PARALLEL COMPUTING AND TRANSPUTER APPLICATIONS
M. Valero, E. Oñate, M. Jane, J.L. Larriba and B. Suárez (Eds.)
IOS Press/CIMNE, Barcelona 1992

PARALLEL SOUND SYNTHESIS: PARALLEL SONIFICATION OF MULTIVARIATE ·DATA ON A TRANSPUTER PLATAFORM

M. van Muiswinkel, L.H. Trip and P.M.A. Sloot*
High Performance Computing Group
Department of Computer Systems, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam
THE NETHERLANDS

SUMMARY.

In this paper we propose a parallel sound synthesis model based on the well defined VOice SIMulation (VOSIM) model. Our major interest is in the parallelisation of this model and the mapping onto a transputer platform. We designed transputer based hard- and software for sound synthesis from multivariate data sets to actual audible sound.

1. INTRODUCTION.

With the advent of computational physics gigabytes of data resulting from a single experiment need to be interpreted by scientists. Several tools are in use to present this data in an edible form. Most of these tools work in the visual domain. As a counterpart of visualization, sound can be used as an orientation guide to locate 'hot spots' of important information in a multidimensional dataset. This acoustic "visualization" is also known as audiolization or sonification.

To get meaningful sound from experimental data, a linear transform model is required. Linear dependencies in the dataset should result in perceivable linear dependencies in sound. If this can be established for some number of dimensions the applied transform can be made orthogonal. Since there rarely exists a direct, linear relationship between an arbitrary dataset and a sound signal in terms of perception (like Speeth's seismometer sounds [1]), the transform can be split up in two parts. The first part transforms the data to a linear representation in different auditory domains like intensity, pitch, timbre etc. The second part generates a time domain signal from the transformed data.

In this paper we present the second part of the transform: sound synthesis, its model and implementation. From this model an analysis of the computational requirements of sound synthesis is desired, showing the need for parallel processing. We base our sound synthesis algorithm on the well established VOSIM (VOice SIMulation) model [2], and report on the scalability of a parallel implementation of this model on a Transputer network.

2. BACKGROUND.

2.1 Sound Synthesis.

Sound synthesis is a method to produce a sound without the help of an acoustic instrument. It's commonly used in music and speech applications. In digital synthesis the sound is represented by a discrete time signal. The digital synthesis process computes this signal from a number of parameters that represent dimensions of the auditory domain, e.g. pitch (frequency) and loudness (amplitude). The parameters can be static or time variant, their data content is typically much lower then the represented sound signal. In this view sound synthesis is a data expansion method.

*Author to whom all correspondence should be addressed.

A sound synthesis program expands input data to a discrete time signal. The signal, a stream of numbers, is then converted into an analog signal. A high quality audio signal that reproduces the audio band from 0 to 20 KHz must have at least 40000 samples per second. This means that an average of 25 microseconds is available to calculate one sample. However, if it is not possible to produce the sample in this time interval the samples are produced at a rate that is too low, and audible clicks or distortions will result. To meet this time constraint either the sample rate must be decreased, resulting in loss of sound quality, or the computational power must be increased (i.e. adding more processors in a parallel environment).

DePoli [3] gives two classes of sound synthesis techniques : (1) *generation techniques*, which directly produce the signal from given parameters and (2) *transformation techniques* which modify previously generated signals (i.e. filtering). An example of (1) is *additive synthesis* . According to the Fourier theorem a number of simple sine signals are added to form a sound. Each sine signal has two parameters, frequency and amplitude. For a complex sound many sines are needed.*Subtractive synthesis* is an example of (2). An existing sound signal with preferably a rich spectrum is modified through the use of digital filters. Each filter adds a typical local maximum in the spectrum of the signal, called a formant. Speech signals have typically four to six formants [4]. The synthesis model that is used in this paper, VOSIM, belongs to class 1 although a class 2 description can also be found in [5].

Historically, VOSIM was implemented on special purpose analogue electronic circuits, followed by an implementation on a PDP15 and a special purpose digital electronic circuit [6]. These classic implementations are expensive and rather inflexible. Work on implementations on modern microprocessors show that VOSIM is a calculation intensive algorithm [7]. In many applications sound synthesis algorithms are implemented in a serial manner. Although Moorer [8] remarks (in 1977) that music is inherently a parallel process only recently results of research that studies parallel sound synthesis have been published [9] [10].

2.2 Sonification

The visualization of scientific data has as a drawback that only a limited number of dimensions is available. Beside the three spatial dimensions time (animation) and colour form the major ones. Using sound as well may improve the computer - human interface dramatically. Yeung discusses a method of sonification where, in sound alone, nine dimensions are available [11]. In his method pitch, loudness, damping (of a cosine signal), spatial direction, duration, repetition and rests are the auditory dimensions. Pollack and Ficks show that the information content of an auditory display increases with the amount of stimulus aspects that can be handled independently [12]. Several researchers report performance improvements with respect to the discrimination of datasets when sound is added to a visual display [13] [14].

Most systems use sonification complementary to visualization. For example Rabenhorst, Farrell et al. describe an interactive system that gives the user access to three 3D datasets [15]. Craig and Scaletti describe a sonification toolset where sound classes can be designed to fit as good as possible on the different dimensions of the dataset under sonification [16]. The VOSIM sound synthesis method is a strong candidate for sonification purposes because formal methods can be applied to obtain input parameters for speech [17] and for music [18].

3. A PARALLEL SOUND SYNTHESIS MODEL

When designing a synthesizer capable of generating at least 4 voices that is flexible and expandable we need a parallel system consisting of more than one processor to provide the necessary processing power over a single processor and the flexibility and expandability necessary to adapt the synthesizer to our ever increasing demands. To design such a parallel sound synthesis system, thorough knowledge of the mathematical structure of VOSIM is necessary. This leads to an abstract description of VOSIM. From this abstract description we can derive a model of parallel VOSIM describing the independent functions that may operate in parallel. This description of parallel VOSIM is used to analyse the time complexity of the algorithm, and to predict the performance of a parallel system executing the parallel VOSIM algorithm.

VOSIM is a sound synthesis model that produces sounds with a formant structure by repeating a sequence of $\sin^2$ periods and a delay (see Figure 1).

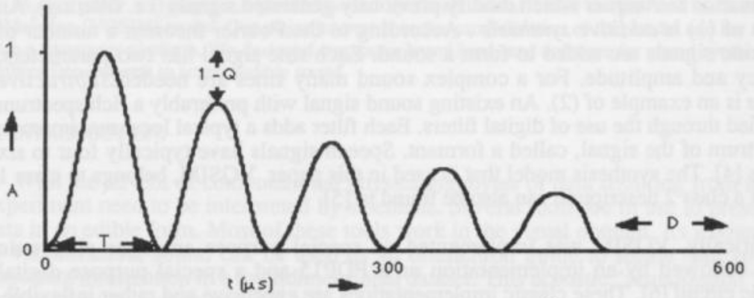
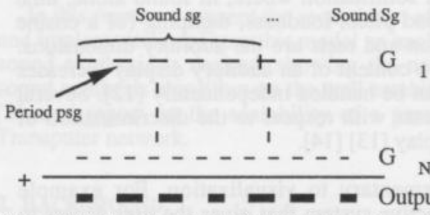


Figure 1. One period of a VOSIM sound. T is the duration of one $\sin^2$ period. N is the number of $\sin^2$ periods (in this case 5). A is the amplitude of the first $\sin^2$ period (A is shown in arbitrary units) D is the delay following the N $\sin^2$ periods, and Q is the Quench factor of the second and following $\sin^2$ periods.

A VOSIM generator produces output according to the parameters describing a sound. When a sound is finished, the next sound is processed.



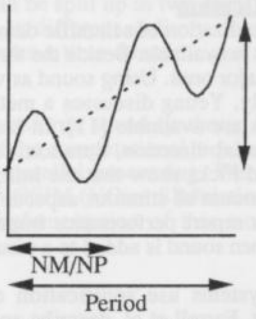


Figure 2. Adding the output of several generators results in synthetic music or speech. The generators produce sequences of sounds. These sounds are sequences of periods.

Figure 3 Sinus modulation.

To produce speech or music, the output of several generators has to be added in the time domain (see Figure 2). For instance to synthesize a male voice, the output of 5 generators needs to be added. In general we want to add a large number of generators:

$$\text{output}(t) = \sum_{g=1}^{N} G_g(t) \qquad (1),$$

$$G_g(t) = \bigcup_{s=1}^{S} \text{sound}_{sg}(t) \,,$$

where $\bigcup_{s=1}^{S}$ is the time ordered union of s = 1 to S. (2)

A sound is a sequence of periods.

$$\text{sound}_{sg}(t) = \bigcup_{p=1}^{NP} \phi_{psg}(t),$$

where $\bigcup_{s=1}^{S}$ is the time ordered union of p = 1 to NP (3)

A period consists of a sequence of sin^2 periods dampened by a staircase function and followed by a delay. The duration of a sin^2 period and the dampening is fixed within a period (see Figure 1). Within a sound, the duration of the sin^2 period, the delay and the amplitude may vary. These characteristics of a sound are described by a parameter set of 12 numbers. These parameters are:

T_s = The duration of the first sin^2 period of the first period of the sound s
dT_s = $NP_s(T_{ps} - T_{p+1s})$ = the increase of T_s over the sound s
D_s = Delay of the first period of the sound s
dD = $NP_s(D_{ps} - D_{p+1s})$ = the increase of D_s over the sound s(see Figure 3)
AM_s = Amplitude of Modulation of D_s
A_s = Amplitude of the first sin^2 period of the first period of sound s
dA_s = $NP_s(A_{ps} - A_{p+1s})$ = increase of A_s over the sound s
Q_s = Quench factor of the staircase shaped attenuation of the amplitude of the sin^2 periods in one period ($0 \le Q_s \le 1$)
N_s = Number of sin^2 periods in one period
MM_s = Modulation Method. 0 = sin modulation, 1 = random modulation.(see Figure 3)
NM_s = Number of Modulation periods in the sound s (see Figure 3)
NP_s = Number of Periods in sound s

The sound sg is parameterised by these 12 parameters:

$$\text{sound}_{sg}(t) = (t, T_{sg}, dT_{sg}, D_{sg}, dD_{sg}, AM_{sg}, A_{sg}, dA_{sg}, Q_{sg}, N_{sg}, MM_{sg}, NM_{sg}, NP_{sg}) \quad (4)$$

A period is parameterised by 5 parameters:

$$\phi_{psg}(t) = (t, T_{psg}, D_{psg}, A_{psg}, N_{sg}, Q_{sg}) \quad (5)$$

$$A_{psg} = A_{sg} + \frac{pdA_{sg}}{NP_{sg}} \quad (6) \qquad T_{psg} = T_{sg} + \frac{pdT_{sg}}{NP_{sg}} \quad (7)$$

$$D_{psg} = D_{sg} + \frac{pdD_{sg}}{NP_{sg}} + \mu_{sg}\left(AM_{sg}, MM_{sg}, NM_{sg}\right) \quad (8)$$

$$\mu_{sg} = \begin{cases} MM_{sg} = 0: & AM_{sg} \sin\left(\frac{2\pi p}{NP_{sg} NM_{sg}}\right) \\ MM_{sg} = 1: & U\left(-AM_{sg}, AM_{sg}\right) \end{cases} \quad (9)$$

U is the uniform distribution.

4. COMPUTATIONAL ASPECTS.

4.1 Functional Aspects.

From section 3 it can be seen that the structure of the VOSIM algorithm is partly self-similar (see e.g. Figure 2), operations on the signal level are reflected by operations on the generator level which are in turn reflected by operations on the sound and period level. method results in a typical data driven algorithm. Farming is a typical parallelization paradigm well suited for this type of data driven, self-similar algorithms [19]. From the description of VOSIM it seems that we have a set of basic tasks on three functional levels: (1) period level, (2) sound level and (3) generator level

Sounds, periods and generators are all described by independent parameter sets, although the possibility of random modulation of the delay allows for uncertainties in the duration of the sounds. This information concerning the history of previous sounds or periods must be passed on to the next sound or period, making it impossible to calculate successive sounds or periods at the same time. The generators however do have this history information and can therefore be calculated in parallel. This view of the algorithm leads to the following parallelization scheme: (see Figure 4).

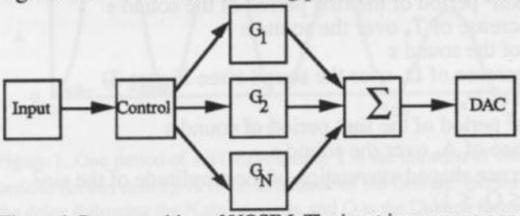


Figure 4. Decomposition of VOSIM. The input is a process or a file giving the parameters of all the generators. The control process distributes these parameters over the generators G_1 .. G_N. The output of the generators is then added and send to a DA convertor.

4.2 Implementation Aspects.

A program that implements a VOSIM generator performs two tasks. First, the samples that represent the desired sound must be created. For the generation of the $\sin^2$ periods we use a table lookup algorithm because this is the fastest and easiest to implement. Samples representing the delay have a value equal to zero. All the generators keep a copy of the $\sin^2$ table in their memory. Next, the parameters necessary to generate the $\sin^2$ periods (internal parameters as step size and phase) must be derived from the parameters describing the periods and sounds. To do this, the parameters describing the sounds are fetched from disk or an interface. The parameters for the periods are derived from the parameters describing the sounds according to the mathematical description of VOSIM. From the description of the periods the internal parameters are calculated, and finally the $\sin^2$ samples are generated.

The samples produced by the synthesizer must be offered to the DA convertor at a rate that is equal to the sample rate. To meet this demand the samples and the processes that produce them must be synchronised to the sample clock. In our hardware the DA convertor is synchronised by a clock. The handshake protocol of the transputer link is used to signal the sample generating processes when a new sample must be send to the DA convertor. The notion that samples must be available when the DA convertor needs them and that the link protocol takes care of the synchronization leads to the conclusion that if samples are produced at a rate that is equal to or higher than the sample rate, the samples arrive at the correct rate and that undistorted sounds are produced.

The samples produced by the generators are packed into packets to reduce the time involved in setting up the communication for the samples. Also, if buffers are used, full advantage is taken of the transputer's capability to handle communication and computation simultaneously. The generators are implemented as a two stage pipeline, where a packet of samples is generated parallel to the transmission of a packet to the farmer or another worker.

The farming paradigm can be implemented in two different ways. In the first implementation the workers implement one or more generators and the farmer takes care of the distribution of the parameter sets over the generator processes and the gathering and adding of the results. The result of the addition is then send to the DA convertor (see Figure 5).

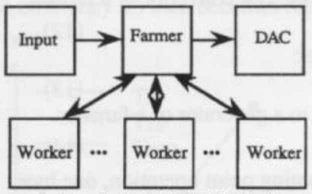


Figure 5. Farming of VOSIM. The farmer distributes the input over the workers, adds the results, and sends the output to the DA convertor.

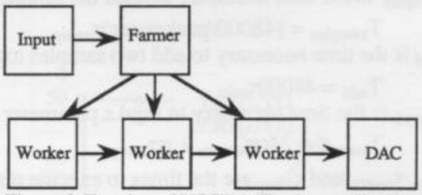


Figure 6. Farming of VOSIM. The farmer distributes the input over the workers. The workers add the results of their left neighbours, and send the results to their right neighbours. The rightmost worker sends the results to the DA convertor.

In the second possible implementation the output of the workers is send to the other workers in a pipeline fashion. The workers also add their output to the output send to them. The last worker sends the output to the DA convertor (see Figure 6).

The development of the VOSIM synthesizer was done using a Macintosh IIx with an internal TPM Mac Transputerboard (1 T800 + 2 Mbyte RAM). Connected to this Transputerboard are the Audio Sample Convertor (ASC) and a system based on a VME backplane with several T800 Transputers (see Figure 7).

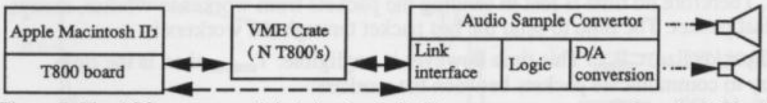


Figure 7. The ASC converts a digital signal supplied by a transputer to an analog signal. The ASC was developed at the electronics department of our faculty. [20] It consists of three parts, (1) a link interface, (2) logic to convert the byte wide link interface to the bit serial input of (3) the four times oversampling filter and digital to analog convertor (DAC). The ASC accepts 48000 32 bits words / sec. Each word consists of a 16 bit right- and left channel sample.

5. TIME COMPLEXITY.

Splitting a sequential algorithm into a number of parallel tasks generates overhead that stems from the calculations involved in the farmer (the division of the input data in tasks, and the distribution of these tasks over the slaves) and the communication between the farmer and the slaves. The VOSIM algorithm can be split up in the following operations:

1. Vector initialisation
2. Parameter transformation
3. Initialisation of internal parameters
4. Sin^2 sample generation
5. Delay generation
6. Adding sounds
7. Sending the sounds to the DA convertor

For the complexity analysis we need to find an expression for the relation between the expected number of calculations and the time necessary to perform these calculations. In Table 1 the number of calculations per operation is listed, yielding the following expression for the time necessary to calculate σ sounds and ρ periods on a sequential machine T_{seq}:

$$T_{seq}(N) = N(T_G + T_{add} + T_{sounds}) + T_{samples} \qquad (10)$$

where T_G is the time necessary to calculate σ sounds and ρ periods per generator:

$$T_G(\tau_{calc}, \tau_{comm}, \tau_{setup}) = ((22\rho + 1)\sigma + 264000)\tau_{calc} \qquad (11)$$

$T_{samples}$ is the time necessary to send the samples to the DA convertor or a farmer:

$$T_{samples} = (48000/packetsize)\tau_{setup} \qquad (12)$$

T_{add} is the time necessary to add two samples together:

$$T_{add} = 48000\tau_{calc} \qquad (13)$$

T_{sounds} is the time necessary to send s parameter sets to a generator or a farmer:

$$T_{sounds} = 24\sigma\tau_{comm} + \sigma\tau_{setup} \qquad (14)$$

τ_{calc}, τ_{comm} and τ_{setup} are the times to execute one floating point operation, one byte communication and one communication set-up. Decomposing the VOSIM algorithm according to the first farming scheme described in the previous section leads to the following expression for the execution times on a parallel machine T_{f1}:

$$T_{f1}(N,p) = \frac{NT_g}{p} + N(T_{sounds} + T_{add}) + (N+1)T_{samples} \qquad (15)$$

When the VOSIM algorithm is decomposed according to the second farming scheme the expression for the execution time is T_{f2}:

$$T_{f2}(N,p) = \frac{NT_g}{p} + NT_{sounds} + T_{output}(N,p) \qquad (16)$$

Since the worker processes are implemented as a two stage pipeline where a packet of samples is calculated in parallel with the communication of the previous packet, and the fact that during one sample calculation at least two samples can be send, we may assume that the communication of packets is completely hidden behind the generation of the packets. Therefore no time is lost in sending the packets from worker to worker, except for the last packet. The time to send the last packet through the workers is $p(48000/packetsize)\tau_{comm}$. This time however is negligible. T_{output} then is the time necessary to communicate packets between the workers:

$$T_{output}(N,p) = \left\lceil \frac{N}{p} \right\rceil T_{add} + p(48000/packetsize)\tau_{setup} \qquad (17)$$

Basic Operations	Number of FLOP's	Number of Bytes communicated	Number of times to initiate communication	Number of Basic Operations per second
1	1	24	1	σ
2	8			σρ
3	14			σρ
4	8			28000
5	2			20000
6	N	2		48000
7		2	1/packetsize	48000

Table 1. Amount of Floating Point Operations (FLOP's). For the basic operation mentioned above the number of FLOP's to perform these operations are listed, as well as the amount of communication involved.

In the analysis we assume a sample frequency of 48000 Hz and a $\sin^2$ to delay ratio of 1.4:1. The samples are created using a lookup table. σ and ρ are the mean number of sounds and periods per second.

We estimate the figures for τ_{calc}, τ_{comm} and τ_{setup} based on the figures for a Meiko machine used by the department [21] :

$$\tau_{calc} = 3.04\ \mu s,\ \tau_{comm} = 1.8\ \mu s,\ \tau_{setup} = 8.1\ \mu s \qquad (18)$$

The estimated execution time, speed-up and efficiency for the three different implementations are given in Figures 8 ,9 and 10. The values for σ, ρ and packetsize are fixed to respectively 10, 400 and 100. Figure 11 shows the expected speed-up per processor if N is kept constant. The speed-up S is defined as: T_{seq}/T_{fi} where $i \in \{1,2\}$. The efficiency (ε) then becomes S/p.

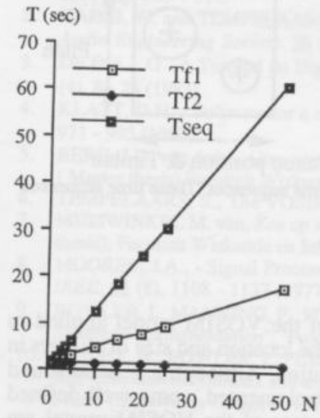


Figure 8. Execution times of sequential implementation and the two parallel implementation. N = p.

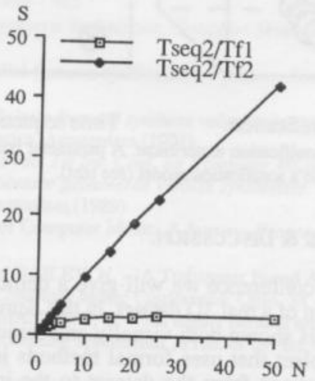


Figure 9. Speed-up of the parallel implementations over the sequential implementation. N = p.

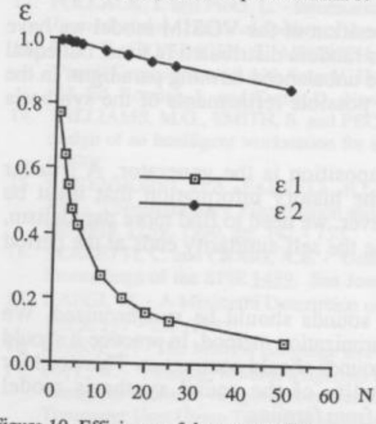


Figure 10. Efficiency of the two parallel implementations. N = p.

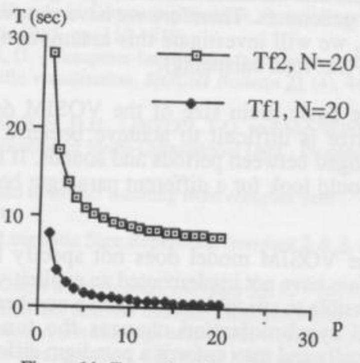


Figure 11. Execution time of the two farming implementations. N = 20.

6. SONIFICATION MODEL.

Data from an experiment [22] can be represented by a three parameter box (3 x 64 channels). The data is assumed to contain several clusters. By slicing the box in planes we get 64 elements that can be sonified in a time sequence. A plane can contain local maxima. Each local maximum is sonified with a combination of two sounds. Pitch (frequency on a log scale) is used to indicate the vertical position, horizontal position is mapped to the formant frequencies and the stereo positions of these two sounds. One formant scales from 350 to 880 Hz. and the other scales from 4400 down to 900 Hz. (see Figure 12). This gives the timbre a vowel like quality. The size of the clusters is mapped to the amplitude. All transformations are on a log scale to fit to the human perception.

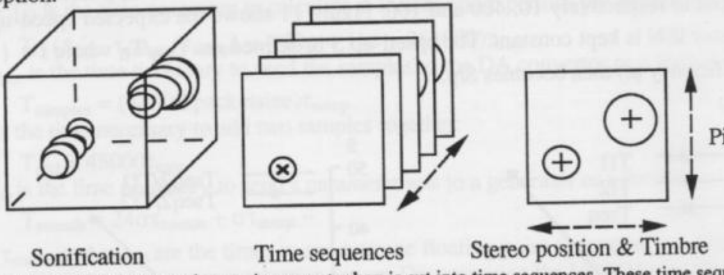


Figure 12. Sonification experiment. A parameter box is cut into time sequences. These time sequences are mapped to a sonification model (see text).

7. RESULTS & DISCUSSION.

At the conference we will give a demonstration of the VOSIM model applied to sonification of a real 3D dataset. In this demonstration the location and size of clusters in 3D space is shown both visually and through sonification. Although a general sound design toolset that uses formal methods is not yet implemented, some well defined transformations from the dataset to the input parameters of the VOSIM model are possible.

In the time complexity analysis and decomposition of the VOSIM model we have assumed that the calculation load per generator is random distributed in time, but equal for all generators. Therefore we have choosen the unbalanced farming paradigm. In the future, we will investigate this assumption and possible refinements of the synthesis model (i.e. load balancing).

The basic grain size of the VOSIM decomposition is the generator. A smaller grainsize is difficult to achieve because of the history information that must be exchanged between periods and sounds. If however, we need to find more parallelism, we should look for a different paradigm because the self-similarity ends at the period level.

The VOSIM model does not specify how sounds should be synchronised. We therefore have not implemented an explicit synchronization method. In practice it should be possible to specify a time when a number of sounds should commence. This need for global synchronization changes the functionality of the sound synthesis model drastically and may enforce a paradigm different from farming.

Finally, the wish to develop a sound synthesis system that is capable of producing samples in real time, raises questions about delay and real time characteristics. This also calls for a refinement of our model.

8. ACKNOWLEDGEMENTS.

We gratefully acknowledge the assistance of Jos van den Berg in the technical development of the VOSIM system and Pierre van Berkel for introducing us to the VOSIM model.

REFERENCES.

1. SPEETH, S.D. - Seismometer Sounds, *Journal of the Acoustical Society of America*, 33 (7), 1961, pp. 909 - 916.
2. KAEGI, W. and TEMPELAARS, S., - Vosim - A New Sound Synthesis System., *Journal of the Audio Engineering Society*, 26 (6), 1978, pp 418 - 425
3. DE POLI, G - A Tutorial on Digital Sound Synthesis Techniques, *Computer Music Journal* 7, (4), 8 - 26 (1983)
4. KLATT, D.H. - Software for a cascade / parallel formant synthesizer, *J. Acoust. Soc. Am.*, 67, (3) 971 - 995 (1980).
5. BERG, J.D. van den, - *Alias vervorming bij digitale formant synthese volgens de vosim methode*. (Master thesis) Faculteit Wiskunde en Informatica, Amsterdam (1991).
6. TEMPELAARS, S., The VOSIM Oscillator. *Proc. 1st Int. Conf. on Computer Music*. MIT 1976.
7. MUISWINKEL M. van, *Een op een signaalprocessor gebaseerde VOSIM synthesizer* (Master thesis), Faculteit Wiskunde en Informatica, Amsterdam (1989)
8. MOORER, J.A., - Signal Processing Aspects of Computer Music: A Survey, *Proceedings of the IEEE*, 65 (8), 1108 - 1137 (1977).
9. BOWLER, I., MANNING, P., PURVIS, A. and BAILEY, N. - 'A Transputer Based Additive Synthesis Implementation', Proc. of the Int. Computer Music Conference, Ohio, 58 - 61 (1989).
10. BAILEY, N., BOWLER, I., PURVIS, A. and MANNING, P. - 'An highly parallel architecture for real -time music synthesis and digital signal processing application', Proc. of the Int. Computer Music Conference, Glasgow, 169 - 171, (1990).
11. YEUNG, E.S. - Pattern Recognition by Audio Representation of Multivariate Analytical Data, *Anal. Chem.* 52, 1120 - 1123 (1980).
12. POLLACK, I. and Ficks, L. - Information of Elementary Multidimensional Auditory Displays, *Journal of the Acoustical Society of America* 26 (2), 155 - 158 (1954).
13. BLY, S., FRYSINGER, S., MEZRICH, J., MANSUR, D., MORRISON and LUNNEY, D. - 'Panel: Communicating with Sound', Human Factors in Computing Systems, Proceedings of CHI '85, Ed. Borman, L. and Curtis, D., New York, 1985, pp. 115 - 119.
14. WILLIAMS, M.G., SMITH, S. and PECELLI, G. - Computer-human interface issues in the design of an intelligent workstation for scientific visualisation, *SIGCHI Bulletin* 21 (4), 44 - 49 (1990).
15. RABENHORST, D.A., FARRELL, E.J., JAMESON, D.H., LINTON, T.D. and MANDELMAN, J.A. - 'Complementary Visualization and Sonification of Multi-Dimensional Data', Proceedings of the SPIE 1259, 1990, pp. 147 - 153.
16. SCALETTI, C. and CRAIG, A.B. - 'Using sound to extract meaning from complex data', Proceedings of the SPIE 1459, San Jose,1991.
17. KAEGI, W. - A Minimum Description of the Linguistic Sign Repertoire, *Interface* 2 & 3, 141 - 156 & 137 - 158 (1973) .
18. KAEGI, W. - The MIDIM Language and Its VOSIM Interpretation, *Interface* 15, 83 - 161 (1986).
19. DAY, W. - 'Farming: towards a rigorous definition and efficient transputer implementation.', Transputer Systems - Ongoing Research, Ed. Allen, A.,Proceedings of the 15th World Occam and Transputer User Group Technical Meeting, Aberdeen, Scotland, UK, April 1992, pp. 49 - 62.
20. BERG, J.D. van den, - *De Audio Sample Convertor voor Transputersystemen*. (Internal rapport) Elektronicagroep,Faculteit Wiskunde en Informatica, Amsterdam (1990).
21. HOEKSTRA, A.G., SLOOT, P.M.A., HAAN, M.J. de and HERTZBERGER, L.O. - 'Time complexity analysis for distributed memory computers', Computing Science in the Netherlands, Proceedings 1, Ed. Leeuwen, J. van ,1991, pp. 249 - 266.
22. SLOOT, P.M.A., VAN DER DONK, E.H.M. and FIGDOR, C.G. - Computer Assisted Centrifugal Elutriation Part II: Multiparametric Statistical Analysis, *Comp. Meth. Prog. Biomed.*, 27, 37 (1987).