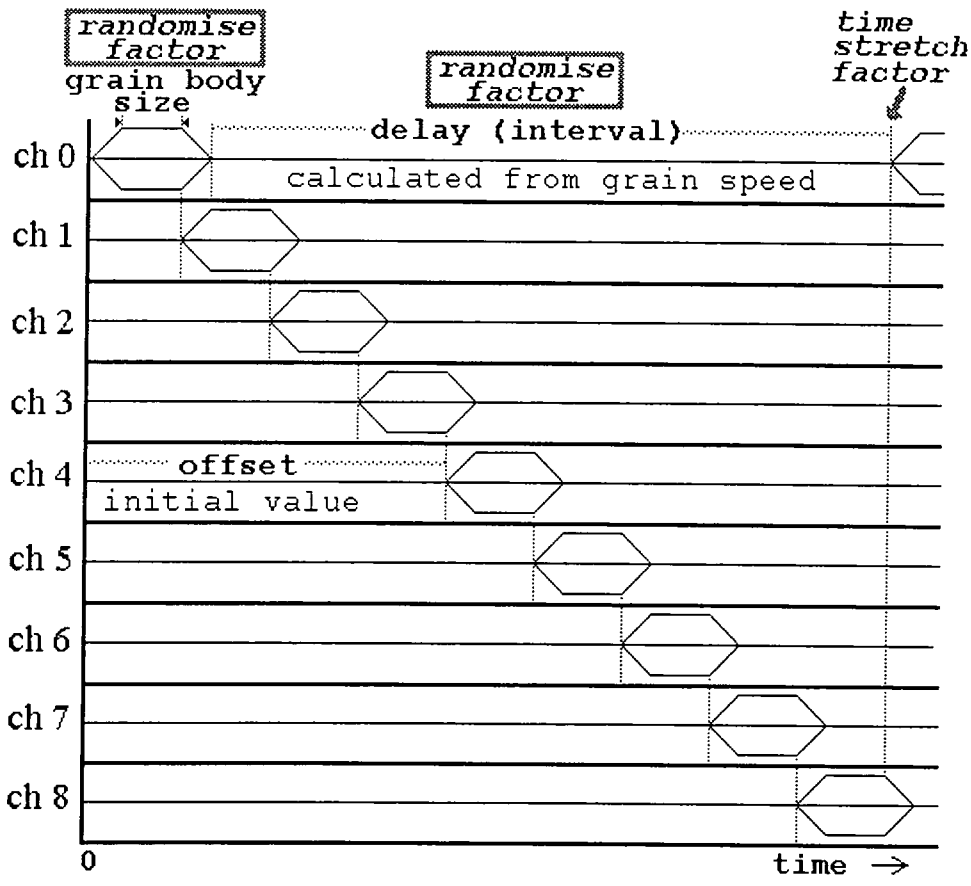


Figure 8.2.1.: Timing Chart of the Sound Granulation.

This top-down approach allows each voice to granulate the sound individually. In addition, each voice has its own random number generator with a different seed. These measures assure genuine "asynchronous" granulation that is not compromised by the subsequent processes of accumulating sound data, which uses the same bottom-up method as described in the case of the additive synthesis model. This approach ensures a much richer depth of granulation in subjective terms than other methods that frequently economise on computation by using a single random number generator for all voices.

The current program is based on a 20 msec [or 640 samples in 32 kHz sampling rate] grain that consists of a 160 sample-long rising ramp, a 320 sample-long body and a 160 sample-long decaying ramp. As in the core of the original implementation, due to the on-chip memory size limitation [4k bytes], the length of the ramp is fixed. The length of the grain body is variable from 0 to 3,200 sample-long, and this facilitates a variety of body-ramp ratios that controls the frequency profile of the side-bands. In the case of Truax's system (Truax 1988), the size of the grain ramp is also variable. In this configuration, the size of the grain ramp is fixed, avoiding the re-calculation of the ramp, but this limitation is compensated for by making the grain-body size variable, since the ramp-body ratio dictates the frequency bandwidth of the granulated sound, regardless of the size of the ramp, as shown in Chapter 7.

At a preliminary stage, I employed shorter grain models; 20, 40, 80, 160 and 320 samples long. The grain speed can be changed from 0.1 to 50 gps per voice. The maximum available grain density is determined by the

grain size; for instance, the maximum grain speed for the 160 sample-long model is 100, providing that the time interval between the grains is kept at zero.

Using Gabor's "sliding window" technique (Gabor 1946), the sound granulation program becomes capable of time stretch/compress applications, by means of a technique derived from wave-table synthesis; address generation. In traditional sound granulation [without time stretch/compress factor], a duration of sound [equivalent length to a grain] is read from the sound storage, then the next reading point is moved the length of a grain. In the case of the sound stretch/compress, the next reading point of the source sound is not always moved the length of a grain. In this implementation, the movement of the window could be varied from zero [that is repeating a window, thus producing sound freezing; infinite stretching] to double the grain size [2x time compression].

The specification of the program is shown in the Table below.

Number of Voices	9
Grain Size	320-3200 sample-long
Ramp Size	160 sample-long (fixed)
Grain Density	0.01-50 gps per voice
Time-stretch Ratio	0.5 - infinity (freeze)
Memory Size (each voice)	256k bytes (128k samples)

Table 8.2.1.: Specification of the Real-Time Sound Granulation Implementation.

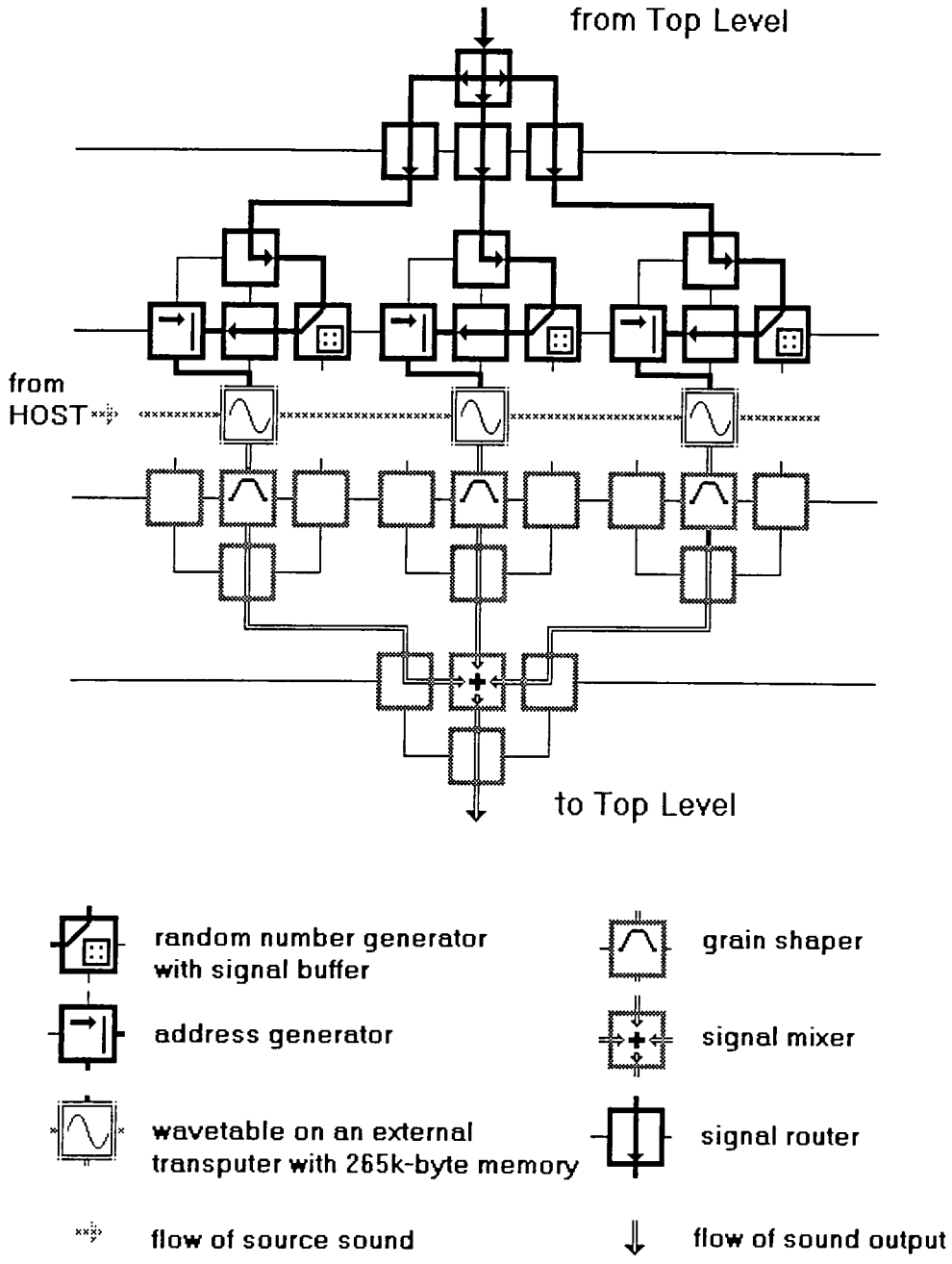


Figure 8.2.2.: Sound Granulation with Long Sound Storage (part).

For a granulation of speech, it is often necessary to use grain-models longer than 320 samples [or 10 msec] to avoid the artefacts of the ramp. In the case of natural musical sounds, however, subjective tests have shown that a 320 sample-long model may often prove acceptable at the upper boundary and in many situations much shorter grains can produce interesting transformations, *see* Chapter 7 for details.

Since a change of the parameters can be made only at the end of the grain, the reaction of the system would be slow at a low grain speed, and the change may cause a deadlock, because of the difference between the size of data in the PC's keyboard buffer and the speed of data processed that is beyond the control of the TDS; the operation system for the host transputer and a host PC. At a fast speed, such as 20 gps or more [an interval of 50 msec or less], the change of the parameter is almost in real-time.

For additive synthesis programs, the network can handle an audio data flow at a 32 kHz sampling rate with a control data flow up to 500 bytes-per-second. In the case of the granular synthesis programs, a deadlock of the type described above has not occurred, since the control data traffic is not so high as in the case of the additive synthesis and a constant stream of sound output is maintained.

8.3. Conclusion

A real-time granular synthesis and a real-time sound granulation were implemented on to a part of the 160 Transputer Network. In the granular synthesis configuration, some transputers had to be assigned solely for the purposes of memory storage, and this arrangement resulted in the use of fixed short sound samples. This was actually against the original concepts of the network and thus not optimal. By using external transputers with extra memory, the sound granulation application allows more flexible applications, such as time-shifting and time-stretching of much longer sampled sounds.

The system for sound granulation requires a large amount of sample storage. This, however, does not necessarily have to be an independent memory block for each voice, unless the system is used for sound morphing. Therefore, the ideal shape of a network for sound granulation would be a tree structured network such that the bottom layer is connected to a large block of memory through a high-speed bus. For real-time sound processing, the bus must additionally be connected to an ADC or a digital sound interface.

Chapter 9. Discussion

9.1. Efficiency of the Applications

In the optimised configuration for the nine-voice polyphony with a dynamic note allocation at a single-sampling-rate, 72 oscillators are allocated to each note: there is a significant element of inefficiency in their deployment. As noted earlier, due to the relationship between the Nyquist frequency and the harmonic contents, some of the higher order harmonics have to be muted.

For example, in the case of "concert" A [440 Hz], its 37th harmonic [16,280 Hz] is above the Nyquist rate [16,000 Hz] of the 32 kHz sample frequency. This means that half of the allocated oscillators are kept muted. Real efficiency of the configurations should be measured by an average of "active" harmonics per key, by applying all the available oscillators per voice to every 88 keys in an acoustic piano range, instead of the number of oscillators allocated to a voice.

Program Type	Sampling Rate(s) [kHz]	Voices	Allocation of Oscillators per Voice	Available Oscillators in Total	Average Active Harmonics per Key
Fixed Allocation [1]	32	81	8	648	6.136
Fixed Allocation [2]	32	88	16, 12, 8, 4	752	8.341
Dynamic Allocation [1]	32	27	24	648	16.06
Dynamic Allocation [2]	*32*	*13*	*48*	*624*	*26.08*
Dynamic Allocation [3]	32	9	72	648	33.66
Multi-Rate Fixed	8, 16, 32	88	24, 16	1296	13.61
Multi-Rate Dynamic [1]	10.6, 32	22	36	816	21.46
Multi-Rate Dynamic [2]	4, 8, 32	15	80	1200	32.71

N.B. The program in *italic* has not been implemented.

Table 9.1.1.: Efficiency of the Applications.

Another matter concern is the usage of processors. In the case of the program for multi-rate 13-dynamic-voice synthesis, several processors are unused, due to the underlying network architecture: 3*n* grouping [where *n* = 1, 2, 3...]. Since a T800 transputer has four communication links, the network that is formed is the "1-3" revised ternary-tree structure [see Figure 2.4.1.]. This leads to 3*n* division-based configurations, due to the software's controlling and accumulation purposes. If the processor had been designed with five links rather than four, the network could have been constructed using a "1-4" revised quadruped-tree structure, resulting in a 2*n* [binary-tree] or 4*n* division-based configuration which would be altogether more flexible than the 3*n* approach adopted on the 160 Transputer Network.

In the multi-rate applications, a part of the network has to be assigned as two filter banks to boost the production of sounds at the lower sampling rates to match with that of the desired output. The number of processors assigned for the filter banks has to be 4*n*: four is the number of transputers on a leaf [the smallest dividable section] of the network. The constraints of real-time synthesis restrict the permitted length of the interpolation filters. Also, a transputer has finite capabilities, thus a number of them are required for a single filter. This leads to some degree of compromise, involving a FIR filter of reasonable length with an optimal distribution of it on a part of the network.

9.2. Provisions for New Processors

9.2.1. T9000

Details of the successor to the T800, the T9000, were announced in 1991 (INMOS 1991), but production difficulties significantly delayed the shipment of commercial quantities. The figures on the data-sheet show that the T9000 [50 MHz clocked] has about ten times the performance of the T800 [20 MHz] and 16k bytes of on-chip memory [4k bytes on the T800], representing a significant improvement on the existing generation of transputer products in both capacity and performance (May, et al. 1992).

Our research group obtained a few of the processors in a prototype form, de-rated to 20 MHz, and tested them in a PC-based environment. It was, however, difficult to make a straight forward comparison between the T800 and the T9000, since the T9000 was still not fully functional at the time of writing and the language used for these tests was parallel C, which creates larger and potentially less efficient codes than Occam used on T800s.

There are also some significant differences between the transputers; the T9000 has a 64-bit FPU [32-bit FPU on T800], a calculation power of 200 MIPS [whereas the T800 performs 20 MIPS peak at 20 MHz clock] and the four links of the T9000 provide a maximum bi-directional data bandwidth of 70 Mbytes per second, whereas the four links of the T800 are restricted to 11.2 Mbytes per second. When these performance data were fully implemented, the T9000 would be able to execute a sound

synthesis task, such as a bank of recursive sine oscillators, at a significantly higher sampling rate and in higher precision than its predecessors.

Also, the 16k bytes on-chip memory is sufficient for use as local data storage for multi-segmented amplitude envelopes. For the granulation of sampled sound, a number voices may be implemented on a single T9000 providing each can share the contents of an external memory.

In late 1996, SGS-Thomson [the parent company of INMOS] announced that the production of T9000 would cease in 1997, thence, the company would focus on to its own ST020 product family rather than the Transputer family acquired with the company; INMOS. At this time, the fastest T9000 model available is still 25 MHz clocked, half of the figure announced in 1991, resulting in less than 100 MIPS. The T9000 equivalent in the ST020 family is ST020-450; a 32-bit processor [32 MIPS at 40 MHz] with four communication links at 20 Mbit/s which is supported by ANSI C compiler (Beckett 1996).

9.2.2. DSP 56300

Motorola's latest DSP product, the DSP 56300 family, was announced in 1995. The first family member of the 24-bit DSPs, DSP 56301, provides 80 MIPS at 80 MHz clock rate (Motorola 1995). Its performance is eight times faster than its predecessor, DSP 56000, and is thus comparable to a T9000 [200 MIPS at 50 MHz]. A 100 MIPS at 100

MHz clock version is expected in 1997 (Motorola 1996). The DSP chip has two ESSI and one SCI interfaces for communication with external processes, useful for communication with non-family devices.

In case of INMOS's Transputer family, however, the high speed links are primarily designed for communicating with other transputers, hence connections to devices other than members of the Transputer family require special signal converters, such as C011. This means that the true potential of transputers is realised when a number of them are interconnected to create a mesh array or a tree structure, thus creating a self-contained massively parallel distributed architecture.

9.2.3. ADSP-2106x SHARC

Another example of the modern processors is Analog Devices' SHARC [Super-Harvard ARchitecture Computer] family. ADSP-21060 is a 32-bit processor that is capable of 40 MIPS and 120 MFLOPS [peak] at 40 MHz clock with ten DMA channels supported by a DMA controller that allows ten simultaneous channels communication without processor overhead, separate on-chip buses, and six point-to-point links and two serial ports for connection with other processors (Analog Devices 1996). Its dual ported 4-megabit SRAM is the largest in size on-chip memory of any processor (BittWare 1997). The processor is as powerful as the DSP 56300 family. There is, however, a significant feature to note: the high-speed DMA channels [240 Mbytes/s] also allow memory-access intensive operations, such as FFT.

9.3. Shape of Network for Sound Synthesis

In arguing the case for parallel architectures based on the transputers, it has to be acknowledged that a number of high-speed DSP chips in a unitary fabrication are now available. For many tasks, however, a single high-speed DSP chip is not necessarily superior in terms of performance to a parallel and distributed system based on a number of less powerful and financially attractive processors, especially if the latter have high-speed communication links. The network structure, basically, should be a tree or hyper-cube, since such arrangements are good for distributing control information and accumulating sound output. A tree or a cube, however, is not necessarily homogeneous in calculation power and memory size.

When a DSP chip has four communication links and is configured as a three-to-one signal mixer, a minimum requirement for real-time operation is the capacity to fetch a data packet from each channel, to sum them up and to output the new data packet within a sampling period that requires less than 10 MIPS at standard audio sampling rates; 32 kHz, 44.1 kHz or 48 kHz. This gives weight to the argument that tasks demanding environments such a multi-processor network, as an audio processor, could be more efficiently realised by a hybrid network consisting of high-speed processors with signal routers, rather than a homogeneous configuration of high-speed processors.

The idea of an "hybrid" network also can be applied to memory allocation, such that the tasks associated with sound synthesis, such as amplitude

envelopes, can have larger local cache memories than those associated with the processes of data accumulation and distribution, such as routers and mixers.

For example, because of the design restrictions on the networked transputers, just 4k-byte of on-chip memory each, and the general architecture of the network, only simple amplitude envelopes can be applied to the additive synthesis methods described above. If the processors at the bottom of the tree were to be replaced with ones offering more calculation power and memory, this combination may be able to produce more sophisticated amplitude envelopes. In other words, length and strength of tree branches and nodes are not necessarily homogeneous.

When the 160 Transputer Network was designed, the memory access speed was not so fast, hence the larger on-board memory meant a heavier load to the CPU. Naturally, the original rationale behind the design of the network was that a real-time distributed system should not require a large amount of on-board memory for intermediate data storage. These days, most of the modern DSP chips have a fast memory access facility, such as DMA channels. Besides, thanks to technical advances in the fabrication of semiconductors, the price of high-performance memory became lower and then the size became larger. Therefore, large on-chip or on-board memory may no longer become an obstacle to building a distributed system.

Chapter 10. Conclusion

The potential application of the 160 Transputer Network as a dedicated real-time audio processor was confirmed, in particular, as an additive synthesis engine. The network was also tested as a resource for other methodologies such as granular synthesis and sound granulation. The processing power of the network can only be fully extracted with an optimal software configuration, especially in terms of load balancing and smooth communication protocols.

The original rationale behind the network architecture was that a real-time distributed system should not require a large amount of on-board memory for intermediate data storage. At the time of building the network, this was acceptable, since the access speed to ancillary memory space would not have satisfied the conditions for real-time operation, certainly at a modern audio quality like CD [44.1 kHz sampling rate] or DAT [48 kHz sampling rate]. This unique feature of the design, the absence of any external memory local to the networked transputers, brought some restrictions for programming; such as no memory intensive applications like data storage for wavetables and amplitude envelopes. It now has to be acknowledged that due to rapid innovations in both processors and memory-chips, a large quantity of high-speed memory on a processor and that local to a node have become the norm and are not a significant cause of delay for system design.

The capability of the fifteen-year-old MIDI standard, as a keyboard driven controller for real-time sound synthesis, was examined by practical testing. It was confirmed that as an event oriented standard, there is still some

spare capacity to transmit a multiple-keyboard performance. This suggests that it may not be necessary to have an alternative to the MIDI standard in near future, as some ad-hoc improvisations, such as a multiple MIDI cable application supported by a high-power PC, are currently satisfactory to most musicians.

The network architecture, the modified ternary tree, provides short path lengths between the arbitrary nodes, also between the siblings at the same level, and the same path length between the tree-top and each leaf of the tree leads to no phase delay on the additive synthesis programs. The extra path at the same level serves to route the information avoiding the hub, "hot spots", that otherwise interrupt the functions of the network as a whole. However, this research also suggests that the network structure for sound synthesis is not necessarily homogeneous, in terms of calculation power and memory size, since a large on-chip or on-board memory is not an obstacle for real-time processing, thanks to high-speed memory access technology.

The optimised methods for additive synthesis, by resource management and by multi-rate approach, are effective by a significant order of magnitude in enhancing the quality of tumbrel detail. The real-time implementations of the methods on to the network, however, brought limited success, due to lack of memory for amplitude envelopes; each networked transputer has a 4k-byte on-chip memory without any external memory. The multi-sample-rate optimisation has also been shown to be effective for accelerating the processes of dynamic tone generation. For the real-time implementations of the multi-rate technique on to the

network, however, the up-sampling filters may introduce some phase delay and other distortions, due to hardware limitations in memory size for filter coefficients and in calculation power.

In the case of the implementation of granular synthesis and sound granulation in real-time, despite the conceptual simplicity, the signal processing requirements were tough. Because of the nature of the synthesis methods, composers' demands for real-time implementation is enormously high. Only a few fully developed real-time models are now available world-wide, including this application on the 160 Transputer Network. The implementation with adequate control proved the viability of parallel and distributed computing for the applications, especially in the independence of each voice implemented in parallel that provides extra depth of the sound. The sound granulation on the system is, however, restricted to short sampled sounds or simple synthetic sounds, due to lack of sample storage, in terms of memories and a high speed hard disc.

The T800 transputers used for these configurations are no longer ranked as top-of-the-range in terms of performance. Besides, the Transputer family itself has disappeared from the production lines, after the manufacturer lost its commercial independence. The long term significance of these investigations, however, lies in the knowledge acquired in implementing concepts of parallel audio processing in a practical context. The algorithms with which the architectures were developed can also be transplanted onto the various ranges of higher performance processors now available for parallel and distributed computing.

References

Analog Devices, Inc. 1996. *ADSP-21062/ADSP-21060.* Norwood, Massachusetts: Analog Devices, Inc.

Andrenacci, P., E. Favreau, N. Larosaa, A. Prestigiacomo, C. Rosati, and S. Sapir. 1992. "MARS: RT20M/EDIT20 Development tools and graphical user interface for a sound generation board." In *Proceedings of the 1992 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 340-343.

Aspnäs, M., R. J. R. Back, and T-E. Malén. 1990. "Hathi-2 Multiprocessor System." *Microprocessors and Microsystems.* 14(7): 457-466.

Asta, V., A. Chauveau, G. di Giuhno, and J. Kott. 1980. "The 4X: a real-time digital synthesis system." *Automazione e Strumentazione.* 28(2): 119-133.

Audsley, G. A. 1905. *The Art of Organ-Building.* London: Marston & Co. Ltd.

Bailey, N. J., I. Bowler, A. Purvis, and P. D. Manning. 1990. "An Highly Parallel Architecture for Real-time Music Synthesis and Digital Signal Processing Application." In *Proceedings of the 1990 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 169-171.

Bailey, N. J. 1992. *On the Synthesis & Processing of High Quality Audio Signals by Parallel Computers.* PhD Thesis, University of Durham.

Bartoo, T., D. Murphy, R. Ovans, and B. Truax. 1994. "Granulation and Time-Shifting of Sampled Sound in Real-Time with a Quad DSP Audio Computer System." In *Proceedings of the 1994 International Computer Music Conference (Århus, DENMARK).* San Francisco, California: International Computer Music Association. pp. 335-337.

Bastiaans, M. J. 1980. "Gabor's Expansion of a Signal into Gaussian Elementary Signals." In *Proceedings of the IEEE.* 68 (4): 538-539.

Bastiaans, M. J. 1985. "On the Sliding-Window Representation in Digital Signal Processing." *IEEE Transactions on Acoustic, Speech, and Signal Processing.* ASSP-33(4): 868-873.

Beckett, D. 1996. "SGS-Thomson ST020-450 Transputer Datasheet." On URL=http://www.hensa.ac.uk/parallel/transputer/documentation/st020-450/datasheets/index.html.

BittWare. 1997. "SHARC ADSP-21060/62." On URL=http://www.bittware.com/products/tech/sharc.htm.

Boittiaux, B., G. Goncalves, and M. P. Haye. 1992. "A Transputer Network for a Device Simulator." *Microprocessing and Microprogramming.* 35: 127-132.

Bowler, I., P. D. Manning, A. Purvis, and N. J. Bailey. 1989. "Additive Sound Synthesis on a Multi-transputer Network." In *Proceedings of the 1989 International Computer Music Conference.* San Francisco, California: International Computer Music Association.

Carson, J. 1922. "Notes on the theory of modulation." In *Proceesings of the Institute of Radio Engineers* 10: 57-64.

Chowning, J. 1973. "The synthesis of complex audio spectra by means of frequency modulation." *Journal of the Audio Engineering Society.* 21(7): 526-534.

Clarke, J. 1990. "An FOF synthesis tutorial." In B. Vercoe (ed.) *Csound: A Manual for the Audio Processing System.* Cambridge, Massachusetts: MIT Media Laboratory.

Cordy, W. J. and W. Waite. 1980. *Software Manual for the Elementary Functions.* Englewood Cliff, N.J.: Prentice-Hall.

Dolson, M. 1986. "The Phase Vocoder: A Tutorial." *Computer Music Journal.* 10(4): 14-27.

Dutilleux, H., A. Grossmann, and R. Kronland-Martinet. 1988. "Applications of the wavelet transform to the analysis, transformation, and synthesis of musical sound." Presented at the 85th Convention of the Audio Engineering Society. New York: Audio Engineering Society.

Eckel, G., M. R. Iturbide, and B. Becker. 1995. "The development of GiST, a Granular Synthesis Toolkit Based on an Extension of the FOF Generator." In *Proceedings of the 1995 International Computer Music Conference.* San Francisco, California: International Computer Music Assocciation. pp. 296-302.

Fletcher, H., E. Blackham, and R. Stratton. 1962. "Quality of piano tones." *Journal of the Acoustical Society of America.* 34(6): 749-761.

Gabor, D. 1946. "Theory of Communication." *Journal of Institution of Electrical Engineers.* 17(3): 429-457.

Gabor, D. 1947. "Acoustical Quanta and the Theory of Hearing." *Nature* 159(4044): 591-594.

Gordon, J. W., and J. O. Smith. 1985. "A Sine Generation Algorithm for VLSI Applications." In *Proceedings of the 1985 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 165-168

Green, D. M. 1971. "Temporal Auditory Acuity." *Psychological Review.* 78(6): 540-551.

Hart, J. F., E. W. Cheney, C. L. Lawson, H. J. Maehly, C. K. Mesztenyi, J. R. Rice, H. C. Thacher, and C. Witzgall. 1968. *Computer Approximations.* Wiley: New York.

von Helmholtz, H. L. F. 1863. *On the Sensations of Tone as a Physiological Basis for the Theory of Music.* (Translation of the 1877 Edition) Dover, 1954.

INMOS Limited. 1987. *IMS T800 Transputer: product data.* Bristol: INMOS Limited.

INMOS Limited. 1989. *The Transputer Databook.* Trowbridge UK: Redwood Burn Ltd.

INMOS Limited. 1991. *The T9000 Transputer Products Overview Manual.* INMOS Limited.

International MIDI Association. 1983. *MIDI Musical Instrument Digital Interface Specification 1.0.* North Hollywood, California: International MIDI Association.

Itagaki, T., A. Purvis, and P. D. Manning. 1994. "Real-time Synthesis on a Multi-processor Network." In *Proceedings of the 1994 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 382-385.

Itagaki, T., D. K. Phillips, P. D. Manning, and A. Purvis. 1995. "An Implementation of Optimised Methods for Real-time Sound Synthesis on a Multi-processor Network." In *Book of Abstracts of Parallel Computing 95.* Gent, Belgium. p. 100.

Itagaki, T., P. D. Manning, and A. Purvis. 1996. "Real-time Granular Synthesis on a Distributed Multi-processor Platform." In *Proceedings of the 1996 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 287-288.

Itagaki, T., S. Johnson, P. D. Manning, D. J. E. Nunn, D. K. Phillips, A. Purvis, and J. R. Spanier. 1996. "Durham Music Technology: Activity Report." In *Proceedings of the 1996 International Computer Music Conference (Hong Kong).* San Francisco, California: International Computer Music Association. pp 126-128.

Jansen, C. 1991. "Sine Circuitu." In *Proceedings of the 1991 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 222-225.

Kleczkowski, P. 1989. "Group Additive Synthesis." *Computer Music Journal.* 13(1):12-20.

Lee, A. S. C. 1995. "CSOUND Granular Synthesis Unit Generator." In *Proceedings of the 1995 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 230-231.

Lindemann, E., M. Starkuer, and F. Dechelle. 1990. "The IRCAM Musical Workstation: Hardware Overview and Signal Processing Feature." In *Proceedings of the 1990 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 132-135.

Lindemann, E., M. Puckette, E. Viara, and M. Starkier. 1990. "The IRCAM Signal Processing Workstation - An Environment for Research in Real-Time Musical Signal Processing Performance." *Microprocessing and Microprogramming.* 30: 167-174.

Lippe, C. 1993. "A Musical Application of Real-time Granular Sampling Using the IRCAM Signal Processing Workstation." In *Proceedings of the 1993 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 190-193.

MacKenzie, C. L. 1985. "Structural Constraints on Timing in Human Finger Movements." In Goodman, D., R. B. Wilberg, and I. M. Franks (eds.) *Advances in Psychology: Differing Perspectives in Motor Learning, Memory and Control.* North-Holland: Elsevier Science Publications B.V.

Maehle, E., and W. Obelöer. 1992. "DELTA-T: A User-Transparent Software-Monitoring Tool for Multi-Transputer Systems." *Microprocessing and Microprogramming.* 35: 245-252.

Mathews, M. V. 1969. *The Technology of Computer Music.* Cambridge, Massachusetts: MIT Press.

May, D., R. Shepherd, and P. Thompson. 1992. "The T9000 Transputer." On URL=http://www.hensa.ac.uk/parallel/vendors/inmos/T9000/T9000.ps.Z.

McMillen, K. 1994. "ZIPI: Origins and Motives." *Computer Music Journal.* 18(4): 47-51

Moore, F. R. 1977. "Table Lookup Noise for Sinusoidal Digital Oscillators." *Computer Music Journal.* 1(2): 26-39.

Moore, F. R. 1988. "The Dysfunctions of MIDI." *Computer Music Journal.* 12(1): 19-28.

Moorer, J. A. 1977. "Signal processing aspects of computer music." *Proceedings of the IEEE.* 65(8): 1108-1137.

Motorola. 1995. "DSP 56301 Data Sheet." On URL=http://www.mot.com/SPS/DSP/home/eng/tec/56301ds.html.

Motorola. 1996. "DSP 56300 Family Overview." On URL=http://www.mot.com/SPS/DSP/home/prd/owr/DSP56300.html.

Norris, M. 1997. "SoundMaker Plug-Ins." *Computer Music Journal.* 21(1): 43-46.

Norskog, L. 1993. "SOX - SOund eXchange." On URL=http://www.spies.com/Sox/.

Nyquist, H. 1928. "Certain topics in telegraph transmission theory." *Transactions of the American Institute of Electrical Engineers.* April.

Phillips, D. K., A. Purvis, and S. Johnson. 1994. "A Multirate Optimisation for Real-Time Additive Synthesis." In *Proceedings of the 1994 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 364-367.

Phillips, D. K., A. Purvis, and S. Johnson. 1996. "Multirate Additive Synthesis." In *Proceedings of the 1996 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 496-499.

Phillips, D. K. 1997. *Algorithms and architectures for the Multirate Additive Synthesis of musical tones.* PhD Thesis, University of Durham.

de Poli, G., and A. Piccialli. 1988. "Forme d'onda per la sintesi granulare sincronica." In *the Proceesings of VII Colloquio de Informatica Musicale.* Rome: Associazione Musica Verticale. pp. 70-75.

Puckette, M. 1985. *A real-time music performance system.* Cambridge, Massachusetts: MIT Experimental Music Studio.

Puckette, M. 1991. "FTS: A Real-Time Monitor for Multiprocessor Music Synthesis." *Computer Music Journal.* 15(3): 58-67.

Purvis, A., R. Berry, and P. D. Manning. 1988. "A Multi-transputer Based Audio Computer with MIDI and Analogue Interface." Presented at Euromicro 1988, Zürich, Switzerland. Reprinted in *Microprocessing and Microcomputing 25* (1989): 271-276.

Rasch, R. A. 1978. "The Perception of Simultaneous Notes such as in Polyphonic Music." *Acustica.* 40: 21-33.

Ricci, A. 1997. "SoundMaker." On URL=ftp://ftp.alpcom.it/software/mac/Ricci.

Roads, C. 1978. "Automated Granular Synthesis of Sound." *Computer Music Journal* 2(2):61-62.

Roads, C. 1991. "Asynchronous Granular Synthesis." In de Poli, G., A. Picciali., and Roads, C. (eds.) *Representations of Musical Signals.* London: MIT Press, pp. 143-186.

Roads, C. 1996. *Computer Music Tutorial.* Cambridge, Massachusetts: MIT Press. ISBN: 0-262-68082-3.

Rodet, X. 1984. "Time domain format-wave-function synthesis." *Computer Music Journal.* 8(3):9-14.

Rodet, X., Y. Potard, and J. B. Barriere. 1984. "The CHANT Project: From the Synhesis of the Singing Voice to Synthesis in General." In Road. C. (ed.) 1989. *The Music Machine.* London, England: MIT Press. pp. 449-465.

Rodet, X., and Ph. Depalle. 1992. " A new additive synthesis method using inverse Fourier transform and spectral envelopes." In *Proceedings of the 1992 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 410-411.

Sandell, G. J. 1994. "SHARC Timbre Database." On URL=ftp://ftp.ep.susx.ac.uk/pub/sandell/README.html

Serra, X. 1994. "Sound hybridization based on a deterministic plus stochastic decomposition model." In *Proceedings of the 1994 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 348-351.

Spanier, J. R. (1998, in preparation). *Algorithms and VLSI Architectures for Parametric Additive Synthesis.* PhD Thesis, University of Durham.

de Tintis, R. 1995. "GRAINS: a software for real-time granular synthesis and sampling running on the IRIS-MARS workstation." In *Proceedings of XI CIM Colloquio di Informatica Musicale.* Bologna, Italy: Associazione di Informatica Musicale Italiana. pp. 221-223.

Todoroff, T. 1995. "Real-Time Granular Morphining and Spatialisation of Sound with Gestural Control within MAX/FTS." In *Proceedings of the 1995 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 315-318.

Truax, B. 1988. "Real-Time Granular Synthesis with a Digital Signal Processor." *Computer Music Journal.* 12(2): 14-26.

Truax, B. 1994. "Discovering Inner Complexity: Time Shifting and Transposition with a Real-time Granulation Technique." *Computer Music Journal* 18(2): 38-48.

de Vel, O. Y., and P. Thomas. 1990. "Multitransputer Architecture for the Low-level Characterization of Speech Spectrograms." *Microprocessors and Microsystems.* 14(5): 267-275.

Vercoe, B. 1986. *CSOUND Reference Manual.* London: MIT Press.

Volder, J.E. 1959. "The CORDIC trigonometric computing technique." *IRE Transactions of Electronic Computing.* 8(3): 330-334.

Wawrzynek, J., and T. von Eicken. 1990. "VLSI PARALLEL PROCESSING FOR MUSICAL SOUND SYNTHESIS." In *Proceedings of the 1990 International Computer Music Conference.* San Francisco, California: International Computer Music Association. pp. 136-139.

Xenakis, I. 1971. *Formalized Music.* Bloomington: Indiana University Press. ISBN: 0-25-4332378-9.

Bibliography

Adams, R., and T. Kown. 1994. "A Stereo Asynchronous Digital Sample-Rate Converter for Digital Audio." *IEEE Journal of Solid-State Circuits.* 29(4): 481-488.

Anderson, A. J. 1992. "Selection criteria in the development of a multiple processor based DSP system." *Journal of Microcomputer Applications* 15: 327-346.

Anderson, D. P., and R. Kuivila, R. 1986. "A Model of Real-Time Computation for Computer Music." In *Proceedings of the 1986 ICMC (The Hague).* San Francisco, CA: ICMA. pp. 35-41.

Anderson, K. H. 1977. "A Digital Sound Synthesizer Keyboard." *Computer Music Journal.* 2(3): 16-23.

Armani, F., L. Bizzarri, E. Favreau, and A. Paladin. 1992. "MARS - DSP environment and applications." In *Proceedings of the 1992 ICMC (San José, CA).* San Francisco, CA: ICMA. pp. 344-347.

Armani, F., A. Paladin, and C. Rosati. 1994. "MARS Applications Using APPLI20 Development Tools: a Case of Study." In *Proceedings of the 1994 ICMC (Århus).* San Francisco, CA: ICMA. pp.230-235.

Bailey, N., and A. Purvis. 1989. "An Implementation of CSOUND on the Transputer." In *Proceedings of the International Conference on the Applications of Transputers 1989.* Liverpool, UK.

Bailey, N., I. Bowler, A. Purvis, and P. Manning. 1990. "Concurrent CSOUND: Parallel Execution for High Speed Direct Synthesis." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 46-49.

Bailey, N., A. Purvis, P. D. Manning, and I. Bowler. 1991. "Some Observations on Hierarchical, Multiple-Instruction-Multiple-Data Computers." In *Proceedings of EuroMicro 1991.* Vienna, AUSTRIA.

Bartoo, T., and B. Truax. 1992. "Electroacoustic composer's workstation project." In *Proceedings of the 1992 ICMC (San José, California).* San Francisco, CA: ICMA. p.446.

Bate, J. A. 1990. "UNISON - A Real-Time Interactive System for Digital Sound Synthesis." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 172-174.

Bate. J. A. 1992. "MAX+Unison- Interactive control of a digital signal processor." In *Proceedings of the 1991 ICMC (San José, CA).* San Francisco, CA: ICMA. pp. 356-357.

Behrens, U., L. Hagge, and W. O. Vogel. 1994. "The ZEUS Eventbuilder: Experience with a Distributed Real-time Parallel Transputer System." *IEEE Transactions on Nuclear Science.* 41(1): 239-245.

Bekker, H., and M. Renardus. 1990. "Design of a Transputer Network for Searching Neighbours in M.D. Simulations." *Microprocessing and Microprogramming.* 30: 159-166.

Bischoff, J., R. Gold, and J. Horton. 1978. "Music for an Interactive Network of Microcomputers." *Computer Music Journal.* 2(3): 24-29.

Bosi, M. 1990. "The Sound Accelerator as a Real-Time DSP Environment: Encoding/Decoding Audio Signals." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 175-177.

Bowcott, P. 1990. "High Level Control of Granular Synthesis using the concepts of Inheritance and Social Interaction." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 50-52.

Bowler, I., P. Manning, A. Purvis, and N. Bailey. 1989. "A Transputer-Based Additive Synthesis Implementation." In *Proceedings of the 1989 ICMC (Columbus, Ohio).* San Francisco, CA: ICMA.

Bowler, I., P. Manning, A. Purvis, and N. Bailey. 1990. "New Techniques for a Real-Time Phase Vocoder." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 178-180.

Bowler, I., P. Manning, A. Purvis, and N. Bailey. 1990. "On Mapping N Articulation onto M Synthesiser-Control Parameters." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 181-184.

Bregman, A. S., and J. Campbell. 1971. "Primary Auditory Stream Segregation and Perception of Order in Rapid Sequence of Tones." *Journal of Experimental Psychology.* 89(2): 244-299.

Bresin, R., and A. Vedovetto. 1994. "Neural Network for Musical Tones Compression, Control, and Synthesis." In *Proceedings of the 1994 ICMC (Århus).* San Francisco, CA: ICMA. pp. 368-371.

Brümmer, L. 1994. "Using a Digital Synthesis Language in Composition." *Computer Music Journal.* 18(4): 35-46.

Cavaliere, S., G., di Giugno, and E. Guarino. 1992. "MARS - The X20 device and the SM1000 board." In *Proceedings of the 1992 ICMC (San José, CA).* San Francisco, CA: ICMA. pp. 348-351.

Chamberline, H. 1985. *Musical Applications of Microprocessors.* Indianapolis, Indiana: Hayden Books. ISBN: 0-8104-5768-7.

Chapman, D., M Clarke, M. Smith, and P. Archbold. 1996. "Self-Similar Grain Distribution: A Fractal Approach to Granular Synthesis." In *Proceedings of the 1996 ICMC (Hong Kong).* San Francisco, CA: ICMA. pp. 212-213.

Chareyron, J. 1990. "Digital Synthesis of Self-modifying Waveforms by Means of Linear Automata." *Computer Music Journal.* 14(4): 25-41.

Clarke, E. F. 1982. "Timing in the Performance of Erik Satie's 'Vexations'." *Acta Pyschologica.* 50: 1-19.

Clarke, J. M., P. D. Manning, R. Berry, and A. Purvis. 1988. "VOCEL: New implementation of the FOF synthesis method." In *Proceedings of the 1988 ICMC (Cologne, West Germany).* San Francisco, CA: ICMA. pp. 333-348.

Comerford, P. 1993. "Simulating an Organ with Additive Synthesis." *Computer Music Journal.* 17(2): 55-65.

Cox, S., S. Y. Huang, P. Kelly, J. X. Liu, and F. Taylor. 1992. "An Implementation of Static Functional Process Networks." *Lecture Notes in Computer Science.* 605: 497-512.

Crochiere, R. E., and L. R. Rabiner. 1983. *Multirate Digital Signal Processing.* London: Preintice-Hall International Inc. ISBN: 0-13-605162-6.

Dale, S. M. 1988. *MIDI Interfacing of a transputer based system.* 3H Project, School of Engineering and Applied Science, University of Durham.

Dannenberg, R. 1989. "Real-Time Scheduling and Computer Accompaniment." Mathews, M. V., and J. R. Pierce (eds.) *Current Directions in Computer Music Research.* Cambridge, Massachusetts: MIT Press. ISBN: 0-262-13241-9.

Dechelle, F., M. de Cecco, E. Maggi, and N. Schnell. 1996. "New DSP Application on FTS." In *Proceedings of the 1996 ICMC (Hong Kong).* San Francisco, CA: ICMA. pp. 188-189.

Denyer, P., and D. Renshaw. 1985. *VLSI SIGNAL PROCESSING: A Bit-Serial Approach.* Workington, UK: Addison-Wesley Publishing Co. ISBN: 020 1144042.

Depalle, Ph., and X. Rodet. 1990. "U.D.I. : A Unified D.S.P. Interface for sound signal analysis and synthesis." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 225-227.

Depalle, Ph., and G. Poirot. 1991. "SVP: A Modular System for Analysis, Processing and Synthesis of Sound Signals." In *Proceedings of the 1991 ICMC (Montreal).* San Francisco, CA: ICMA. pp. 161-164.

Desain, P., and H. Honing. 1989. "The Quantization of Musical Time: A Connectionist Approach." *Computer Music Journal.* 13(3): 56-66.

Dingle, A., and I. H. Sudborough. 1993. "Simulation of Binary Trees and X-Trees on Pyramid Networks." *Journal of Parallel and Distributed Computing.* 19: 119-124.

Douglas, A. 1968. *The Electronic Musical Instrument Manual.* London: Sir Isaac Pitman and Sons Ltd. ISBN: 0 273 36193 7.

Feiten, B., and T. Ungvary. 1990. "Sound Data Base using Spectral Analyse Reduction and an Additive Synthesis Model." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 72-64.

Fitz, K., W. Waker, and L. Haken. 1992. "Extending the McAulay-Quatieri Analysis for Synthesis with a Limited Number of Oscillators." In *Proceedings of the 1991 ICMC (San José, CA).* San Francisco, CA: ICMA. pp. 381-382.

Fleming, P. J. (ed.) 1988. *Parallel Processing in Control -- transputer and other architectures.* (IEE computing series; 38.) London: Peter Peregrinud Ltd.

Freed, A., X. Rodet, and Ph. Depalle. 1993. "Synthesis and Control of Hundreds of Sinusoidal Partials on a Desktop Computer without Custom Hardware." In *Proceedings of the 1993 ICMC (Tokyo).* San Francisco, CA: ICMA. pp. 98-101.

Freed, A., X. Rodet, and Ph. Depalle. 1993. "Synthesis and Control of Hundreds of Sinusoidal Partials on a Desktop Computer without Custom Hardware." In *Proceedings of International Conference on Signal Processing Applications and Technology.* pp. 1024-1030.

Freed, A. 1995. "Bring Your Own Control to Additive Synthesis." In *Proceedings of the 1995 ICMC (Banff).* San Francisco, CA: ICMA. pp. 303-306.

Fujinaga, I., and J. Vantomme. 1994. "Genetic Algorithms as a Method for Granular Synthesis Regulation." In *Proceedings of the 1994 ICMC (Århus).* San Francisco, CA: ICMA. pp. 138-141.

George, E. B., and M. J. T. Smith. 1992. "Analysis-by-Synthesis/Overlap-Add Sinusoidal Modeling Applied to the Analysis and Synthesis of Musical Tones." *Journal of Audio Engineering Society* 40(6): 497-516.

Germain, C., J. L. Bechennec, D. Etiemble, and J. P. Sansonnet. 1993. "A Communication Architecture for a Massively Parallel Message-Passing Multicomputer." *Journal of Parallel and Distributed Computing.* 19: 338-348.

Gogins, M. 1995. "Gabor Synthesis of Recurrent Iterated Function Systems." In *Proceedings of the 1995 ICMC (Banff).* San Francisco, CA: ICMA. pp. 349-350.

Goodwin, M., and X. Rodet. 1994. "Efficient Fourier Synthesis of Nonstationary Sinusoids." In *Proceedings of the 1994 ICMC (Århus).* San Francisco, CA: ICMA. pp. 333-334.

Goodwin, M., and A. Kogon. 1995. "Overlap-Add Synthesis of Nonstationary Sinusoids." In *Proceedings of the 1995 ICMC (Banff).* San Francisco, CA: ICMA. pp. 355-356.

Graps, A. 1995. "An Introduction to Wavelets." In *IEEE Computational Science and Engineering.* Summer 1995. 2(2).

Gupta, A. K., and S. E. Hambrusch. 1993. "Multiple Network Embeddings into Hypercubes." *Journal of Parallel and Distributed Computing.* 19 (1993): 73-82.

Haken, L. 1992. "Computational Methods for Real-Time Fourier Synthesis." *IEEE Transactions on Signal Processing.* 40(9): 2327-2329.

Haken, L. 1995. "Real-Time Timbre Modification using Sinusoidal Parameter Streams." In *Proceedings of the 1995 ICMC (Banff).* San Francisco, CA: ICMA. pp. 162-163.

Hamman, M. 1991. "Mapping Complex Systems Using Granular Synthesis." In *Proceedings of the 1991 ICMC (Montreal).* San Francisco, CA: ICMA. pp. 475-478.

Helmuth, M., and A. Ibrahim. 1995. "The FCurve Sound Generator with Granular Synthesis." In *Proceedings of the 1995 ICMC (Banff).* San Francisco, CA: ICMA. pp. 63-64.

Higgins, R. J. 1990. *Digital Signal Processing in VLSI.* New Jersey: Prentice Hall. ISBN: 012212887X.

Holm, F. 1994. "Control of Frequency and Decay in Oscillating Filters Using Multirate Technique." In *Proceedings of the 1994 ICMC (Århus).* San Francisco, CA: ICMA. pp. 372-375.

Hou, E. S. H., N. Ansari, and H. Ren. 1994. "A Genetic Algorithm for Multiprocessor Scheduling." *IEEE Transaction on Parallel and Distributed Systems.* 5(2): 113-120.

Houghton, A. D., A. J. Fisher, and T. F. Malet. 1995. "An ASIC for Digital Additive Sine-wave Synthesis." *Computer Music Journal.* 19(3): 26-31.

Hu, Y. H. 1992. "The Quantization Effects of the CORDIC Algorithm." *IEEE Transactions on Signal Processing.* 40(4) April 1992: 834-844.

INMOS Limited. 1988. *Transputer Development System.* Hemel Hempstead, Herts.: Prentice Hall International (UK) Ltd.

INMOS Limited. 1998. *occam 2 Reference Manual.* Hemel Hempstead, Herts.: Prentice Hall International (UK) Ltd.

Itagaki, T., P. D. Manning, and A. Purvis. 1995. "An Implementation of Real-time Granular Synthesis on a Multi-processor Network." In *Proceedings of the 1995 ICMC (Banff, CANADA).* San Francisco, CA: ICMA. pp. 493-494.

Itagaki, T., D. J. E. Nunn, D. K. Phillips, D. Batjakis, A. Purvis, and P. D. Manning. 1995. "Activity Report from Durham Music Technology." In *the Proceedings of XI CIM Colloquio di Informatica Musicale*, November 1995, Bologna, Italy, pp. 51-54. (in English)

Itagaki, T., P. D. Manning, and A. Purvis. 1997. "Distributed Parallel Processing: Lessons Learned from a 160-Transputer Network." *Computer Music Journal.* 21(4): 42-54, back cover.

Jaffe, D. A. 1985. "Ensemble Timing in Computer Music." *Computer Music Journal.* 9(4): 38-48.

Jaffe, D. A. 1990. "Efficient Dynamic Resource Management on Multiple DSPs, as implemented in the NeXT Music Kit." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 188-190.

Jaffe, D. A., and J. O. Smith. 1993. "Real Time Sound Processing & Synthesis on Multiple DSPs Using the Music Kit and the Ariel QuintProcessor." In *Proceedings of the 1993 ICMC (Tokyo).* San Francisco, CA: ICMA. pp. 464-467.

Jansen, C. 1992. "Sine Circuitu." In *Proceedings of the 1991 ICMC (San José, CA).* San Francisco, CA: ICMA. pp. 451-452.

Kaper, H., D. Ralley, J. Restrepo, and S. Tipei. 1995. "Additive Synthesis with DIASS_M4C on Argonne National Laboratory's IBM POWERparallel System (SP)." In *Proceedings of the 1995 ICMC (Banff).* San Francisco, CA: ICMA. pp. 351-352.

Kaplane, S. J. 1981. "Developing a Commercial Digital Sound Synthesizer." In Road. C. (ed.) 1989. *The Music Machine.* London, England: MIT Press. pp. 611-622.

Kirk, R., and R. Orton 1990. "MIDAS: A Musical Instrument Digital Array Signal Processor." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 127-131.

Kriese, C., and S. Tipei. 1992. "A compositional approach to additive synthesis on supercomputers." In *Proceedings of the 1991 ICMC (San José, CA).* San Francisco, CA: ICMA. pp. 394-395.

Kronland-Martinet, R. 1988. "The Wavelet Transform for Analysis, Synthesis, and Processing of Speech and Music Sound." *Computer Music Journal.* 12(4): 11-20.

Lee, M., A. Freed, and D. Wessel. 1991 "Real-Time Neural Network Processing of Gestural and Acoustic Signals." In *Proceedings of the 1991 ICMC (Montreal).* San Francisco, CA: ICMA. pp. 277-280.

Lee, M., and D. Wessel. 1992 "Connectionist Models for Real-Time Control of Compositional Algorithms." In *Proceedings of the 1991 ICMC (San José , CA).* San Francisco, CA: ICMA. pp. 277-280.

Lee, P. J. 1986. "Design of a Transputer Evaluation System." MSc Electronics Project Report, University of Durham.

Lewis, T. G., and H. El-Rewini. 1992. *Introduction to Parallel Computing.* London: Prentice-Hall International Inc.

Lin, Y. C. 1993. "Perfectly Overlapped Generation of Long Runs for Sorting Large Field." *Journal of Parallel and Distributed Computing.* 19: 136-142.

Lippe, C., and M. Puckette. 1991. "Musical Performance Using the IRCAM Workstation." In *Proceedings of the 1991 ICMC (Montreal).* San Francisco, CA: ICMA. pp. 533-535.

Loy, G. 1985. "Musicians Make a Standard: The MIDI Phenomenon." *Computer Music Journal.* 9(4): 8-26.

Maggi, E., and F. Armani. 1994. "The MARS Station: Algorithm Design and Real Time Performance." In *Proceedings of the 1994 ICMC (Århus).* San Francisco, CA: ICMA. pp. 237-238.

Maggi, E., and F. Dechelle. 1996 "The evolution of the graphic editing environment for the IRCAM musical workstation." In *Proceedings of the 1996 ICMC (Hong Kong).* San Francisco, CA: ICMA. pp.185-187.

Manning, P. 1993. *Electronic and Computer Music.* (second edition) Oxford: Oxford University Press. ISBN: 0-19-816328-2.

Marans, M. 1991. "Timing is Everything." *Keyboard.* 1991 (December): 95-103.

Marans, M. 1992. "Timing is Still Everything." *Keyboard.* 1992 (January): 35-37.

Mathews, M. V., and J. R. Pierce (eds.). 1989. *Current Directions in Computer Music Research.* (paperback edition, 1991) Cambridge, Massachusetts: MIT press. ISBN: 0-262-63139-3.

McAulay, R. J., and T. F. Quatieri. 1986. "Speech Analysis/Synthesis Based on a Sinusoidal Representation." *IEEE Transactions on Acoustics, Speech, and Signal Processing.* ASSP-34(4): 744-754.

McGee, W. F., and P. Merkley. 1991. "A Real-Time Logarithmic-Frequency Phase Vocoder." *Computer Music Journal.* 15(1): 20-27.

McMillen, K., D. L. Wessel, and M. Wright. 1994. "The ZIPI Music Parameter Description Language." *Computer Music Journal.* 18(4): 52-73.

McMillen, K., D. L. Wessel, and M. Wright. 1994. "A Summary of the ZIPI Network." *Computer Music Journal.* 18(4): 74-80.

Miguet, S., and Y. Robert. 1992. "Reduction Operation on a Distributed Memory Machine with a Reconfigurable Interconnection Network." *IEEE Transactions on Parallel and Distributed Systems.* 3(4): 500-505.

Miranda, E. R. 1995. "Cellular Automata Synthesis of Acoustic Particles." In *Proceedings of the 1995 ICMC (Banff).* San Francisco, CA: ICMA. pp. 233-234.

Moog, R. A., and T. L. Rhea. 1990. "Evolution of the Keyboard Interface: The Bösender 290 SE Recording Piano and the Moog Multiply-Touch-Sensitive Keyboards." *Computer Music Journal.* 14(2): 52-60.

Moore, F. R. 1977. *Realtime Interactive Computer Music Synthesis.* Doctoral dissertation, Department of Electrical Engineering, Stanford University.

Motorola. 1990. *DSP56000/DSP56001 Digital Signal Processor User's Manual.* Phoenix, Arizona: Motorola.

van Oostrum, P. 1996. "Bibliography on synthesizers, Midi, Computer and Electronic Music" on URL=ftp://ftp.cs.ruu.nl/pub/MIDI/DOC/bibliography.html

Orlarey, Y. 1990. "An Efficient Scheduling Algorithm for Real-Time Musical Systems." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 194-198.

Orton. R, A. Hunt, and R. Kirk. 1991. "Graphical Control of Granular Synthesis using Cellular Automata and the Freehand Program." In *Proceedings of the 1991 ICMC (Montreal).* San Francisco, CA: ICMA. pp. 416-418.

Otto, P., R. Bidlack, and S. Master. 1992. "-MixNet- A Comprehensive Digital Audio Production System." In *Proceedings of the 1991 ICMC (San José , CA).* San Francisco, CA: ICMA. pp. 247-248.

Ovans, R., D. Murphy, and T. Bartoo. 1995. "THE INFINITE DELAY LINE Granulation as an In-Line Effect." In *Proceedings of the 1995 ICMC (Banff).* San Francisco, CA: ICMA. pp. 241-242.

Palmer, C. 1989. "Mapping Musical Thought to Musical Performance." *Journal of Experimental Psychology: Human Perception and Performance* 15(12): 331-346.

Palmieri, G., and S. Sapir. 1992. "MARS - Musical Applications." In *Proceedings of the 1992 ICMC (San José, CA).* San Francisco, CA: ICMA. pp. 352-353.

Parash, A., and U. Shimony. 1992. "An Expandable Real-Time Transputer Sound Generator." In *Proceedings of the 1991 ICMC (Montreal).* San Francisco, CA: ICMA. pp. 226-228.

Parent, N. 1991. "Parametric Spectrumization." In *Proceedings of the 1991 ICMC (Montreal).* San Francisco, CA: ICMA. pp. 127-128.

Penfold, R. A. 1986. *MIDI projects.* London: Bernard Babani LTD.

Phillips, D., A. Purvis, and S. Johnson. 1994. "A Multirate Optimisation for Real-Time Additive Synthesis." In *Proceedings of the 1994 ICMC (Århus).* San Francisco, CA: ICMA. pp. 364-367.

Pinkston, R. F. 1990. "DSP-Sound: A Software Synthesis Package for Real-Time DSP-based Systems." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 199-201.

de Poli, G., A. Picciali, and C. Roads. (eds.) 1991. *Representations of Musical Signals.* Cambridge, Massachusetts: London: MIT Press. ISBN 0-262-04113-8.

Pope, S. T., and G. van Rossum. 1995. "Machine Tongues XVIII: A Child's Garden of Sound File Formats." *Computer Music Journal.* 19(1): 25-63.

Pountain, D. 1987. *A tutorial introduction to OCCAM programming.* Bristol: INMOS Ltd.

Raytchev, R. 1993. "Global representation of local time measurements in transputer networks." *Microprocessing and Microprogramming.* 36: 215-222.

Roads, C. 1978. "Automated Granular Synthesis of Sound." *Computer Music Journal.* 2(2): 61-62.

Roads, C. 1988. "Introduction to Granular Synthesis." *Computer Music Journal.* 12(2): 11-13.

Roads, C. (ed.) 1989. *The Music Machine.* London, England: MIT Press. ISBN 0-262-18131-2.

Roads, C. 1993. "Musical Sound Transformation by *Convolution.*" In *Proceedings of the 1993 ICMC (Tokyo).* San Francisco, CA: ICMA. pp. 102-109.

Rodet, X., Y. Potard, and J.B. Barrière. 1984. "The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General." In Road. C. (ed.) 1989. *The Music Machine.* London, England: MIT Press. pp. 449-465.

Rubine, D., and P. McAvinney. 1990. "Programmable Finger-tracking Instrument Controllers." *Computer Music Journal.* 14(1): 26-40.

Rumsey, F. 1990. *MIDI systems and control.* Cambridge, Massachusetts: Butterworth & Co. Ltd.

Serra, M., D. Rubine, and R. Dannenberg. 1990. "Analysis and Synthesis of Tones by Spectral Interpolation." *Journal of Audio Engineering Society.* 38(3). pp. 111-128.

Shen, H. 1992. "Occam implementation of process-to-processor mapping on the Hathi-2 transputer system." *Microprocessing and Microprogramming.* 33: 173-189.

Shimony, U., and Y. Zarfati. 1990. "Second-Order Universal Processing Device for Real-Time Music Synthesis." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 205-207.

Sindler K. W. 1984. "Dynamic Timbre Control for Real-Time Digital Synthesis." *Computer Music Journal.* 8(1): 28-42.

Smith, J. O. 1985. "Fundamentals of Digital Filter Theory." In Road. C. (ed.) 1989. *The Music Machine.* London, England: MIT Press. pp. 509-519.

Smith III, J. O. 1991. "Viewpoints on the History of Digital Synthesis." In *Proceedings of the 1991 ICMC (Montreal).* San Francisco, CA: ICMA. pp. 1-10.

Smith, J. O., and J. B. Angell. 1982. "A Constant-Gain Digital Resonator Tunes by a Single Coefficient." *Computer Music Journal.* 6(4): 36-40.

Spanier, J. R., S. Johnson, and A. Purvis. 1996. "Optimisations of the FOF Algorithm for VLSI Implementaion." In *Proceedings of the 1996 ICMC (Hong Kong).* San Francisco, CA: ICMA. pp 126-128.

Steffensen, K., B. Slipsager, M. Folmer, and S. Brandorff. 1993. "DIEM Multi DSP box for live performance." In *Proceedings of the 1993 ICMC (Tokyo).* San Francisco, CA: ICMA. pp. 409-411.

Strasburger, H., S. Kohler, and I. Radauer. 1990. "Score Input to CSound via the Midi Keyboard." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. p.208.

Sundberg, J., and V. Verrilllo. 1980. "On the anatomy of the retard: A study of timing in music." *Journal of the Acoustical Society of America* 68(3): 772-779.

Sundberg, J., L. Frydén, and A. Askenfelt. 1983. "WHAT TELLS YOU THE PLAYER IS MUSICAL? An analyses-by-synthesis study of music performance." In Sundberg, J. (ed.) *Studies of Music Performance.* No. 39. Royal Swedish Academy of Music. pp. 61-75.

Sundberg, J., A. Friberg, and L. Frydén. 1991. "Common Secrets of Musicians and Listeners: An analysis-by-synthesis Study of Musical Performance." In Howell, P., R. West. and I. Cross. (eds.) *Representing Musical Structure.* London: Academic Press Limited.

Tan, B. T .G., and S. L. Gan. 1993. "Real-Time Implementation of Asymmetrical Frequency-Modulation Synthesis." *Journal of Audio Engineering Society.* 41(5):357-363.

Texas Instruments. 1991. *TMS320C4x User's Guide.* Houston, Texas: Texas Instruments.

Transtech Devices Limited. 1989. *Transtech TMB04 User Manual.* High Wycombe, Bucks.: Transtech Devices Ltd.

Truax, B. 1990. "Chaotic Non-linear Systems and Digital Synthesis: An Exploratory Study." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 100-103.

Truax, B. 1991. "Composition with Time-Shifted Environmental Sound using a Real-Time Granulation Technique." In *Proceedings of the 1991 ICMC (Montreal).* San Francisco, CA: ICMA. pp. 487-490.

Truax, B. 1993. "Time-Shifting and Transposition of Sampled Sound with a Real-Time Granulation Technique." In *Proceedings of the 1993 ICMC (Tokyo).* San Francisco, CA: ICMA. pp. 82-85.

Truax, B. 1994. "Discovering Inner Complexity: Time Shifting and Transposition with a Real-time Granulation Technique." *Computer Music Journal.* 18(2): 38-48.

Truax, B. 1996. "Time-Stretching of Hyper-Resonated Sound Using a Real-Time Granulation Technique." In *Proceedings of the 1996 ICMC (Hong Kong).* San Francisco, CA: ICMA. pp. 491-492.

Tyrrell, A. M., D. M. Howard, and N. A. Beasley. 1992. "Transputer Model of the Human Peripheral Hearing System." *Microprocessing and Microprogramming.* 35: 619-624.

Vail, M. 1991. "Digital Pianos." *Keyboard.* April 1991. pp 80-102.

Vercoe, B. and D. Ellis. 1990. "Real-Time CSOUND: Software Synthesis with Sensing and Control." In *Proceedings of the 1990 ICMC (Glasgow).* San Francisco, CA: ICMA. pp. 209-211.

Voss, R. F., and J. Clarke. 1978. "1/f Noise in Music: Music from 1/f Noise." *Journal of the Acoustical Society of America* 63(1): 258-263.

Weinreich, G. 1979. "The Coupled Motions of Piano Strings." *Scientific American.* January 1979. pp. 94-102.

Winker, T. 1992. "FollowPlay: A MAX Program for Interactive Composition." In *Proceedings of the 1992 ICMC (San José, CA).* San Francisco, CA: ICMA. pp. 433-434.

Wright, M. 1994. "Examples of ZIPI Application." *Computer Music Journal.* 18(4): 81-85.

Wright, M. 1994. "A Comparison of MIDI and ZIPI." *Computer Music Journal.* 18(4): 86-91.

Wright, M. 1994. "Answers to Frequently Asked Questions about ZIPI." *Computer Music Journal.* 18(4): 92-96.

Zaveršek, P., and P. Kolbezen. 1992. "Dynamic Allocation on the Transputer Network." *Lecture Notes in Computer Science.* (634): 825-826.

abbreviations used in this Bibliography

ICMC: International Computer Music Conference
ICMA: International Computer Music Association
CA: California

Publications based on the work in Durham

Itagaki, T., A. Purvis, and P. D. Manning. 1994. "Real-time Synthesis on a Multi-processor Network." In *Proceedings of the 1994 International Computer Music Conference (Århus, Denmark).* San Francisco, California: International Computer Music Association. pp. 382-385.

Itagaki, T., P. D. Manning, and A. Purvis. 1995. "An Implementation of Real-Time Granular Synthesis onto a Multi-Processor Network." In *Proceedings of the 1995 International Computer Music Conference (Banff, Canada).* San Francisco, California: International Computer Music Association. pp. 493-494.

Itagaki, T., D. K. Phillips, P. D. Manning, and A. Purvis. 1995. "An Implementation of Optimised Method for Real-time Sound Synthesis on a Multi-processor Network." In *Book of Abstracts* of 5th International Conference on Parallel Computing, September 1995, Gent, Belgium. p.100.

Itagaki, T., D. J. E. Nunn, D. K. Phillips, D. Batjakis, A. Purvis, and P. D. Manning 1995. "Activity Report from Durham Music Technology." In *the Proceedings of XI CIM Colloquio di Informatica Musicale*, November 1995, Bologna, Italy. pp. 51-54. (in English)

Itagaki, T., P. D. Manning, and A. Purvis. 1996. "Real-time Granular Synthesis on a Distributed Multi-processor Platform." In *Proceedings of the 1996 International Computer Music Conference (Hong Kong).* San Francisco, California: International Computer Music Association. pp. 287-288.

Itagaki, T., S. Johnson, P. D. Manning, D. J. E. Nunn, D. K. Phillips, A. Purvis, and J. R. Spanier. 1996. "Durham Music Technology: Activity Report." In *Proceedings of the 1996 International Computer Music Conference (Hong Kong).* San Francisco, California: International Computer Music Association. pp. 126-128.

Itagaki, T., P. D. Manning, and A. Purvis. 1997. "Distributed Parallel Processing: Lessons Learned from a 160-Transputer Network." *Computer Music Journal.*21(4): 42-54, back cover.

Glossary of Terms and Abbreviations

AC	Alternative Current
acyclic graph	a graph without any cycles
ADC	Analogue to Digital Converter
additive synthesis	Fourier synthesis, sound synthesised by super-position of sine waves.
address generation	a process during execution of an instruction in which an effective address is calculated by means of indexing or indirect addressing.
AES	Audio Engineering Society
AM	Amplitude Modulation
ANSI	American National Standards Institute
array processor	A computer designed primarily to perform data parallel calculations on arrays or matrices. The two principle architectures used are the processor array and the vector processor.
asynchronous	A method of transmission which does not require a common clock, but separates fields of data by stop and start bits.
BPS	bytes per second, a unit of memory access speed or communications transfer speed.
CD	Compact Disc, a digital sound storage media, 16-bit format at 44.1 kHz sampling rate.
chain	A topology in which every processor is connected to two others, except for two end processors that are connected to only one other.
channel	A point-to-point connection between two processes or processors through which messages can be sent. Programming systems that rely on channels are sometimes called connection-oriented, to distinguish them from the more widespread connectionless systems in which messages are sent to named destinations rather than through named channels.
CISC	Complicated Instruction Set Computer; a computer that provides many powerful but complicated instructions. This term is also applied to software designs that give users a large number of complex basic operations. See also RISC.
clock cycle	The fundamental period of time in a computer.

clock time — Physical or elapsed time, as seen by an external observer. Non relativistic time.

communication overhead — A measure of the additional workload incurred in a parallel algorithm due to communication between the nodes of the parallel system.

CMOS — Complementary Metal Oxide on Silicon

concurrent compute — A generic category, often used synonymously with parallel computer to include both multi-computer and multi-processor.

concurrent processing — simultaneous execution of instructions by two or more processors within a computer.

configuration — A particular selection of the types of processes that could make up a parallel program. Configuration is trivial in the SPMD model, in which every processor runs a single identical process, but can be complicated in the general MIMD case, particularly if user-level processes rely on libraries that may themselves require extra processes.

control process — A process which controls the execution of a program on a concurrent computer. The major tasks performed by the control process are to initiate execution of the necessary code on each node and to provide I/O and other service facilities for the nodes.

co-processor — an additional processor attached to a main processor, to accelerate arithmetic, I/O or graphics operations.

CORDIC — COdinate Rotation DIgital Computer

CPU — Central Processing Unit

cube-connected cycles network — a processor organisation that is a variant of a hyper cube. Each hyper cube node becomes a cycle of nodes, and no node has more than three connections to other nodes.

CSOUND — software synthesis language (Vercoe 1986)

cycle — a cycle of the computer clock.

DAC — Digital to Analogue Converter

DAT — Digital Audio Tape

DC — Direct Current

deadlock — A situation in which each possible activity is blocked, waiting on some other activity that is also blocked. If a directed graph represents how activities depend on others, then deadlock arises if and only if there is a cycle in this graph.

DIN	German Industrial Standard: *Deutsche Industrie Norm*
DSP	Digital Signal Processing
distributed computer	A computer made up of many smaller and potentially independent computers, such as a network of workstations. This architecture is increasingly studied because of its cost effectiveness and flexibility.
distributed memory	Memory that is physically distributed amongst several modules. A distributed memory architecture may appear to users to have a single address space and a single shared memory or may appear as disjoint memory made up of many separate address spaces.
DMA	Direct Memory Access; allows devices on a bus to access memory without requiring intervention by the CPU.
DRAM	Dynamic RAM; memory which periodically needs refreshing, and is therefore usually slower than SRAM but is cheaper to produce.
EPROM	Electronically Programmable ROM; a memory whose contents can be changed using special hardware. This usually involves removing the chips from their environment in order to "burn" a new pattern into them.
FFT	Fast Fourier Transform is a technique for the rapid calculation of discrete Fourier transform of a function specified discretely at regular intervals. The technique makes use of a butterfly data structure.
FIFO	First In First Out
FIR	Finite Impulse Response (Filter)
FLOPS	Floating Point Operations: unit for measurement of floating point operation
Flynn's Taxonomy	A classification system for architectures that has two axes: the number of instructions streams executing concurrently, and the number of data sets to which those instructions are being applied. The scheme was proposed by Flynn in 1966.
FM	Frequency Modulation
FOF	*Fonction d'Onde Formantique* (French), Formant wave-function
FPU	Floating Point Unit: either a separate chip or an area of silicon on the CPU specialised to accelerate floating point arithmetic.
gps	grain per second: grain density in granular synthesis/sound granulation

guard	A logical condition that controls whether a communication operation can take place. Guards are usually defined as part of the syntax and semantics of CSP-based languages. In OCCAM, guards are used in "ALT"; such as communication port, channel and timer, with or without conditions with Boolean variables.
heterogeneous	Containing components of more than one kind. A heterogeneous architecture may be one in which some components are processors, and others memories, or it may be one that uses different types of processor together.
homogeneous	Made up of identical components. A homogeneous architecture is one in which each element is of the same type; processor arrays and multi-computers are usually homogeneous.
hop	A network connection between two distant nodes.
hot-spot contention	an interference phenomenon observed in multi-processors caused by memory access statistics being slightly skewed from a uniform distribution to favour a specific memory module.
hyper-cube	A topology of which each node is the vertex of a d-dimensional cube. In a binary hyper cube, each node is connected to n others, and its co-ordinates are one of the 2^n different n-bit sequences of binary digits.
interconnection network	the system of logic and conductors that connects the processors in a parallel computer system; such as bus, mesh and hyper cube.
inter-processor communication	the passing of data and information among the processors of a parallel computer during the execution of a parallel program.
I/O	input/output
ICMA	International Computer Music Association
ICMC	International Computer Music Conference
IFFT	Inverse FFT
IIR	Infinite Impulse Response (Filter)
IMW	IRCAM Music Workstation
IRCAM	Paris based research institution: *Institut de Recherché et Coordination Acoustique/Musique.*
IRIS	Paliano (Italy) based research organisation: *Istituto di Ricerca per l'Industria dello Spettacolo.*
ISPW	IRCAM Sound Processing Workstation (= IMW)

i860	general-purpose processor developed by Intel capable of 80 MFLOPS at 40 MHz
latency	The time taken to service a request or deliver a message which is independent of the size or nature of the operation. The latency of a message passing system is the minimum time to deliver a message, even one of zero length that does not have to leave the source processor.
LED	Light- Emitting Diode
link	A one-to-one connection between two processors or nodes in a multi-computer.
load balance	The degree to which work is evenly distributed among available processors. A program executes most quickly when it is perfectly load balanced, that is when every processor has a share of the total amount of work to perform so that all processors complete their assigned tasks at the same time. One measure of load imbalance is the ratio of the difference between the finishing times of the first and last processors to complete their portion of the calculation to the time taken by the last processor.
LSB	Least Significant Bit
mapping	often used to indicate an allocation of processes to processors; allocating work to processes is usually called scheduling.
MARS	Musical Audio Research Station, MARS, developed by IRIS s.r.l. [Italy]. See IRIS.
MAX	interactive graphic programming environment created by Opcode (Puckette and Zicarelli 1990)
message passing	A style of inter-process communication in which processes send discrete messages to one another. Some computer architectures are called message passing architectures because they support this model in hardware, although message passing has often been used to construct operating systems and network software for uni-processors and distributed computers.
MIDI	Musical Instrument Digital Interface, published in 1983 by International MIDI Association [23634 Emelita Street, Woodland Hills, California 91367 USA.]
MIMD	Multiple Instruction, Multiple Data; a category of Flynn's taxonomy in which many instruction streams are concurrently applied to multiple data sets. A MIMD architecture is one in which heterogeneous processes may execute at different rates.

MIPS	Million Instructions Per Second: unit for measurement of processor performance, referring to integer or non-floating point instructions.
MISD	Multiple Instructions, Single Data. A member of Flynn's taxonomy almost never used.
MOPS	Million Operations Per Second, usually used for a general operation, either integer, floating point or otherwise.
motherboard	A printed circuit board or card on which other boards or cards can be mounted. Motherboards will generally have a number of slots for other boards, by which means the computer system may be expanded.
MSB	Most Significant Bit
multi-computer	A computer in which processors can execute separate instruction streams, have their own private memories and cannot directly access one another's memories. Most multi-computers are disjoint memory machines, constructed by joining nodes (each containing a micro-processor and some memory) via links.
multi-processor	A computer in which processors can execute separate instruction streams, but have access to a single address space. Most multi-processors are shared memory machines, constructed by connecting several processors to one or more memory banks through a bus or switch.
multi-programming	the ability of a computer system to time share its (at least one) CPU with more than one program at once.
multi-(sampling)rate	digital sound processing using more than one sampling rate.
multi-tasking	Executing many processes on a single processor. This is usually done by time-slicing the execution of individual processes and performing a context switch each time a process is swapped in or out, but is supported by special-purpose hardware in some computers. Most operating systems support multi-tasking, but it can be costly if the need to switch large caches or execution pipelines makes context switching expensive in time.
NAMM	National Association of Music Merchants, USA based association.
NeXT	a music workstation based on a 68030 as main processor with a 68882 as floating-point co-processor and DSP 56001 as sound processor and a DAC.

network	A physical communication medium. A network may consist of one or more buses, a switch, or the links joining processors in a multi-computer.
NICAM	Nearly Instantaneous Compounded Audio Multiplex, standardised by IBA, BREMA and BBC.
node	generic term used to refer to an entity that accesses a network.
note	1. music event, tone 2: (polyphonic) note, voice, channel: a sound generating unit in a synthesis system.
NTSC	National Television System Committee; a TV/video signal format
Nyquist frequency	the highest frequency that can be produced in a digital audio system; a half of the sampling rate
OCCAM	programming language for the Transputer family
optimal	Cannot be bettered. An optimal mapping is one that yields the best possible load balance; an optimal parallel algorithm is one that has the lowest possible time-processor product.
PAL	Phase Alternation Line; TV/video signal format
parallel computer	A computer system made up of many identifiable processing units working together in parallel. The term is often used synonymously with concurrent computer to include both multi-processor and multi-computer. The term concurrent generally dominates in usage in the USA, whereas the term parallel is the more widely used in Europe.
parallelisation	Turning a serial computation into a parallel one. Also sometimes turning a vector computation into a parallel one. This may be done automatically by a parallelising compiler or (more usually) by rewriting (parts of) the program.
partitioning	process of restructuring a program or algorithm into independent computational segments.
PC	Personal Computer: usually means IBM compatible one
PCB	Printed Circuit Board
pipe	A communication primitive which involves the transmission of information through a linearly connected subset of the nodes of a parallel computer.

pipelining	Overlapping the execution of two or more operations. Pipelining is used within processors by perfecting instructions on the assumption that no branches are going to preempt their execution; and in multi-processors and multi-computers, in which a process may send a request for values before it reaches the computation that requires them.
process	the fundamental entity of the software implementation on a computer system.
processor array	A computer that consists of a regular mesh of simple processing elements, under the direction of a single control processor. Processor arrays are usually SIMD machines, and are primarily used to support data parallel computations.
PWM	Pulse Width Modulation
RAM	Random Access Memory; computer memory which can be written to and read from in any order.
ring	A topology in which each node is connected to two others to form a closed loop.
RISC	Reduced Instruction Set Computer; a computer that provides only a few simple instructions but executes them extremely quickly. RISC machines typically rely on instruction prefetching and caching to achieve higher performance than CISC machines. The term is also applied to software designs that give users a small number of simple but efficient operations
ROM	Read-Only Memory; a computer memory which cannot be written to during normal operation.
routing	The act of moving a message from its source to its destination. A routing technique is a way of handling the message as it passes through individual nodes.
sampling rate (frequency)	the rate at which digital sound samples are taken. It signifies the bandwidth of a digital audio system.
scalar processor	A computer in the traditional von Neumann sense of operating only on scalar data.
scheduling	Deciding the order in which the calculations in a program are to be executed, and by which processes. Allocating processes to processors is usually called mapping.
SCI	Sequential Circuit Interface
SCSI	Small Computer Systems Interface; a hardware standard for interfacing to devices such as discs.

SHARC	Super-Harvard ARchitecture Computer: a family of DSP produced by Analog Devices.
shared memory	Memory that appears to the user to be contained in a single address space and that can be accessed by any process. In a uni-processor or multi-processor, there is typically a single memory unit, or several memory units interleaved to give the appearance of a single memory unit.
shared variables	Variables to which two or more processes have access, or the model of parallel computing in which inter-process communication and synchronisation are managed through such variables.
SIMD	Single Instruction Multiple Data; a category of Flynn's taxonomy in which a single instruction stream is concurrently applied to multiple data sets. A SIMD architecture is one in which homogeneous processes synchronously execute the same instructions on their own data, or one in which an operation can be executed on vectors of fixed or varying size.
SISD	Single Instruction Single Data; a category of Flynn's taxonomy in which a single instruction stream is serially applied to a single data set. Most uni-processors are SISD machines.
SMPTE	Society of Motion Picture and Television Engineers; SMPTE time-code (format); used in film and TV, a data-rate of 2400 bps.
SM1000	sound generation board on IRIS-MARS workstation, consisting of two IRIS X20 DSP chips [25.6 MIPS at 25 MHz each] for sound processing controlled by a Motorola 68302.
SPMD	Single Program, Multiple Data; a category sometimes added to Flynn's taxonomy to describe programs made up of many instances of a single type of process, each executing the same code independently. SPMD can be viewed either as an extension of SIMD, or as a restriction of MIMD.
SRAM	Static RAM; memory which stores data in such a way that it requires no memory refresh cycle and hence has low power consumption. Generally this type of RAM is faster but more expensive than DRAM.
synchronisation	The act of bringing two or more processes to known points in their execution at the same clock time. Explicit synchronisation is often necessary in SPMD and MIMD programs.

synchronous	Occurring at the same clock time. For example, if a communication event is synchronous, then there is some moment at which both the sender and the receiver are engaged in the operation.
TDS	Transputer Development System. The operation system for the host transputer and a host PC
tick	unit for timing in MIDI.
time sharing	Dividing the effort of a processor among many programs so they can run concurrently. Time sharing is usually managed by an operating system.
topology	A family of graphs created using the same general rule or that share certain properties. The processors in a multi-computer, and the circuits in a switch, are usually laid out using one of several topologies, including the mesh, the hyper cube, the butterfly, the torus and the shuffle exchange network.
TRAM	TRAnsputer Modules
Transputer	A single integrated circuit which contains a CPU, communications links, memory and some cache memory. The name transputer refers to a proprietary series of chips manufactured by INMOS, although other node chips have had similar characteristics.
tree	a connected, undirected, acyclic graph. The most commonly encountered tree in computer science is the regular binary tree, in which a root node has two children but no parent, each interior node has a single parent and two children, and leaf nodes have a single parent but no children.
UART	Universal Asynchronous Receiver-Transmit(ter); a standard protocol for device drivers or a integrated circuit.
uni-processor	A computer containing a single processor. The term is generally synonymous with scalar processor.
USI	Universal Synthesizer Interface, proposed by Dave Smith (the president of Sequential Circuits) and Chet Wood in 1981.
VCA	Voltage Controlled Amplifier
VCF	Voltage Controlled Filter
VCO	Voltage Controlled Oscillator
vector processor	A computer designed to apply arithmetic operations to long vectors or arrays. Most vector processors rely heavily on pipelining to achieve high performance. See also array processor.

velocity — amplitude of a MIDI key press information.

VLSI — Very Large Scale Integration; applied to technologies capable of putting hundreds of thousands or more components on a single chip, or sometimes applied to those chips so manufactured.

von Neumann architecture — Used to describe any computer which does not employ concurrency or parallelism. Named after John von Neumann (1903-1957) who is credited with the invention of the basic architecture of current sequential computers.

voice — (see "note")

Appendix 1. Sound Samples

The software for the sound generation, described in Chapters 5, 6 and 8 is basically for real-time sound production, and is not designed for sound recording. I attempted to re-write these programs to accommodate a recording process. Some of them, however, did not perform as designed, due to restrictions and limitations in the hardware, mainly memory shortage, and in the software [see Chapter 2]. In addition, some of the programs reached the limit of the transputers' performance, leaving no room for extra tasks. For the reasons above, some of the implementations are only available for live performance.

All the sound samples attached to this thesis were produced at a 32 kHz sampling rate and monaural format, with two exceptions at a 44.1 kHz sampling rate. The original files were written in a 16-bit raw binary [Zilog format; high-byte followed by low-byte], and could be converted into other formats by the "SOX" program (Norskog 1993). As I did not have access to a PC with a sound-card, I was not able to test the conversion to a WAV format, that is a standard multi-media format for Windows software.

In this edition, an analogue audio cassette tape is attached.

List of Sound Samples

Appendix 2. Pin Layouts of the 160 Transputer Network

PIN No.	column left	column right	PIN No.	column left	column right
1	GND	GND	16	Up Not Reset	Dn Not Analyse
2	GND	GND	17	Up Not Analyse	Up Not Error
3	GND	GND	18	Dn Not Error	Link 00 Out
4	NC	NC	19	Link 00 In	Link 01 Out
5	NC	NC	20	Link 01 In	Link 13 Out
6	GND	GND	21	Link 13 In	Link 02 Out
7	GND	GND	22	Link 02 In	Link 03 Out
8	GND	GND	23	Link 03 In	Link 04 Out
9	NC	NC	24	Link 04 In	Link 05 Out
10	+5V	+5V	25	Link 05 In	Link 06 Out
11	+5V	+5V	26	Link 06 In	Link 07 Out
12	NC	NC	27	Link 07 In	Link 08 Out
13	NC	NC	28	Link 08 In	Link 09 Out
14	NC	NC	29	Link 09 In	Link 10 Out
15	NC	NC	30	Link 10 In	Link 11 Out
			31	Link 11 In	Link 12 Out
			32	Link 12 In	Dn Not Reset

Table A2.1.: 32-Pin Socket Layout of a PCB

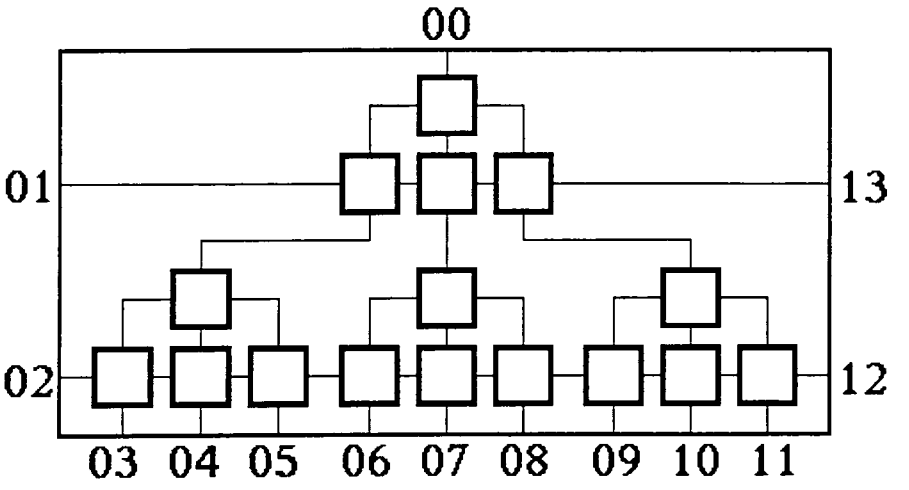


Figure A2.1.: Link Numbering of a PCB

Pin No.	column left	column right
1	Board 9 Link 12 Out	Board 9 Link 12 In
2	Board 9 Link 13 Out	Board 9 Link 13 In
3	Board 1 Link 01 Out	Board 1 Link 01 In
4	Board 1 Link 02 Out	Board 1 Link 02 In
5	Up Not Reset	NC
6	Up Not Analyse	Up Not Error
7	Board 0 Link 00 Out	Board 0 Link 00 In
8	Board 0 Link 01 Out	Board 0 Link 01 In
9	Board 0 Link 13 Out	Board 0 Link 13 In
10	Board 0 Link 02 Out	Board 0 Link 02 In
11	Board 0 Link 12 Out	Board 0 Link 12 In
12	GND	GND
13	GND	GND

Table A2.2.: Pin Layout of Connector out of Main PCB (Tree Top Side)

Pin No.	Column Left	Column Right
1	NC	NC
2	Link 03 In	Link 03 Out
3	Link 04 In	Link 04 Out
4	Link 05 In	Link 05 Out
5	Link 06 In	Link 06 Out
6	Link 07 In	Link 07 Out
7	Link 08 In	Link 08 Out
8	Link 09 In	Link 09 Out
9	Link 10 In	Link 10 Out
10	Link 11 In	Link 11 Out

Table A2.3.: Pin Layout of Connector out of Main PCB
(Tree Bottom side)

Appendix 3. MIDI-to-Transputer Interface

The basic requirements for the interface board are to enable data to be converted between two serial formats; MIDI data and transputer link data, and their transmission speed; 31.25 KBaud and 20 MBaud. MIDI data is sent with a start bit followed by eight data bits and a stop bit. The transmission is continuous and asynchronous. On the other hand, transputer link data is sent with two start bits followed by eight data bits and a stop bit. When the data packet is received, the transputer sends an acknowledgement, a high bit followed by a low bit, to the other end of the link.

Although both data formats are serial, a standard MIDI interface converts serial data into an internal parallel format and *vice-versa*. To convert decoded signals from MIDI to a format suitable for a transputer link an intermediate parallel processing stage is thus necessary. To convert a MIDI serial signal into a parallel format, the Universal Asynchronous Receiver Transmitter [UART] 6402 is recommended in the MIDI specification. Its basic operation is to convert data between serial protocol and a handshake parallel data port. The UART 6402 is a general purpose device. Hence, its users have to define the data format and speed. For receiving and sending the MIDI data, 31.25 KBaud, a clock signal of 500 kHz is recommended. The devices of the transputer family communicate through links. However, to enable the easy connection of links to non-transputer devices, a link adapter is used.

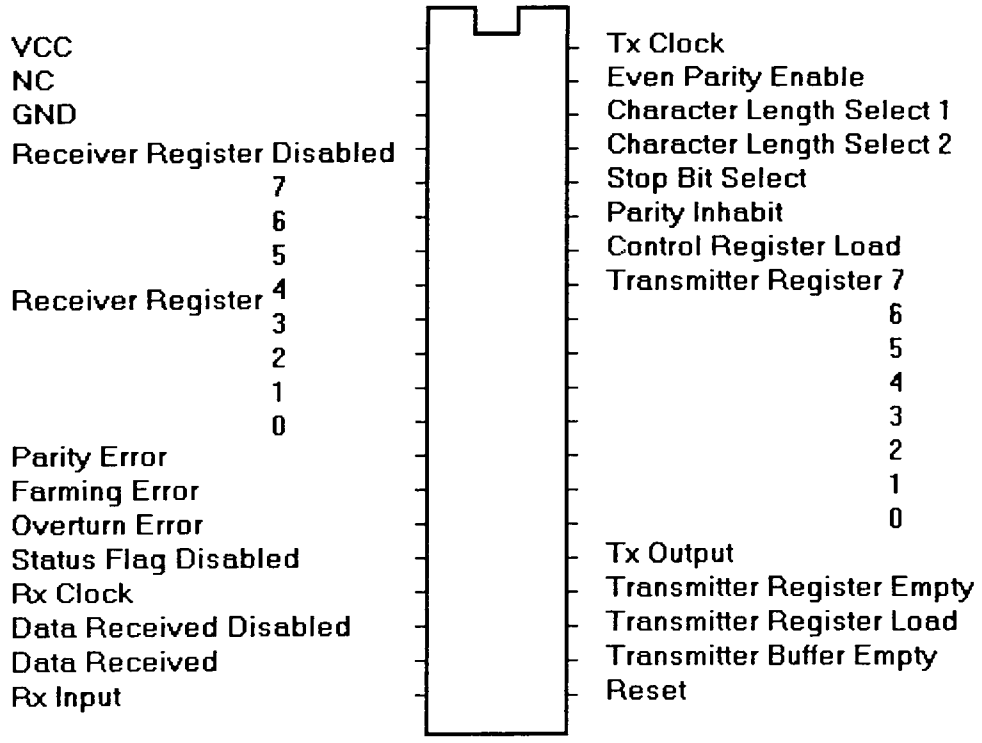


Figure A3.1.: UART 6402 pinout.

C011 is designed for full duplex transputer link communication with a standard microprocessor and sub-system architecture, by converting between bi-directional serial link data and parallel data streams. The link adapter runs at a speed of either 10 Mbits/sec or 20 Mbits/sec, and has two modes. In this project, a 20 Mbits/sec link was used and a C011 was configured into Mode 1; that is where the C011 converts between a parallel link and two independent fully handshaken byte-wide interfaces; one input and one output, for a transputer family device. [Mode 2 is the other way around; from transputer link serial format to parallel format.]

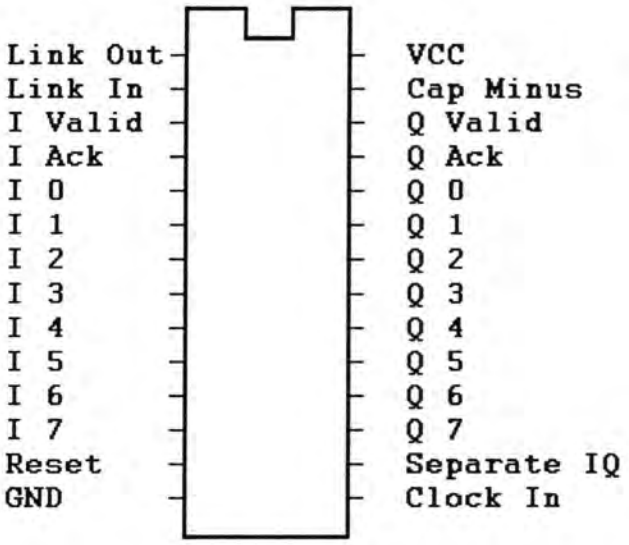


Figure A3.2.: C011 pin-out (Mode 1).

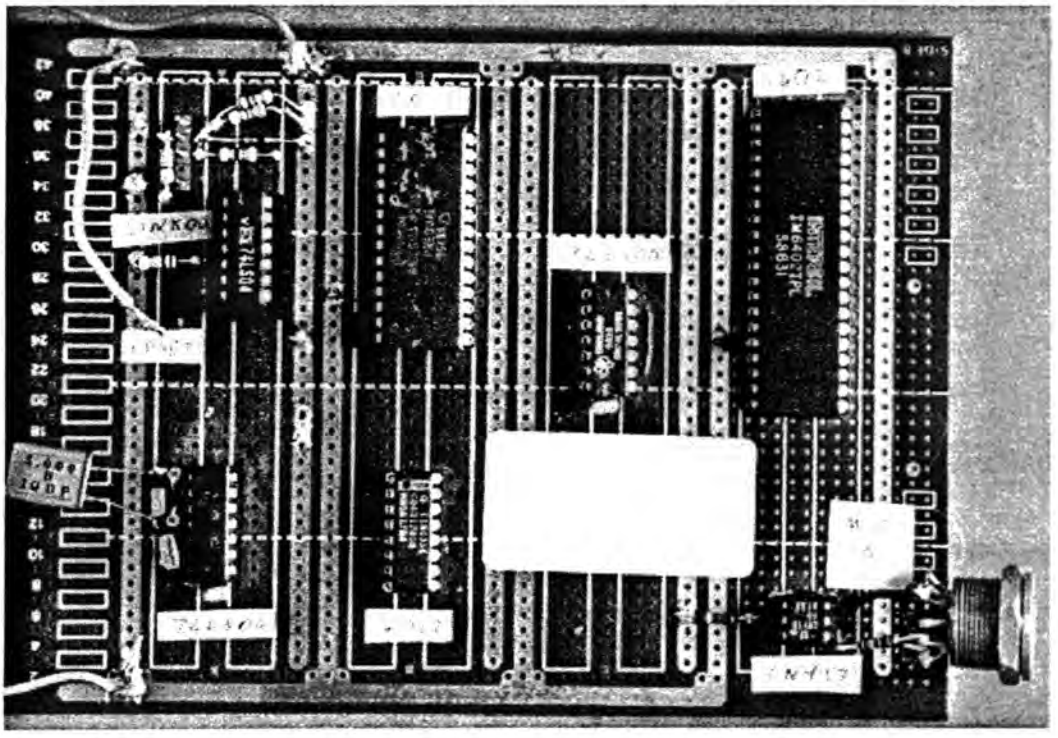


Figure A3.3.: Prototype MIDI-to-Transputer Interface Board.

The handshaking pins of the UART can not be directly connected with those of the C011, due to their incompatibility. An additional logic circuit is required. This circuit was designed and its prototype version built onto a multipurpose circuit board, about nine years ago (Dale 1988). Parts were wire wrapped and the circuit proved unreliable. Eventually the board failed all together.

After some reviews and improvements, a new circuit on a printed circuit board was assembled, as a replacement. The main changes were as follows:

1. To reduce micro disturbance of noise in to the lines,
   a) The number of jumper lines was minimised.
   b) The length of clock and signal circuits were consequently minimised.
2. To obtain a sharp reset signal, a switch was added.

As a result, the circuit board is smaller and offers a higher reliability than the original one. Its diagram and its overview picture are shown below.

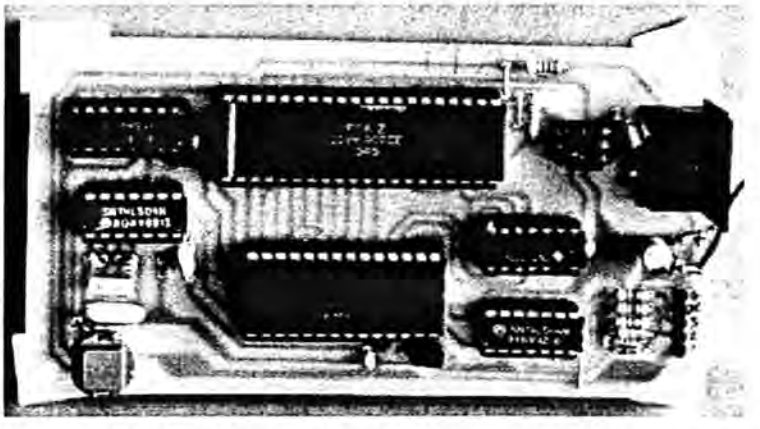


Figure A3.4. : Overview Picture of MIDI-to-Transputer Interface Board.

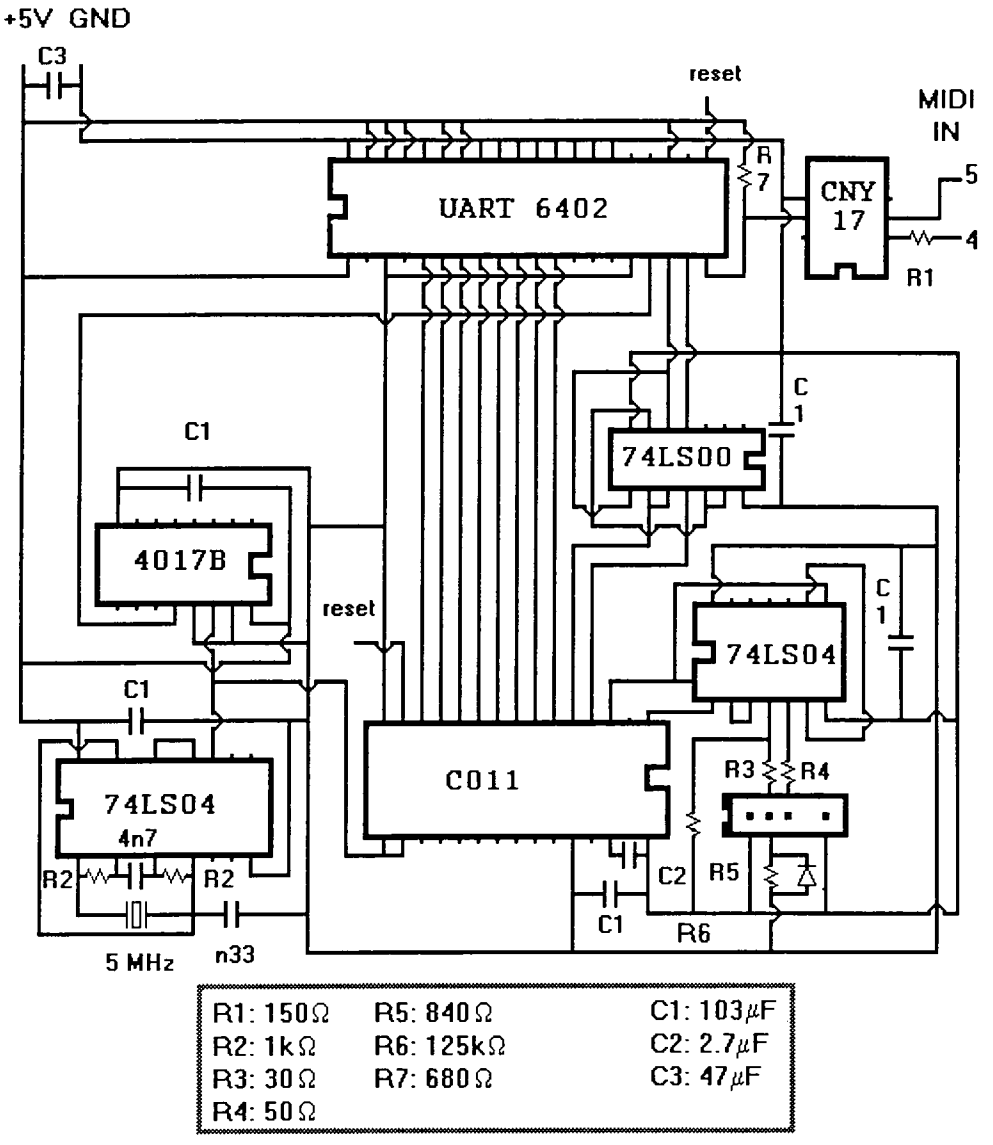


Figure A3.5.: Circuit Diagram of MIDI-to-Transputer Interface Board.

Nevertheless, I discovered some problems with this circuit which may be attributable to the quality of the parts; especially the gate ICs 74LS04 and 74LS00, and the link adapter C011. According to the data sheets for the parts, they should work with a 5 V DC power supply, with an allowance of ±5%. In practice, however, the circuit runs only between 4.80 V and 4.90 V, a strange limitation which was investigated further.

Firstly, I suspected the clock block. Using an oscilloscope, all the clock signal circuits were checked. The circuits work as follows:

An activated crystal creates a 5 MHz signal.

A gate IC 74LS04, which is supposed to work on a 5 MHz signal, cleans the signal.

The clock signal is fed into the C011 and a counter IC 4017B, which forms a 500 kHz signal for the UART.

At this point, a clear 500 kHz equal template square wave should be obtainable if the circuit is driven with a current at the specified potential of 5 V. What I found, however, was that the circuit ran only outside two points; below 4.90 V and over 5.20 V, where the clock signal was stabilised. Between those points, the square wave became distorted; jittering at the rising phase.

By changing the power supply and the chips, and after some trial and error combining different components, the jittering problem was finally cleared. Apart from the quality of the chips, the quality of power supply could have been suspected. Replacement with another power supply, however, resulted in the same operating characteristics and the voltage problem on the circuit has still not been solved.

In the case of over 4.90 V, the UART or the C011 misinterprets the MIDI data, which is usually shifted: for example, from 90h to C0h. In the other case, under 4.80 V, due to the problems in the C011 or the handshaking block, data reflections occurred several times in every one hundred bytes

received; typically every 57 μsec after a packet of MIDI data was received. Around 4.85 V, however, it would seem that all the parts are working well.

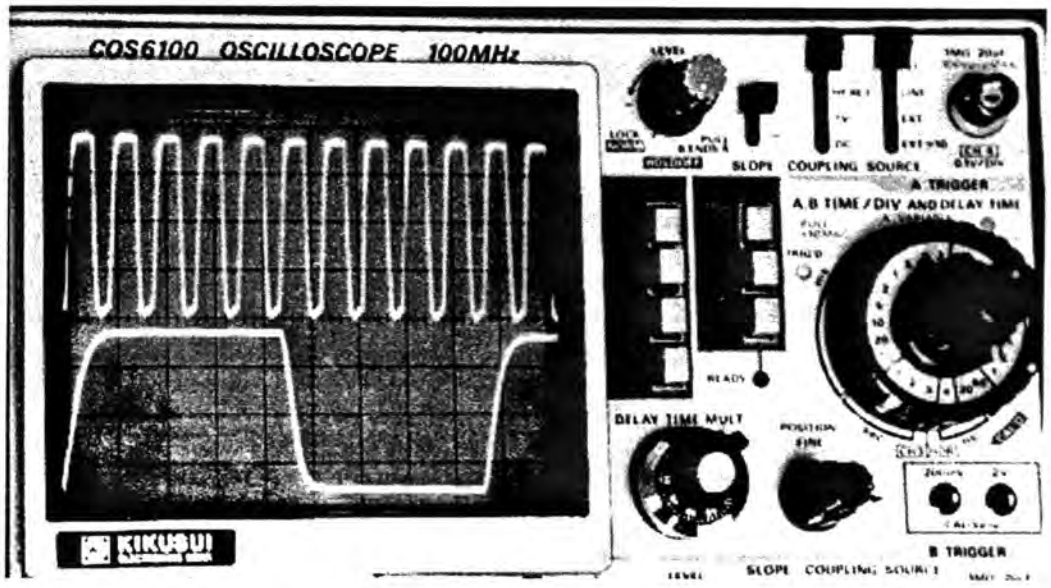


Figure A3.6.: Wave Form at 4.85 V (Top: 5 MHz, Bottom: 500 KHz).