THE DESIGN AND IMPLEMENTATION OF A REAL-TIME SOUND GENERATION SYSTEM*

Paul Dworak
Departments of Music and
Electrical Engineering

Alice C. Parker and Richard Blum
Department of Electrical Engineering
Carnegie-Mellon University
Pittsburgh, PA 15213

ABSTRACT

This paper describes the design and partial implementation of a real-time sound generation system currently being built at Carnegie-Mellon University. The overall design process is followed, and design decisions are discussed in the light of design goals and overall system constraints. The resulting system consists of a keyboard which is light controlled and an array of microprocessors functioning in a pipeline fashion.

I. INTRODUCTION

This paper describes the progress of a single group of researchers in designing, implementing, testing and using a large scale digital system, the Carnegie-Mellon Computer Music System. While observing the progress of such a design, possibilities for monitoring and carefully controlling the design process arise. It becomes apparent that the ideas of structured hardware (LAW 75, TAN 76), differing design styles (THO 76), and design constraints (BAR 76) can be applied to a particular design during both design and analysis phases.

This particular system discussed in this paper is a real-time digital sound generation system. This paper discusses the design and implementation of the above system and attempts to answer the following questions:

Is a structured design approach valid for this system? (When) does it fail?

Can one pick a design style for this system *a priori* by analyzing the design constraints? Is this design style indeed reflected in the implemented system?

How does the level of design constraints affect the structured design approach? Do low-level constraints negate a top-down design approach? Do high-level constraints make a bottom-up approach impossible?

How can one express system parameters on all levels with a single system description?

The following sections of this paper describe the keyboard and overall system design. The last section discusses the above questions, gives the current status of the system, and presents some preliminary conclusions.

The historical background for this and other synthetic sound research can be found in (DWO 75). A system which is functionally similar but architecturally quite different has been built at Aarhus University, Denmark, by Manthey (MAN 75).

* The research described in this paper is supported by the Departments of Music and Electrical Engineering at Carnegie-Mellon University and a grant from the Carnegie Corporation.

II. THE KEYBOARD DESIGN

High-speed interactive digital systems require man-machine interfaces that quickly generate data in a format that is compatible with both the speed of the digital processors employed and with the demands of the algorithms upon which the system operates.

These demands call for a very high-speed system with an input device that controls frequency, amplitude, envelope, and timbre in an efficient way. Since the traditional keyboard is able to represent some of these parameters, and since it is familiar to most musicians, the researchers decided to redesign the keyboard so that it might control all parameters. The speed needed within the system is achieved by using optical data generation and transmission techniques, and by employing Schottky bipolar and ECL logic devices and microprocessors in the sound generation circuitry. These techniques also make possible a low-cost system that provides a maximum amount of control with a minimal investment.

The present keyboard has sixty-four keys, each of which is assigned a digital address (0 to 111 111). If a scan of the keyboard reveals that a particular key is depressed, only the address of that key is read by the keyboard-monitor computer. The frequency with which that address is associated is stored in memory and must be preprogrammed. The user has complete freedom to map pitches onto the keyboard in the way that best suits his needs.

At the present time, the artistic appearance of the keyboard is not considered important. It is assumed that the composer or researcher will know the frequencies that he wishes to represent. No color codes or layout schemes are employed to imply a particular frequency sequence. The keys are simply set next to one another in a slightly semi-circular fashion, so that any finger may lie flat on a key, parallel to its edges. The amplitude of a tone associated with a key is defined as increasing in a logarithmic fashion as the depth to which the key is depressed increases. The speed with which the key is depressed does not define amplitude. However, simple envelopes can be constructed by changing the depth of the key over time. Very fast rise or fall times cannot be created in this way because of the limitations of the hand. However, such envelopes can be created under external program control.

By dividing each key into eight segments, tone color or timbre changes are made possible. Each segment is defined as some frequency multiple of (usually) the lowest segment. (See Figure 1.1).

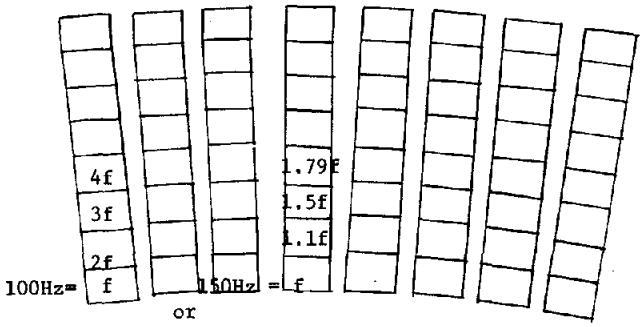


Figure 1.1
Keyboard Layout with Possible
Frequency Relationships

In other words, if the system has a sine wave stored in memory, each segment of this key may reference that waveform. The lowest segment might sound at a fundamental frequency (e.g., 440 Hz.) and the overtones might be in harmonic ratios (e.g., 880 Hz., 1760 Hz., etc.). The waveform generated by any key is synthesized by combining the waveforms referenced by each of that key's segments. Of course, since all frequencies represented on the keyboard are programmable, the segments just as easily can represent either undertones or non-harmonic multiples of the fundamental.

The system thus adds simple waveforms to generate complex waveforms. However, true flexibility in waveform construction demands that the keyboard ought also to be able to control the amplitude and phase of each particular overtone. Since the depth of a segment represents its amplitude independently of its neighbors, it is indeed possible to do this. Phase is further defined as the angle of each key segment at any given depth. Of course, there are some physical limitations that govern which segment combinations can be pressed down by a human finger. Segments 1, 3, and 5 cannot be depressed without depressing segments 2 and 4.

Of course, the key segments need not reference only sine waves. Any waveform can be stored in memory and these waveforms will likewise be added by the keyboard. In this way, both the synthesis of very complex waveforms and control of their degree of complexity become very simple. Human limitations also make impossible the use of different amplitude patterns on adjacent segments or the constant use of perfectly identical amplitude envelopes. As a result, it is proposed that an external minicomputer provide data generated under software control for any of the following keyboard parameters: envelope, maximum amplitude, phase relationships among segments, constant patterns of harmonic relations among key segments, i.e., constant timbre. Data supplied by an external minicomputer would cause the system to ignore the same data coming from the keyboard. The user would determine how much external control is provided at any time. These decisions would depend upon his task.

Since the system constructs waveforms by adding model waveforms that are represented by the key segments, a minimum system requirement is that all segments on one key be processed within one 25 μsec. sample period (the chosen sample period for this system) so that the phase relationships of the component waveforms might be correct as defined on the keyboard by the user.

There are eight amplitude words and eight phase words that must be read from each key. This allows a little more than 1 μsec. to do each READ. This is perfectly feasible for a bipolar or ECL microprocessor, but the interface must generate this data quickly. Furthermore, since these amplitude and phase signals must vary significantly in size, it is difficult to digitize them directly, quickly, efficiently, and with a good amount of resolution. Changes of amplitude can be represented by analog signals, but it would be inefficient to employ separate transducers to generate control voltages on each key segment.

The system will employ a very high-speed and cost efficient method of generating digital words representing amplitude and phase for each key segment. Each key segment contains two light sources, one to control amplitude and one to control phase. The key body itself is constructed so that, as the key is depressed or tilted, more light is passed to an optical fiber (See Figure 1.2).

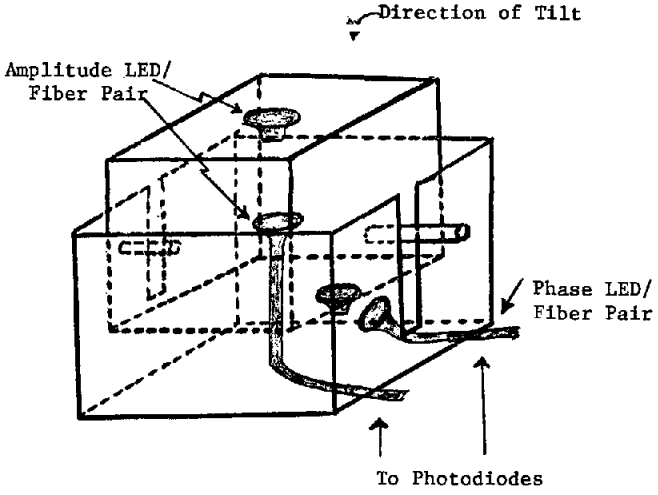


Figure 1.2 A Key Segment
Springs and Internal Mounting
Hardware Omitted

Light is transmitted down the fiber to a bank of sixteen light sensors, eight for amplitude control, eight for phase control. The output of each sensor is amplified to generate an analog voltage that can be varied either logarithmically (for amplitude control) or linearly (for phase control).

These voltages are successively converted to eight-bit digital words by an analog-to-digital converter. As a result, there are 256 logarithmic amplitude steps possible for the construction of envelopes. Also, 256 phase offsets are available for finding the first word to be sampled from waveform memory.

The phase word can serve two purposes. It can control the offset of the starting points of waveforms in relation to one another. In this mode, each phase word must be compared against its last value and must be incremented or decremented according to that last value. Alternately, the phase word can be interpreted as a new increment each time, in which case slow frequency modulation on the keyboard is possible.

The use of light control permits high-speed operation. Set-up time for all sixteen light sensors is intended to be less than 2 μsec. The analog-to-digital conversion time for each sensor is slightly more than 1 μsec. As a result, all signals on one key group can be processed in less than 25 μsec.

The use of an optical system for digital data generation and transmission serves the additional purpose of multiplexing the keys to the central processor in a fast and simple manner. Fiber bundles that originate as single fibers in the key segments and that terminate at the photosensors provide an OR function of events that occur on parallel key segments.

Multiplexing operations actually occur on several levels within the keyboard interface. As the keyboard is being scanned, the next key to be scanned is activated by a 1-of-64 decoder. Information on the key is then multiplexed to photosensors via fiber optics. The analog voltage generated by the photosensors is then multiplexed to a single 1 μsec analog to-digital converter.

III. OVERALL SYSTEM DESIGN

In discussing the evolution of the sound generation system from the beginning recognition of system constraints and choice of design goals to the end realization of the register-transfer level implementation, it becomes apparent that the design choices at all stages are influenced by both "hard" constraints (necessary system parameters) and "soft" constraints (desirable system parameters).

The overall design process is shown in the flowchart, Fig. 3.1. Each of these blocks will now be discussed in detail.

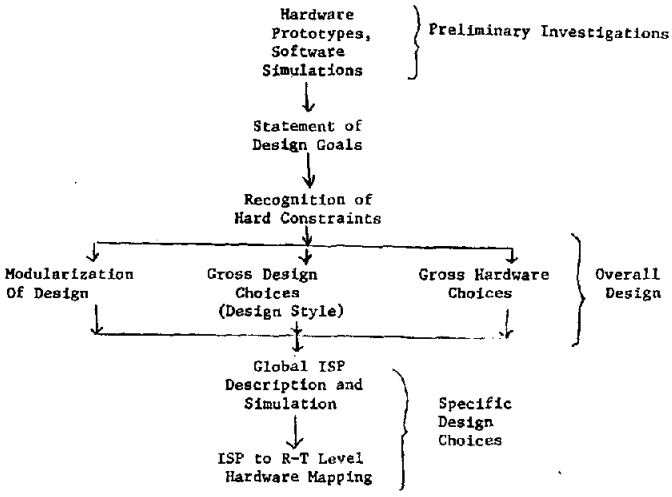


Figure 3.1 Overall Design Process of the Sound Generation System

Preliminary Investigation

Preliminary investigations included hardware prototypes of the keyboard interface system and output circuitry and software simulations of the central processor's functions. The keyboard prototype was constructed, tested and reported (DWO 75). A small system was constructed to output digital data from a memory to a D/A converter so that a preliminary analog tape could be made, and waveforms were generated using software on the UNIVAC 1108. (DWO 76). The effect these investigations had on the overall design was subjective -- they illustrated both the feasibility and desirability of such a system, and also basic problems in timbre recognition, variable timbre being the most important function of the system.

Design Goals

The major aim of the sound system design is to provide a user with tools for generating in real time, a sound with any deterministic or stochastic properties. Particular applications of the instrument are:

music composition

music theory research

microtonal interval student training

psychoacoustic studies

hearing testing and similar biomedical applications

Secondary goals derived from these applications are:

to control waveform structure by controlling, in real time, the amplitude and phase of harmonics added to a fundamental frequency

to allow user mapping of frequencies onto the keyboard interactively

to allow a large frequency range (40 Hz to 20 KHz)

to allow user control over envelope in real time

to allow program control over any parameter for precise waveform generation

to produce a high quality signal

to minimize system cost

These goals introduce some hard constraints on the system design.

Hard Constraints

The constraints introduced by the above design goals fall into the following areas:

Control

Data I/O

Data Storage

Data Manipulation

Control constraints mainly involve the necessity of varying waveshape and keyboard frequency mapping under user control. This indicates that the control must be programmable or that memories must be

loaded with the correct waveform and keyboard frequency data generated outside the sound generation system. Since the overall system operation is special purpose and fixed, the control can be established as the system is implemented - either with PLA's, flip-flops or a read only microstore.

Input/Output constraints concern the keyboard scanning and the output of the digitized waveform to the D/A converter. The system must receive from the keyboard the following information:

location of depressed keys

amplitude and phase represented by depressed keys

Furthermore, it must be able to receive this information quickly enough so that there is not an appreciable time delay between key depression and waveform processing. However, this constraint is easily met. There are 512 key segments on the keyboard. Thus if the complete keyboard information is input every 12 ms (small delay), information from a single key must be input about every 25 μs.

The speed with which data can be output determines the maximum frequency of the sound which the system can produce. According to the sampling theorem, waveforms with frequency of 20 KHz can be output if the systems can output one sample every 25 μs. This puts design constraints on the D/A converter as well as on the digital system.

Since the waveforms can be general, it would be quite difficult to generate them in real time. As a result, a memory is needed for waveform storage. However, in order to minimize memory and hence, system cost, one waveform is used for the entire keyboard and is sampled at different intervals to generate the various frequencies and harmonics represented by the keyboard. The dimensions of this memory required to produce a high quality waveform will be discussed later.

The data manipulation constraints involve accessing the correct waveform value from the stored waveform, multiplying the waveform to generate appropriate amplitude and envelope characteristics and adding simultaneously sounding waveforms together. This implies that the systems must have both indirect and indexed addressing capabilities to access waveform values, and arithmetic capabilities to perform the multiplications and additions. Furthermore, because the number of waveforms must be multiplied and added together every 25 μs, the multiplication is hardwired. With these constraints in mind, we proceed with the overall design discussion.

Overall Design

The choice of an overall design style, the mapping of system operation onto system architecture (modulation of design) and the choice of global implementation features are presented here.

The sound generation system has the following characteristics:

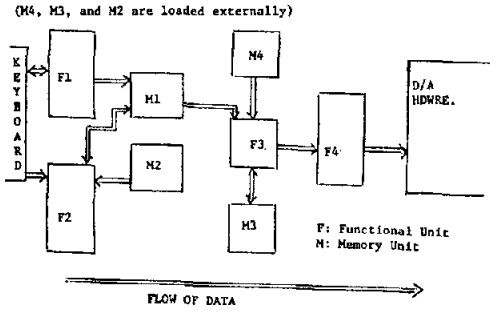
speed of data throughput is very important

data throughput and arithmetic calculations are the gross functional features of the system

the system functions are data independent

These characteristics point to a pipeline architecture for the data/memory portion of the system. This choice of design style should be confirmed by application of specific algorithms developed by Thomas (THO 76) to the ISP description at a later date.

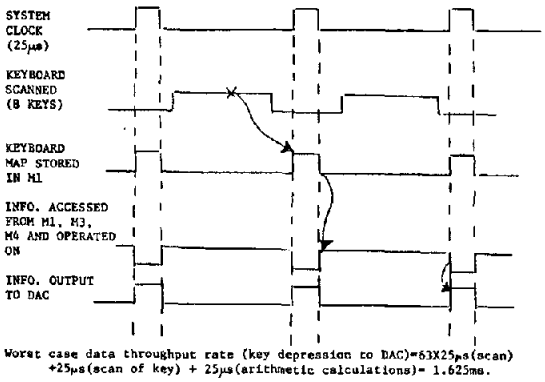
Proceeding in a structured manner, functions were mapped onto blocks in the system, and the blocks ordered in a time sequence from left (data input) to right (data output). Following the same procedure, each data type was mapped onto a memory block. The overall functional and memory block diagram is shown in Fig. 3.2.



F1: Inputs keyboard information and stores it in M1
F2: Gets sample increment information for each key and stores it in M1
F3: Gets keyboard information from M1, waveform information from M3, and envelope information (optional) from M4
F4: Performs the arithmetic functions on the information and outputs the information to the D/A hardware

Figure 3.2 The Overall Functional and Memory Block Diagram of The Sound Generation System

The primary question at this level is how the functional modules perform memory accesses without contention arising, and still optimize the data throughput rate. The simplest answer to this question is synchronous system operation. An overall 25 μs system clock is used, with some functions being performed with clock high, and some with clock low. The system timing is shown in Fig. 3.3. The arrows indicate the path of a single key depression through the system



Worst case data throughput rate (key depression to DAC)=63X25μs(scan) +25μs(scan of key) + 25μs(arithmetic calculations)= 1.625ms.

Figure 3.3 Overall System Timing for the Sound Generation System

Digital System Description

In order to describe the system compactly in some detail, an ISP (BEL 70) description of the system is presented. Writing this description unearths implementation issues on the register-transfer level. The comments given with the description highlight these issues. This description is given in Fig. 3.4.

The remaining issue to be faced is the problem of waveform quality.

```
ISP

F1: = (Keyboard Addrs ← Keyboard Addrs + 1 Next                ! Process Next Key
      (Amplitude ← IN1,                                        ! Amplitude and Phase
      Phase ← IN2) Next                                        ! Come from key
      (If amplitude = 0, then F1) Next                         ! If key not depressed,
      Wait Clock =1                                            ! the next key
      Old amplitude ← M1 [Keyboard Addrs] Next                 ! Memory has old amplitude
      M1 [Keyboard Addrs] ← Amplitude Next                     ! Put in new amplitude
      Keyboard Addrs ← Keyboard Addrs + 1 Next
      (If old amplitude = 0 then New key) Next                 ! Put in new phase
      F1)                                                      ! Process next key

New key: = (If queue [left] = 0 Then (queue [left] ← Keyboard Addrs-1 Next
                                      F1)) Next
                                                               ! Find empty place
          queue ← queue ↑ 1 Next                               ! in queue for key
          New key)                                             ! address

If Not Clock Then
(F2: = (Addrs ← [QUEUE[Right]]_Next                            ! Take address out of queue
      Amplitude ← M1 [Addrs] Next                              ! Find amplitude.  If key
      (If amplitude = 0 then erase) Next                       ! is not depressed remove
      Phase ← M1 [Addrs + 1] Next                              ! from queue
      Freq ← M1 [Addrs + 2] Next                               ! Add frequency increment
      Current ← M1 [Addrs + 3] Next                            ! to current sample
      Current ← Current + Freq Next
      M1 [Addrs + 3] ← Current Next                            ! New current sample
      Current ← Current + Phase Next                           ! Display by phase
      Out ← Sine [Current] Next                                ! Get sample from sine tab
      F4IN1 ← Out   Next                                       ! Pass sample and amplitud
      F4IN2 ← Amp   Next F4FLG  1 Next                         ! to F4
      F2))

Erase: = (Queue[Right] ← 0 Next                                ! Clear this queue positio
         Queue ← Queue  1 Next                                 ! Rotate left
         F2)

F4: = ((If clock then                                          ! If clock flag then
                  (DAC ← SUM Next                              ! output sum to DAC
                   SUM ← 0   Next                              ! clear sum
                                                               ! flag
                   F4)
      );
      (If F4FLG and Not Clock                                  ! If flag from F4
                  (Out ← F4IN1 ↑ F4IN2  Next                   ! Amplitude*Sample
                   Sum ← Sum + Out    Next                     ! Sum samples
                   F4FLG  0 Next F4
      ))
```

Fig. 3.4 Partial ISP Language Description of the Sound Generation System

Signal Processing Considerations

The system utilizes a sampling technique to create the complex waveforms desired, which implies that a model waveform exists in memory and that the system indexes through the memory choosing samples at various increments depending upon the frequency desired to be played. Although any waveform will be able to serve as the model, one which is easy to analyze for distortion is the sine wave and we will base our analysis upon it.

There are certain limitations which must be taken into account before constructing the system. A primary consideration is the size of the waveform memory; how many bits should represent each sample of the wave, and how many samples are necessary to create a high fidelity sound. These are limited by the bit size of the D/A converters and the speed of the processor. High fidelity or good resolution can be depicted graphically as a smooth curve or as a high signal to noise ratio .

In addition, as more waves get added together, more bits are required to express the summation, allowing as many harmonics and voices as possible, but not overflowing the sum. In this system, the sum will get passed to a D/A converter which will be 16 bits wide. The human ear perceives loudness changes as logarithmic changes of amplitude. A shift left or multiply by two is a change of 6 dB. (6= 10 log (2)). Therefore, we can think of each bit position as 6 db, and a range of 60 db will require 10 bits worth of information. Eight simultaneous waveforms require 3 additional bits, and each additional bit will double the number of voices.

The analysis so far has given 6 db/bit and the number of bits for the total number of voices = n(# voices) + 3. With a 16-bit D/A converter it is obvious there are going to be some tradeoffs necessary in choosing the size of the memory to contain the sampling waveform. This waveform will represent full amplitude. Lower amplitudes will be obtained by dividing samples, instead of by starting with a waveform of low amplitude and multiplying to achieve higher volumes. It appears better to start with a high resolution for the waveform and maintain that resolution through division rather than starting with a lower resolution and multiplying.

In Fig. 3.5 there are waveforms consisting of 1024 words and different numbers of bits. It is seen that when 6 bits are used the steps between waveform points are very noticeable, and when 10 bits are used, a very good approximation to a smooth curve is obtained.

Better than a graphical look at resolution is a Fourier analysis of the sampling waveform to determine signal to noise ratio. The Fourier analysis revealed that power gets distributed periodically through the spectrum. More important is the fact that there is a linear relationship between the number of bits used and the signal to noise ratio. A graph can be seen in Fig. 3.6. The signal to noise ratio is approximately equal to 6 x (No. of bits). (The lowest amplitude will have the lowest ratio). Therefore, choosing a signal to noise ratio and amplitude range chooses the number of bits that should make up the sample waveform. The FFT analysis also revealed that the number of samples in a wave did not affect the signal to noise ratio, which is expected from the sampling theorem. Therefore, the criteria in choosing the length of memory is at what lowest frequency one wishes to maintain waves that have no repetition of samples.

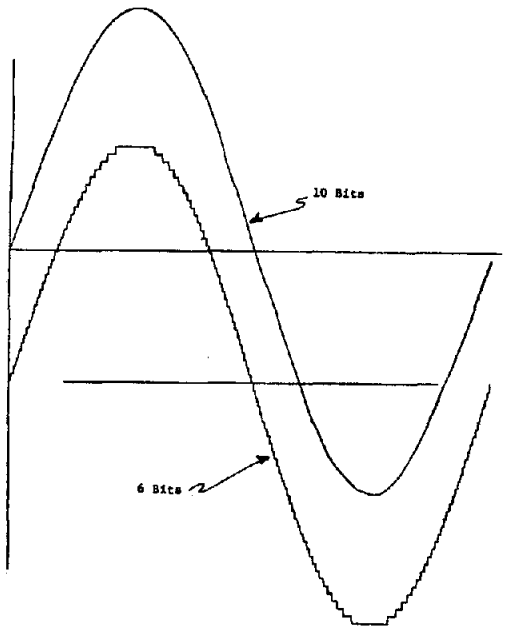


Figure 3.5 1024 Word Sine Waves With 6 and 10 Bit Resolution

$$SNR = 10 \log \frac{\text{Fundamental}}{\text{Noise}}$$

Signal to Noise Ratio in db

# WORDS \ # BITS	16	14	12	10	8	7	6
1024	9.8	8.5	7.4	6.2	4.9	4.3	3.8
128	9.8		7.4	6.1			3.8
32	10.1		7.4	6.0			3.9
16	10.1		7.2	5.9			

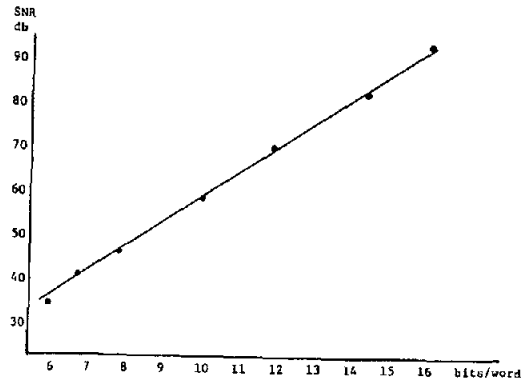


Fig. 3.6 Fourier Analysis of Waveforms

The processor in this system is an Intel 3000. With a cycle time of 150 ns, approximately 16 harmonics can be processed. Allowing 4 bits of information for voice addition, 12 bits remain for resolution. Allowing a 48 db range requires another 8 bits, leaving 4 bits. If 4 bits is the lowest amplitude to be represented, the system has a signal to noise ratio of 24 db.

IV. PRESENT STATUS, DISCUSSION AND PRELIMINARY CONCLUSIONS

At present, the ISP description of the system is being completed and will be used for simulation of the overall system operation. Hardware development at present is focused on the keyboard, keyboard logic and output circuitry. Most of the major integrated circuitry for the system has been specified (including bipolar memories and microprocessors). In fact, each functional unit, except for the adder and multiplier, is to be realized with a bipolar microprocessor, the Intel 3000 series.

Future efforts, once the system is built, include an interactive program running on the PDP-11 and a link to the memories of the sound system from the PDP-11.

The questions posed in the introduction will now be discussed. First, the structured approach appears to produce a design in this case which is feasible, but probably uses substantially more hardware than the brute force approach. In answer to the second question, picking a global design style a priori seemed to involve only the global design, with the architecture of each functional unit and memory a decision to be made later on in the design process, and which could vary radically from the pipeline style chosen for the overall system. The third question, which considered the effects of multi-level constraints on each design level, was answered for the top-down approach. For this system, as long as the design constraints were considered at each level of the design process, it did not matter if the level of the design decisions differed from the level of the design constraints. The last question, the problem of expressing system functioning on all levels with a single description was not answered here. The ISP description for this system is satisfactory because the system has synchronous output and synchronous functional units and a scanned input. Lower level considerations such as timing and I/O synchronization therefore did not have to be considered for this system with ISP.

In conclusion, the system is expected to be one of the fastest real time digital sound generation system which has been constructed to solve the general problem of sound synthesis. In addition, the system cost is expected to be relatively low due to the use of LSI components instead of a minicomputer or large scale computer.

REFERENCES

1. (BAR 76) Barbacci, M. and D. Siewiorek, "The CMU RT-CAS System: An Innovative Approach to Computer Aided Design", NCC, N.Y. , May 1976.

2. (BEL 70) Bell, G. and A. Newell, Computer Structures, McGraw-Hill, 1970.

3. (DWO 75) Dworak, P. and A. Parker, "An Input Interface for a Real-Time Digital Sound Generation System", Proceedings 3rd Annual Symposium on Computer Architecture, Clearwater, Florida, January, 1976.

4. (LAW 74) Lawson, H. and B. Magnhagen, "Advantages of Structured Hardware", Proceedings, 2nd Annual Symposium on Computer Architecture, Houston, Texas, January 1975.

5. (MAN 76) Manthey, M., A Non-technical Description of the EGG Real-Time Sound Synthesizer, Internal Document, Department of Computer Science, University of Aarhus, AArhus, Denmark,March, 1976.

6. (TAN 76) Tanenbaum, A., Structured Computer Organization, Prentice-Hall, 1976.

7. (THO 76) Thomas, D., Internal Document, Department of Electrical Engineering, Carnegie-Mellon University, 1976.

8. (DWO 76) Dworak, P. E., The Utility and Compositional Ramifications of Grouping Theory, PhD Dissertation, Department of Music, Carnegie-Mellon University, 1976.