THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 60, No. 7, September 1981
Printed in U.S.A.

Digital Signal Processor:

Speech Synthesis

By M. R. BURIC, J. KOHUT, and J. P. OLIVE

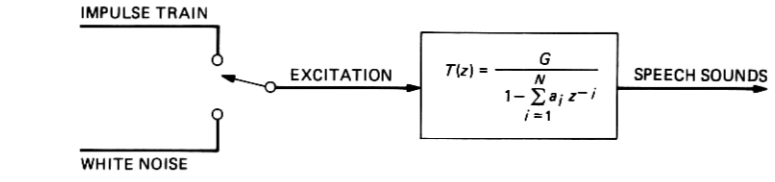
(Manuscript received June 10, 1980)

This paper describes a device that is capable of synthesizing speech in real time and is based on the digital signal processor chip. The device performs a function of a twelfth-order linear prediction coding synthesizer, and as such represents a linear dynamic system approximation of the vocal tract. In this model, short time segments of the speech waveform are derived as output of a system driven by a pseudo-periodic impulse sequence for voiced sounds, or by white noise for unvoiced sounds. The time-varying nature of the system is derived from the input information presented to the device for every new pitch period. Interfacing of the device to standard microprocessors is easy, so that the synthesizer can conveniently be integrated into larger systems.

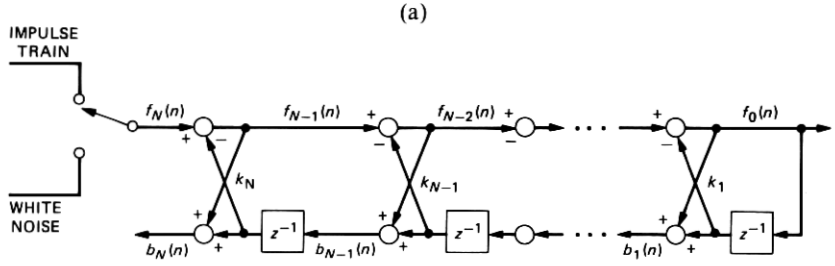
I. INTRODUCTION

Speech synthesis is one of several promising areas of application for the digital signal processor (DSP) chip described in this issue of the *Bell System Technical Journal*. Its computational power, low cost, and easy interfacing are the properties that allow a design of a stand-alone speech synthesizer with very few components outside the DSP chip. Such a synthesizer can be used in a variety of devices intended for providing new services in a business environment, as well as in future residential services.

One of the most successful ways to synthesize speech is based on a linear predictive coding (LPC) model of a vocal tract. See Refs. 1, 2, and 3. In this model, the vocal tract is approximated by a linear dynamic system driven by impulse sequences for voiced sounds, or by white noise for unvoiced sounds (Fig. 1a). The difference equation for such a representation is



(a)



(b)

Fig. 1a—Polynomial form of transfer function.
Fig. 1b—Lattice form of transfer function.

$$s(n) = \sum_{i=1}^{i=N} a_i \, s(n - i) + Gu(n). \tag{1}$$

In this discrete description of a linear system, $s(n)$ is the signal at time instant n, $\{a_i;\ i = 1, 2, \cdots, N\}$ is the set of linear prediction coefficients, $u(n)$ is the system input, and G is the gain coefficient.

A pitch synchronous synthesizer is one in which a new set of coefficients $\{a_i, G\}$ is presented to it for each new pitch period (10 milliseconds average for male voices). The coefficients are held constant for the duration of the pitch interval. This assumes that the system properties are relatively slow-varying with respect to this time scale. The variation is provided by an outside source of information, and it can be obtained at the source either by analyzing real speech in an LPC analyzer, or by complete synthesis based on phonological rules. In the first case, the synthesizer functions as a receiver in a vocoding system, while in the second case, it is a final stage in a speech synthesis system. In addition to the LPC coefficients $\{a_i\}$ and gain coefficient G, the only other variable needed for speech synthesis is the pitch period of the impulse excitation in the case of voiced sounds, or a duration of noise excitation in the case of unvoiced sounds.

The input-output transfer function of a linear system can be preserved under a set of equivalence transformations performed on the $\{a_i\}$ parameters, which yield various representations of the same system. See Refs. 2 and 3. This fact is used to advantage by selecting a representation that is least sensitive to parameter perturbations and errors in finite length arithmetic. In addition, it should provide an easy

test of the stability of the system. One such representation is the lattice form of a linear system, described by the following set of difference equations (Fig. 1b),

$$f_{m-1}(n) = f_m(n) - k_m\, b_{m-1}(n-1)$$
$$b_m(n) = k_m\, f_{m-1}(n) + b_{m-1}(n-1)$$
$$m = N, N-1, \cdots, 1, \tag{2}$$

where $\{f_i$ and $b_i\}$ are forward and backward signals at an i-th lattice stage, and $\{k_i\}$ are relfection coefficients. The output of the system is $\{f_0(n)\}$, and the input is $\{f_N(n)\}$. Even though this form requires more multiplications per sample output, it has a very important property that all the reflection coefficients belong to the interval $(-1, 1)$ in a stable system.

A device that performs the function of a pitch synchronous synthesizer of the twelfth order in the lattice form has been designed and built around the digital signal processor chip. The device synthesizes speech in real time with an output sampling rate of 10 kHz. It is intended to be used in conjunction with some external device capable of providing the necessary system description for every new pitch period. This information is transmitted to the synthesizer in the form of a fifteen-word message, shown in Fig. 2, consisting of a header word used for synchronization and error recovery, a number representing the pitch period of excitation, an excitation amplitude, and finally the system parameters. The parameters are given as 15-bit reflection coefficients, or as 8-bit log-area parameters.

The basic synthesizer may be interfaced to the outside world in a number of ways, and this will be discussed.

II. DESCRIPTION OF THE SYNTHESIZER

The synthesizer block diagram is shown in Fig. 3. The main components are a digital signal processor chip, an interface for the input

HEADER: 0100000_8	
PITCH PERIOD	(0 FOR NOISE EXCITATION, DURATION 10 ms)
AMPLITUDE	(0 TO 255)
PARAMETER 1	(−1 TO +1, NORMALIZED)
⋮	
PARAMETER 12	(−1 TO +1, NORMALIZED)

Fig. 2—Message format.

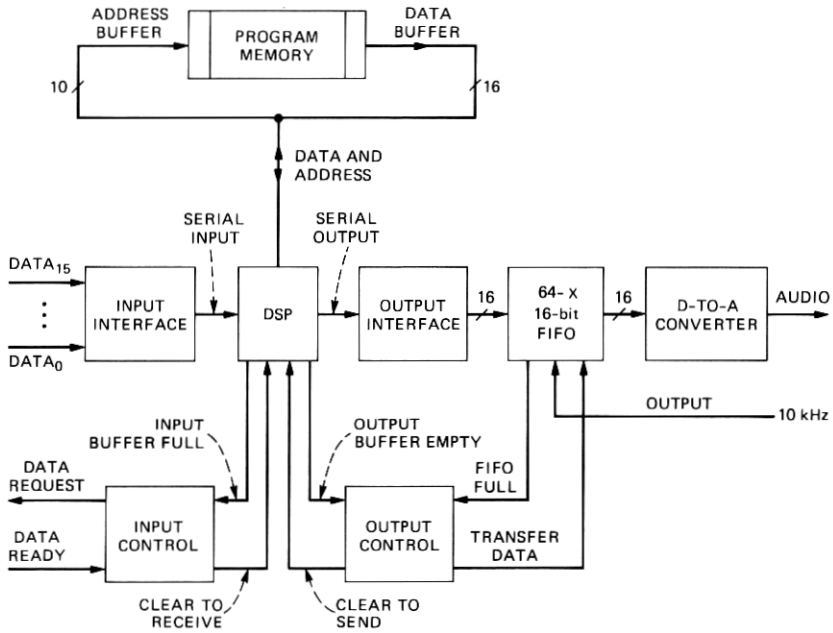


Fig. 3—Synthesizer block diagram.

data, and an output interface which includes a first-in-first-out buffer memory (FIFO). The output of the 64-word FIFO is converted to an analog signal through a digital-to-analog (D-to-A) converter.

The processor derives the synthesis program from a read-only memory (ROM), which is presently external to the processor, but could be contained in the processor's internal memory. The computation within the DSP is done in an arithmetic unit that operates on a 20-bit and a 16-bit operand. Notably, it includes an efficient multiplier and a 40-bit accumulator. These provide a dynamic range which is sufficient for the synthesis application. The parallel and pipelined architecture of the processor maintains a high computational throughput rate.

The processor utilizes a multiplexed address and data bus for accessing instructions/data stored in external memory. To transfer instructions or data from the memory, the processor places an address on the bus during the first half of the bus cycle. An address register is used in the synthesizer to latch the address, which specifies the appropriate memory location. The data from the memory is latched into an external data register, and transferred into the processor during the second half of the bus cycle.

The input interface of the synthesizer consists of a parallel-in serial-out shift register and associated control logic. Data requests from the

processor are passed to the outside source of information, and when a word is received, it is transferred to the processor bit-serially.

The output interface contains a serial-to-parallel converter, first-in-first-out memory buffer, and associated control logic. The output of the DSP is a bit-serial stream, which is shifted into the serial-in parallel-out shift register. The sample is then transferred from the shift register to the FIFO, where it joins the queue of previously computed samples. A sample is taken out of the FIFO queue every 100 microseconds under control of a 10-kHz clock, and applied to the digital-to-analog converter.

The role of the FIFO memory is to even out the differences in the three mutually asynchronous processes that are taking place during synthesis. The first process is the input of synthesis information into the DSP, the second is the computation of the speech samples, and the third is the output of the samples to the D-to-A converter. The input process is largely dependent on the host computer and its transmission capabilities. The use of the output FIFO relaxes the transmission requirements. Without the FIFO, the data would be required in a burst mode for each pitch period. Each burst would consist of 15 words transmitted within one sample period of 100 μs. However, since the FIFO queue contains the accumulated output samples for up to 64 sample periods, the input data rate is effectively decreased to 15 words every 6.4 milliseconds. The computational process within the DSP is asynchronous with respect to the output sampling rate of one sample/100 microseconds. The FIFO allows the DSP to compute new samples at full speed by providing the capability of saving the samples until they are needed. This queuing mechanism requires that the average computation time is less than the interval between output requests. The device meets this requirement, and in fact, the DSP performs the computations faster than required most of the time. This implies that there are times when the FIFO becomes full, at which point the computation will be suspended until output permission is granted to the DSP. This happens when a word is taken out of the FIFO.

III. THE SYNTHESIS PROGRAM

From the above description of the synthesizer, it is clear that it can be used for a number of different synthesis schemes, with a variety of speech representation algorithms (LPC, formant, etc.).

The data rate required by the synthesizer is a very important parameter for a practical implementation of a voice response device. The data rate is a function of a speech production model used by the synthesizer, and of a coding scheme used to represent model parameters in a segment of speech. A discussion of quantization schemes of LPC parameters and their effects to the quality of synthesized speech

is given in Ref. 2. Even though a significant decrease in the input data rate is possible with these approaches, we will describe an implementation that does not employ parameter quantization or interpolation. Such a scheme is useful when the device is used in a synthesis by rule system. Other applications may require some form of input data compression. The same physical device may be used in such cases, the only difference would be in the DSP program.

So far, we implemented two versions of the synthesis program for the synthesizer. Both of them employ the lattice form of a linear system. They differ only in the input data representation, one of them requires 15-bit reflection coefficients, and the other accepts 8-bit log-area parameters. Clearly, the data rate in the second program is much lower than in the first, with only a slight decrease in speech quality. In this program the input data is converted into reflection coefficients by a table look-up procedure. The relationship between the reflection coefficients and log-area parameters is given by

$$k_i = \frac{2^{A_i - A_{i+1}} - 1}{2^{A_i - A_{i+1}} + 1}, \tag{3}$$

where $\{A_i\}$ is a set of log-area parameters. This relationship is implemented in such a way that for each $A_i - A_{i+1}$ there is a value of k_i in a look-up table. Since the log-area parameters are specified by 8-bit numbers, the table contains 256 entries. Once the conversion is made, both programs function in the same way.

There are three major tasks that the program performs repetitively during the synthesis. The first task is obtaining the data for the synthesis of every new pitch period. The input is handled jointly by the program and the input interface. When the program requests data input, the interface obtains it from the host computer, and transfers it into the DSP. The protocol used by the interface is described in the next section. This procedure is repeated for each data request by the DSP, until all parameters describing the next pitch period have been transferred from the host computer.

Because of possible data transfer errors, a mechanism is provided for error recovery. Each pitch period requires a header word and 14 parameters. The program will proceed with the synthesis only if the header is received. This procedure is a sufficient guard against missed or inserted data words. In the worst case, one pitch frame will be incorrectly synthesized. Without the procedure, any inserted or deleted data word would create a permanent synchronization offset with serious perceptual consequences.

The second task of the synthesis program is to compute the speech samples by utilizing the input information and eq. (2). The program makes a decision to synthesize voiced or unvoiced sounds on the basis

of the pitch value. If the pitch period is encoded as a zero, the program simulates a transfer function driven by white noise for 100 samples (10 ms). The noise is computed in the program as a pseudo-random sequence of the length 8192. Otherwise, the input to the transfer function is a single pulse at the beginning of the pitch period, and the number of produced samples is equal to the specified pitch value. The amplitude information is used for scaling the noise value in case of unvoiced sounds, or for scaling the impulse amplitude for voiced sounds.

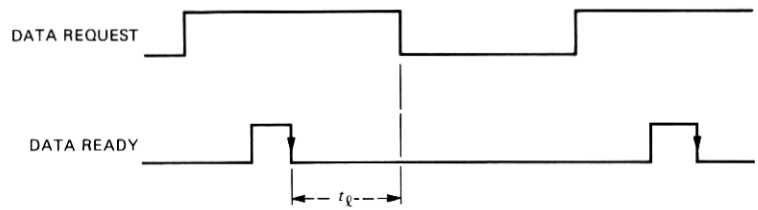
As soon as a sample is computed, it is output to the FIFO memory, and the program continues with the computation of the next sample. This final task of the program is conditioned upon the state of the FIFO buffer, if the buffer is full the processor waits until a word is removed.

IV. INTERFACE DETAILS

The control signals, which facilitate data transfers between the synthesizer and a host processor, consist of a data request signal generated by the device, and a data ready line activated by the host. These two control signals are sufficient to define a complete communication protocol with the synthesizer, so that the device can easily be integrated into a larger system.

The sequence of events that occurs during the synthesis procedure is shown in Fig. 4. When the DSP requests new data, the interface logic raises the data request signal. The host processor monitors this request, and places a new word on the data lines. When the data is stable, the host processor generates an edge-justified data ready signal. The word is latched in the input shift register 100 ns later. Once the input word is latched, a signal clear to receive is sent to the DSP. The DSP then generates the shift-clock pulses necessary to transfer the word into its input buffer. When the shifting is completed (at 400-ns/bit rate), the data request signal goes low, until the DSP makes another request for a new word.

The output of the DSP is enabled by means of a clear-to-send signal



t_ℓ : LATENT TIME EQUALS 6 MICROSECONDS

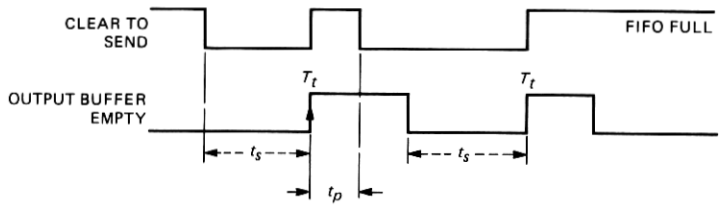
Fig. 4—Input protocol.

(CTS). This signal is granted to the DSP by the FIFO memory. The only times when this signal is not granted are when the FIFO queue is full, and for a short period after the word is placed in the queue. The latter is on the order of 4 microseconds, and is a result of internal data propagation in the FIFO. The output protocol is shown in Fig. 5. If the CTS is granted, the transfer of the data from the DSP to the FIFO memory is done in two phases. In the first phase, the DSP outputs a bit-serial data stream into the output shift register, and in the second phase, the information is transferred from the shift register into the FIFO. This transfer is done upon receipt of a positive transition of the signal Output Buffer Empty, generated by the DSP at the end of output shifting. If the FIFO memory becomes full, the CTS is not granted until a sample has been taken from the head of the queue and placed into the D-to-A converter.

There are two other variations of the basic synthesizer circuit that have been tested also. The first one is a slightly enhanced version of the synthesizer which includes an additional first-in-first-out memory buffer at the input of the system, shown in Fig. 6. The motivation for this input queuing mechanism is again based on the fact that the input process, which feeds the synthesizer with the system coefficients, is asynchronous with respect to the pitch frames. This configuration is especially useful when the host processor has to compute the reflection coefficients needed for synthesis in real-time, while the synthesizer is processing previously obtained parameters. Typically, the time to compute the coefficients has some variance, and this variance is compensated for by the "elasticity" of the input buffer.

The second variation of the basic circuit contains no FIFO buffers. It demonstrates a minimal synthesizer configuration, containing only 15 integrated circuits in addition to the DSP chip. It performs well when the host processor is capable of providing the input data at the required rate.

The synthesizer can easily be connected to standard microcompu-



t_s: SHIFT TIME EQUALS 6 MICROSECONDS
T_t: POINT OF TRANSFER
t_p: PROPAGATION THROUGH FIFO, EQUALS 4 MICROSECONDS

Fig. 5—Output protocol.

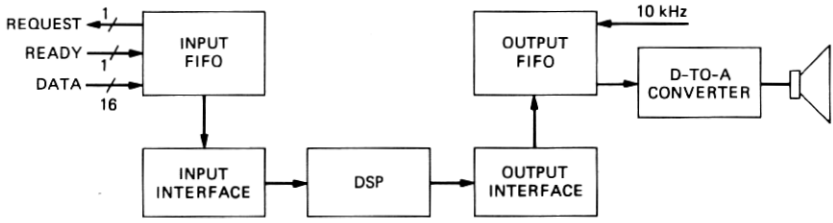


Fig. 6—Synthesizer with input FIFO memory.

ters. Several types of connections have been tried successfully. One of them is shown in Fig. 7, where the synthesizer is contained on a single board interfaced directly to a standard microprocessor bus. The input interface appears as a single memory location in the address space of the processor, and the data is transferred to it by a single "move" instruction. The input FIFO buffer may or may not be implemented. If it is, then the synthesizer interrupts the processor only when the buffer is empty. Without the buffer, the processor is interrupted for each data transfer.

Another type of interface, shown in Fig. 8, contains a synthesizer and direct-memory-access (DMA) circuitry. The iput and output buffers are not needed in this configuration, since the synthesizer obtains the data by accessing the processor memory. The processor sets up a DMA transfer by providing an address and a data count to the synthesizer board. It is interrupted only when the specified number of words have been processed by the synthesizer.

The third way of connecting the synthesizer to a microprocessor is by means of a standard parallel interface board, a standard accessory. In this case the synthesizer is not a part of the microcomputer system, rather it is an outside device.

All of these examples show that the synthesizer may easily be included as a part of intelligent terminals that are usually built around

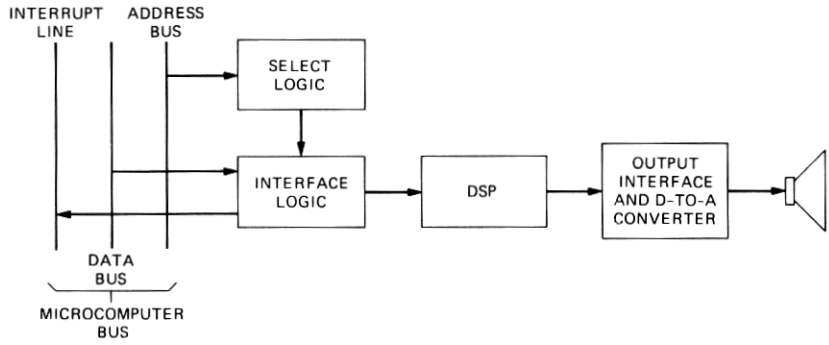


Fig. 7—Interface to microcomputer bus.

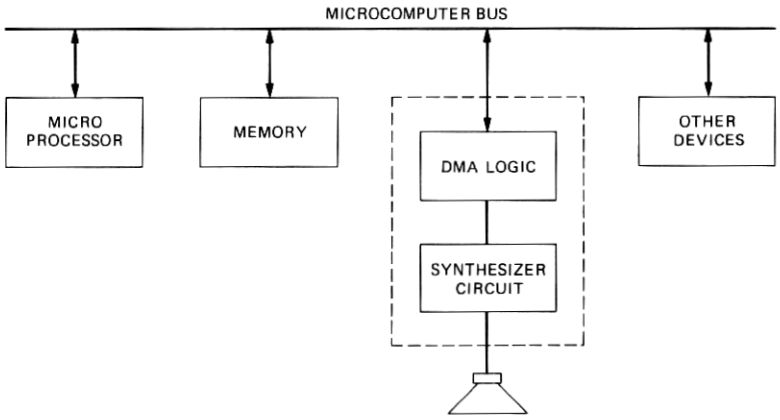


Fig. 8—Interface to microcomputer bus by DMA.

standard microprocessors. A combination of data and voice services can be provided with such configurations. Applications that require low data rate to the synthesizer would use a scheme with quantized LPC parameters, or a synthesis program that provides for interpolation of log-area parameters of longer time intervals.

V. SOFTWARE DRIVERS FOR THE SYNTHESIZER

In order to demonstrate the flexibility of the synthesizer, two types of host computers were used for implementation of the synthesizer drivers.

One is a stand-alone microcomputer, based on an LSI-11 microprocessor. This configuration includes an enhanced mini-*UNIX** operating system, and is intended for real-time speech processing experiments. A parallel interface port is used for driving the synthesizer. The hand-shaking signals required by the parallel port are in agreement with the ones provided by the synthesizer. The data request line interrupts the microprocessor, which then transmits a word to the synthesizer. At the same time a data ready pulse is issued, which is used by the synthesizer to latch the data.

A program that drives the synthesizer with the data from a file containing the speech parameters is used as:

say filename [loop] [frame]

The program reads the file **filename** into a buffer, and transfers it to the synthesizer on the basis of the protocol described earlier. The optional arguments **loop** and **frame** are used for testing purposes. If

* Registered trademark of Bell Laboratories.

a **loop** is present, the driver will synthesize the whole file, go to a mode of repeated transfers of a single pitch frame, whose sequential number is given in the argument **frame**, whose default value is the first pitch period. Once in this mode, the program allows for stepping through successive pitch frames, which is useful for studies of synthesized waveforms, and effects of finite length arithmetic to the stability of the lattice synthesizer. Also, an option is provided for changing the system coefficients during the repetitive pitch frame synthesis.

Another host computer used for driving the synthesizer is an SEL32-75 system. In this configuration, a DMA interface is utilized for transfer of pitch synchronous information to the synthesizer. The hand-shaking protocol between the DMA unit and the synthesizer input logic takes place without an involvement of the computer CPU.

VI. CONCLUSION

A real-time speech synthesizer, and several variations of it, have been designed based on the digital signal processor chip. The DSP has proven to be an important vehicle for digital signal processing applications, and speech processing in particular. A variety of sentences have been synthesized with the device. Informal listening shows no perceptual difference between the speech obtained by the synthesizer and by the general purpose computer using the same algorithm and floating point arithmetic. It has been demonstrated that the synthesizer can be easily integrated into microprocessor based systems for a number of voice response applications.

REFERENCES

1. B. S. Atal and S. L. Hanauer, "Speech Analysis and Synthesis by Prediction of the Speech Wave," J. Acoust. Soc. Amer., *50* (1971), pp 637–55.
2. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1978.
3. J. D. Markel and A. H. Gray, *Linear Prediction of Speech*, New York: Springer-Verlag, 1976.