A TRANSPUTER-BASED ADDITIVE SYNTHESIS IMPLEMENTATION

Ian Bowler*, Peter Moore*,
Alan Purria, Nick Bailey

DURHAM MUSIC TECHNOLOGY
Department of Music, University of Durham, U.K.
School of Engineering and Applied Science, University of Durham, U.K.

ABSTRACT

An implementation of the additive synthesis of audio waveforms on a transputer network is described. Some features of transputers that make them especially suitable for implementing parallel sound synthesis systems are then highlighted. It is shown that the bandwidth available for supplying control parameters to a transputer-based pipeline is proportional to the number of transputers in the pipe, unlike conventional systems where the control bandwidth is fixed (and hence will saturate at some stage). On our network of twelve 2144B 20MHz transputers, we have been able, at a sample rate of 40kHz, to generate 60 (i.e., five per transputer) high quality sinusoidal oscillators with full linear interpolation of the amplitude control parameters. By using stochastic sampling techniques, we have also been able to implement 132 (i.e., eleven per transputer) sinusoidal oscillators of lower quality, suitable for sketching or for generating spectral components more than a few decibels below the combined output signal. This is an impressive performance from a general-purpose computing network, and we consider transputer networks to be ideal tools for the development and implementation of many aspects of the control and synthesis of computer music.

MUSIC SYNTHESIS AND HARDWARE

Additive synthesis, whereby a potentially large number of amplitude-controlled sinusoids are summed to form the output signal, has long been regarded as one of the most fundamental in computer music. It is easy to show, using Fourier theory, that any signal can be synthesised by summing a number of sinusoids of appropriate amplitude and phase. In sound synthesis it is found empirically that the phase of the sinusoids is relatively unimportant perceptually, and only the amplitude is usually controlled. The main barrier to the more widespread adoption of the additive synthesis technique seems to lie in the high computational demands of the algorithm and the difficulty in supplying the large number of amplitude and frequency parameters required. We shall show that transputer systems are well suited to overcoming these problems. The generality of additive synthesis is sometimes seen as a drawback, since it may be difficult for a musician to exercise precise control over all the parameters in real-time. This objection is largely overcome by employing higher-level control strategies such as those proposed by Bowler and Greenough (1989) and Kiczkowski (1989). Because of the generality of additive synthesis, it is also well suited to the emulation of other synthesis algorithms such as Granular Synthesis (Gabor 1947, Xenakis 1971).

There have been many implementations of additive synthesis, such as the 4B developed at IRCAM by Alles and diGugno (1977) and that of Snell (1977). These early designs used dedicated hardware to implement specific algorithms and had a relatively low scale of integration. Since these days there has been a trend towards a higher scale of integration and more programmability in music synthesis systems. Programmability allows greater flexibility in the range of tasks that can be performed, while integration improves the cost/performance ratio. Recent implementations, such as the Digidesign Sound Accelerator (Lowe and Currie 1989), are typically based around VLSI integrated circuits optimised for signal processing (DSP chips). In the current implementation, our use of transputers, which are general purpose microprocessors, is a natural continuation of this trend.
TRANSPUTERS IN MUSIC SYNTHESIS

A transputer is one of the family of devices produced by Inmos Ltd, consisting of a microprocessor with memory, process-scheduling and inter-processor communications hardware all integrated on a single chip. These last two features make transputers particularly suitable for implementing parallel computing systems. Standard benchmarks show that the performance of a single transputer is highly competitive with that of other state-of-the-art microprocessors. Although for signal processing and sound synthesis applications a dedicated DSP chip would probably have an edge over a single transputer, we feel this is more than compensated for by the ease with which many transputers can be made to work together in a parallel system. The general-purpose nature of the transputer allows it to be considered not only as a DSP engine but also as a suitable compute base for higher level control functions. This allows the entire system to be specified and controlled using an integrated set of software tools.

Since more than two transputers are usually required to implement a real-time music synthesis system of adequate capability, it is necessary to consider which topology of inter-processor links to use. Given a set of sequential processes that must be executed on a parallel computing network, optimal use of computing resources is made when all processors spend all their time executing the sequential processes: No time is spent servicing communications, or awaiting data because of data dependency or link saturation. The optimal topology of the network is that which achieves optimal use of computing resources with the smallest number of inter-processor links. In a general-purpose computing network, it is necessary that there must exist at least one communication path between any one processor in the network and any other. The simplest (i.e., smallest number of links per processor) topology that achieves this is a linear string of processors: a pipeline. Thus if a pipeline topology achieves optimal use of computing resources, it is the optimal architecture.

A pipeline is an extremely effective architecture for sound synthesis and processing (e.g., Greenough Bowler and Morris 1985). Samples pass down a chain of processors, each of which implements some modification to the sound. In additive synthesis, for example, each processor in a pipeline would perform three processes: take a set of input samples; add a set of N sinusoids to them; output the new sample sequence. Thus processors generate a sample sequence corresponding to the sum of N \times M sinusoids. Conventional processors would require additional hardware to implement a pipeline, and would implement the three processes in sequence. Transputers, however, have all the necessary pipelining hardware on-chip: Each transputer has four bi-directional serial links, each capable of carrying 8 to 12 high quality sound channels. On-chip DMA hardware allows data transfers to proceed during these links, with only roughly 15% overhead on the processor. Thus the three processes can proceed in parallel on each individual transputer. Assuming each processor takes the same amount of time to compute and sum N sinusoids, the buffering prevents any processor having to pause because of data dependencies. Thus to a very good approximation, the pipeline architecture is optimal for this implementation of additive synthesis. This finding is thought to hold more generally for Music Synthesis and Signal Processing implementations: another project of Durham Music Technology (Bailey et al 1986) seems to suggest that the pipeline topology works well for a wide range of Music Synthesis tasks.

Only two links on each transputer are required to implement a pipeline, leaving two free for receiving control parameters. An important advantage of transputer-based signal processing pipelines now becomes apparent: The communications bandwidth available for control is proportional to the number of transputers in the pipeline, thus avoiding bottlenecks in the supply of control parameters. Because transputers are general-purpose device, they are also well suited to the task of controlling the sound-synthesis programs, allowing complete systems to be built with just one type of computing device: the usual split between 'controlling host' and 'dedicated synthesis'
hardware is avoided: The feasibility of this approach has been demonstrated previously (Purvis et al. 1988).

IMPLEMENTATION

The transputer system used for the current research consisted of a network of twelve 3U-sized cards, developed in house, each having an IMS T414B 20MHz 33-bit fixed point transputer with 256kB of fast (100ns, zero wait-state) external static RAM and support logic (total 14 chips per board). We intend to upgrade this system shortly to use higher performance T800 transputers with fast floating point capability. Equivalent systems are available commercially. This network is interfaced to a high quality 16-bit Digital to Analogue converter system.

The language chosen for the implementation on the transputer was OCCAM (Pountain and May 1987). OCCAM is a high-level language developed along with the transputer which allows the key features of transputer networks to be exploited to the full: inspection of the code generated by the OCCAM compiler has confirmed that there would be little to be gained in this application by using assembler code. Using OCCAM, procedures to implement sequential algorithms are developed in a way similar to that used in other high-level languages. The instructions which control the parallelism of the system need only be considered at an outer level of the program: Indeed it is possible to develop and run code on a single processor and only port it to a larger network when performance requirements dictate. Although higher-level languages such as C, Fortran and Lisp are available for the transputer, it is less easy to exploit the parallel processing capabilities of the network with these basically sequential languages, and they are likely to be less efficient.

Since the multiply instruction on the T414B transputer can in some circumstances take over a microsecond to complete, it was decided to implement table lookup oscillators for efficiency, since only one multiply per sinusoid per sample is required. To ensure high quality oscillators, a table of 32768 32-bit samples was used. For each oscillator at each sample time, the phases of the frequency and amplitude accumulators were updated according to the input control parameters. The corresponding samples from the sine table and log-amplitude lookup table were then multiplied together and added to the output sample sequence. To minimise overheads from loop and link initialisation, the processing was performed in blocks of 32 samples. These steps were performed in parallel with the processes which communicate with other pipeline processors. A control parameter for each oscillator consisted of a 24-bit frequency control word and an 8-bit (log: 0.25dB steps) amplitude control word. These were supplied once for every 32 output samples, and the pipeline transputers performed a linear interpolation between successive values at the sample output rate to ensure smooth transitions. The sample rate chosen for experiments was 40kHz, it being anticipated that transputers further down the pipeline would perform any necessary rate conversion to whatever output rate was required. Using different sample rates for generation would scale computational demands proportionately. In this configuration, each transputer was capable of generating 5 sinusoids, i.e. a total of 60 sinusoids for the entire network.

A larger number of oscillators can be obtained if such high sound quality is not required: for example when sketching or when generating spectral components that are a few dB below the combined output signal. Using a 256-word wavetable and limited amplitude interpolation, it was possible to generate 11 sinusoids per transputer (122 for the network). The subjective quality of these oscillators was improved by stochastic sampling techniques (Wold and Dippe 1989): By adding random noise to the frequency control parameter, the table lookup error is no longer correlated with the signal. Thus rather than alias components being produced at a well-defined frequencies, noise is produced which is spread throughout the audio bandwidth. When Q signals generated in this way are summed, the noise increases as the square root of Q, whereas the
error signal increases linearly with \( Q \) in the case of aliases. Thus as the number of sinusoids increases, the signal to noise ratio of the stochastically sampled signals increases.

**CONCLUSIONS AND FUTURE WORK**

Transputer networks can be used to implement additive synthesis in a way that is both cost-effective and does not suffer from the control parameter supply bottleneck that has plagued some other implementations. Such networks also seem well suited to a wide variety of tasks in the control and generation of computer music.

**REFERENCES**


