THE BELL SYSTEM
TECHNICAL JOURNAL

VOLUME XLV MARCH 1966 NUMBER 3

Digital Computer Simulation of Sampled-Data Communication Systems Using the Block Diagram Compiler: BLODIB

By ROGER M. GOLDEN

(Manuscript received November 23, 1965)

Digital computer simulation of communication systems is accomplished readily by means of the system-oriented programming language called BLODIB (for BLOck DIagram Compiler, B). The language is designed for programming sampled-data systems which may be represented either in block diagram form or in the mathematical representation of the z-transform calculus. Contained within the language are 40 basic "building" blocks from which an entire communication system can be built. In addition, new blocks may be defined as consisting of complex configurations of basic blocks. The structure of BLODIB allows convenient specification of system parameters as well as permitting these parameters to be varied in order to study changes in system performance. The use of BLODIB is demonstrated by its application in the simulation of a voice-coding (vocoder) system.

I. INTRODUCTION

The study of complex communications systems by digital computer simulation is facilitated by use of the Block Diagram Compiler, BLODIB.[1] The source language accepted by the compiler is oriented towards communication system engineers and analysts who are already adept at handling block diagrams (or transfer functions) of complete systems. BLODIB is easily learned even by persons who have had little or no previous programming experience. The language of the compiler was designed specifically for simulating a wide variety of sampled-data

system problems as well as simulating sampled-data approximations to continuous (analog) systems. The capabilities of BLODIB make possible the simulation of systems which were impossible for the old BLODI[2] compiler to handle. In particular, simulation of voice-communication and speech bandwidth compression systems such as vocoders, are accomplished more easily. The digital computer simulation of these systems was pioneered at Bell Telephone Laboratories using the original BLODI language.[3,4] While such systems were programmed in a few weeks time with a minimum of difficulty, it is now possible to program the same systems in about one day.

II. NEED FOR SIMULATION

The fantastic growth of the demand for information handling and processing makes it absolutely essential that our communication systems be as efficient as technically possible. This means that all forms of information exchange including speech communications should be encoded in such a way as to assure the most efficient use of a communication or data channel. The complexity of terminal equipment associated with such coding systems is such that a great deal of design, testing, and redesign must take place before these systems can be used on a regular basis. The testing and debugging (including redesign where necessary) of such equipment requires many costly man-hours. Much of this cost may be reduced or even avoided if the entire system is first evaluated by simulation on the digital computer. This technique provides the flexibility for systematically optimizing system design. The advantage of using a digital computer as a system simulator lies principally in the ease with which system parameters can be changed. Hence, many designs may be tested before choosing the one that would be built for the actual system. It is just this technique that is now being applied to the design and evaluation of voice-communication systems.

III. PREREQUISITES FOR COMPUTER SIMULATION

Three prerequisites must be completed in order to accomplish a digital computer simulation of a voice-communication system. The first is the determination of an appropriate sampled-data (or digital) representation for the system to be studied; the second is the preparation and compilation of a BLODIB computer program which causes the computer to perform the same operations as would be performed by the actual system; and the third prerequisite is the digitalization of real speech for processing by the computer.

These prerequisites will now be treated with respect to a specific voice-communication system for bandwidth compression. This system is more commonly known as a vocoder (an acronym for voice-coder). The vocoder, invented over 35 years ago by Homer Dudley[5] at Bell Telephone Laboratories, codes voices or speech signals for transmission over a communication channel. Parameters describing the speech are transmitted to a receiver where they are used to synthesize a replica of the original speech sounds. Bandwidth compression is achieved because the bandwidth required to transmit these parameters is considerably less than that required to transmit the original speech sounds. It is helpful at this point to give a brief review of vocoder operation before describing how the above prerequisites for simulation are accomplished.

Fig. 1 shows a functional block diagram of a spectrum channel vocoder and in itself can be used as an aid in the writing of the simulation program. The vocoder consists of two main parts: an analyzer for extracting that information which must be transmitted, and a synthesizer for reconstituting the original speech signal.

The analyzer portion of the vocoder consists of two basic parts: a short-time frequency spectrum analyzer and an excitation detector. The spectrum analyzer consists of a number of bandpass filters for separating contiguous bands of frequencies. The output of each analyzing filter is rectified and then smoothed by low-pass filters. The outputs of the smoothing filters are low-frequency signals which represent the short-

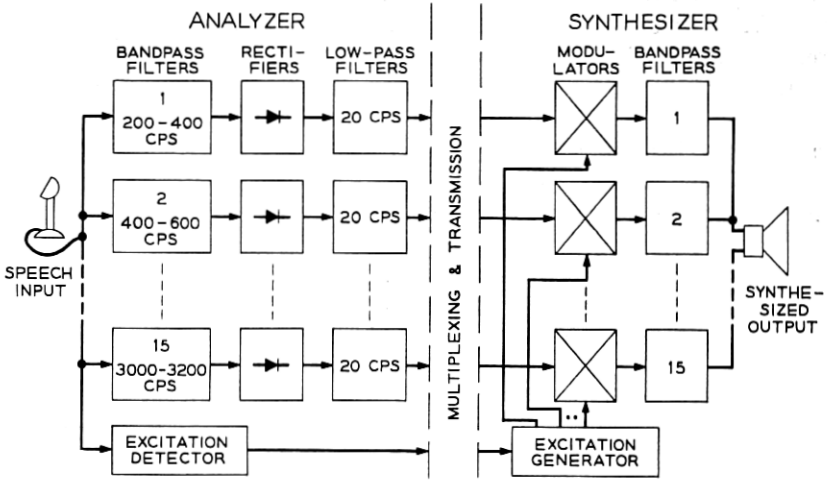

Fig. 1 — Functional block diagram of vocoder.

time spectral energy in each of the analyzed bands. These signals represent the slowly varying characteristics of a speech signal. Physically, they are just the average energy of the signals from the bandpass filters.

The output of the excitation detector (be it conventional pitch detector with voiced-unvoiced decision or a narrow band of the input speech) is transmitted by a suitable frequency or time multiplexing method along with the slowly varying spectrum signals. This information is used by the synthesizer portion of the vocoder to regenerate the input speech.

Synthesis is accomplished by amplitude modulating the frequency harmonics of a locally generated excitation signally the corresponding low-frequency signals. The harmonics of the excitation function are obtained from a set of filters (not shown in Fig. 1) similar to those used for analyzing the input speech signal. The modulated signals are then band limited in order to remove spurious sidebands introduced by the modulation process. The combined output of these filters is a replica of the input speech. Hence, wideband speech is synthesized from a set of low-frequency signals having a relatively narrow over-all bandwidth. This, then, is the complex communication system for which a suitable sampled-data representation is sought.

IV. SAMPLED-DATA REPRESENTATION OF A CONTINUOUS SYSTEM

In order that a suitable linear time-invariant sampled-data representation be found for the vocoder shown in Fig. 1, it is necessary that the transfer functions be known for the various blocks in the system. The representation of these transfer functions may be in either the time domain (a relation between the input time signal and the output time signal) or the frequency domain (a relation between the frequency spectrums of the input and output signals). For example, the various bandpass and low-pass filters are first represented by their frequency domain transfer functions in the form

$$H(s) = \frac{C_0 \prod_{k=1}^{K} (s - \alpha_k)}{\prod_{l=1}^{L} (s - \beta_l)} \tag{1}$$

where

C_0 = constant,
α_k = complex zeros of $H(s)$,
β_l = complex poles of $H(s)$,

K = degree of numerator,
L = degree of denominator, and
s = complex Laplace transform variable.

Conversion of this type of function to an appropriate sampled-data form is accomplished by applying either the z-transform or bilinear z-transformation[6,7] to the above transfer function.

The resulting z-transfer function is then written as a partial fraction expansion in the form,

$$\mathcal{H}(z) = \sum_{m=1}^{M} \frac{A_{1_m} z^{-1} + A_{0_m}}{B_{2_m} z^{-2} + B_{1_m} z^{-1} + 1} \tag{2}$$

where

A_{1_m}, A_{0_m}, B_{2_m}, B_{1_m} are constants,
z^{-1} = exp $(-sT)$ = the unit delay operator, and
T = unit sampling interval.

Each term of the above expansion is then represented in block diagram form as shown in Fig. 2(a). Fig. 2(b) shows the realization of the complete transfer function. This technique is used to realize all of the linear transfer functions in the system. (Incidentally, the design and

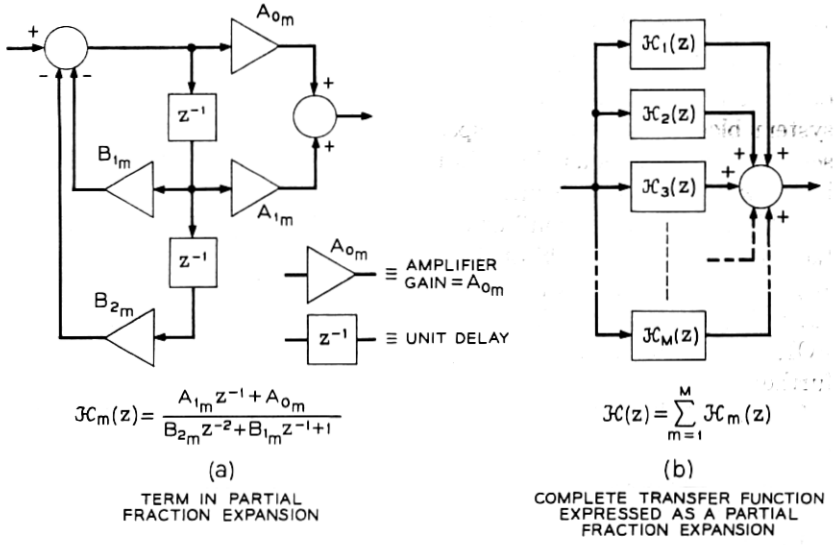

Fig. 2 — Block diagram representation for a partial fraction expansion of a sampled-data transfer function.

BLODIB realization of such functions is readily carried out by digital computer. The output from such a design program is illustrated in the following section.

The function performed by the rectifier is

$$r_{\text{out}}(t) = | r_{\text{in}}(t) | . \tag{3}$$

The sampled-data representation of this function in the time domain is simply

$$r_{\text{out}}(nT) = | r_{\text{in}}(nT) | . \tag{4}$$

The "chopper-modulators" perform the function

$$m_{\text{out}}(t) = m_{\text{in}_1}(t) \cdot \text{sgn } (m_{\text{in}_2}(t)) \tag{5}$$

which is realized by

$$m_{\text{out}}(nT) = m_{\text{in}_1}(nT) \cdot \text{sgn } (m_{\text{in}_2}(nT)). \tag{6}$$

Similarly, appropriate sampled-data representations were made for the excitation detector and generator. This, then, completes the first prerequisite for computer simulation. Next follows the presentation of a BLODIB computer program using the above information.

V. THE BLODIB PROGRAMMING LANGUAGE

As stated earlier, the BLODIB[1] programming language is system oriented in that it allows a direct representation of a communication system block diagram. More specifically, the language is a verbal description of the various blocks in the diagram and information concerning how the blocks are interconnected. In addition, BLODIB makes possible the "construction" or definition of new types of blocks (using facilities called MACROS or SUPERS) from the basic types available. (Special boxes not in the basic set of 40 may also be introduced by supplying an external algorithm coded as a FAP subroutine or as a FORTRAN function. However, this technique will not be discussed further here.)

The use of the MACRO facility as well as the simplicity of BLODIB coding can be demonstrated easily with respect to the realization of the filter transfer functions. Fig. 3 shows the actual BLODIB configuration for the basic terms required in the partial fraction expansion given by (2). Since this "term" or function is required many times, it will be defined as a new type — ADB — and thereafter used as a basic building block. Coding in BLODIB requires giving each block a name (chosen by

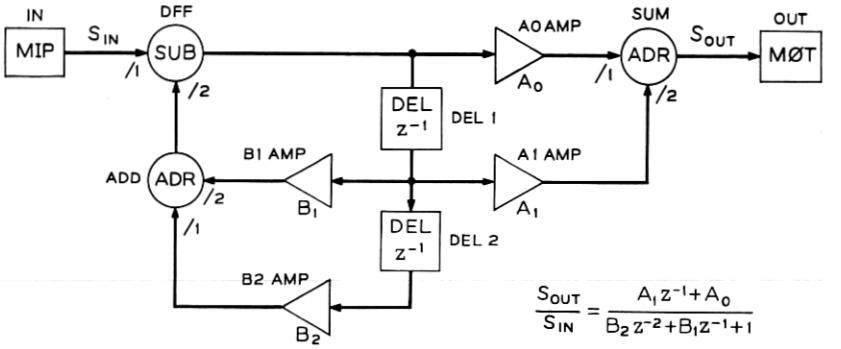

Fig. 3 — BLODIB representation for the transfer function
$\mathcal{H}(z) = (A_1 z^{-1} + A_0)/(B_2 z^{-2} + B_1 z^{-1} + 1)$

the programmer); designating the type of block from the list of basic types, specifying any parameters associated with the block; and finally, listing the names of other blocks to which the output should be connected. Thus, the actual coding for Fig. 3 is given in Table I. The first line of coding defines ADB as a SUPER which may have up to 4 inputs (I1, I2 etc.) and has 4 parameters A1, A0, B2, B1. The boxes MIP and MOT are required input/output boxes for SUPERS. The term END designates the end to the coding for ADB. The order of appearance of the boxes within the SUPER is immaterial since the compiler builds internally a connection matrix from which the blocks are ordered to produce an efficient program.

The partial fraction expansion terms in the various filter transfer functions are then realized using this definition for ADB. Since all of the filters, bandpass and low-pass, are used in several places, these

TABLE I — BLODIB CODING FOR THE TRANSFER FUNCTION
$\mathcal{H}(z) = (A_1 z^{-1} + A_0)/(B_2 z^{-2} + B_1 z^{-1} + 1)$

ADB	MACRO	I1, I2, I3, I4, A1, A0, B2, B1
IN	MIP	1, DFF
DFF	SUB	A0AMP, DEL1
DEL1	DEL	1, B1AMP, A1AMP, DEL2
DEL2	DEL	1, B2AMP
B1AMP	AMP	B1,, ADD
B2AMP	AMP	B2,, ADD/2
ADD	ADR	DFF/2
A0AMP	AMP	A0,, SUM
A1AMP	AMP	A1,, SUM/2
SUM	ADR	OUT
OUT	MOT	
	END	

too are coded as SUPERS. Hence, wherever a filter is required, it can be referenced as a single block. In this way, a complete set of filter blocks is built up quickly from the basic BLODIB language.

Figs. 4 and 5 show, respectively, the frequency and time response of one of the bandpass filters required in the simulation. (These graphs along with the BLODIB programming required for simulation were produced by a special computer program for determining sampled-data filter transfer functions. The block diagram for this filter is shown in Fig. 6. Table II presents the actual BLODIB coding.

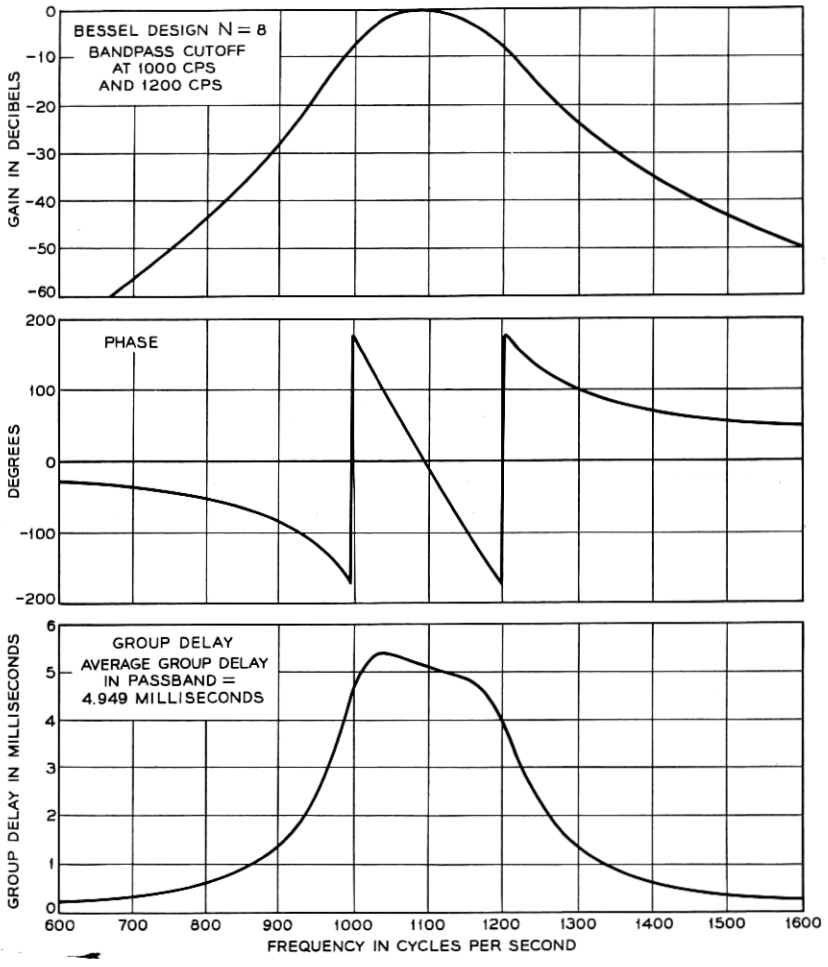

Fig. 4 — Frequency response for a vocoder-channel bandpass filter.

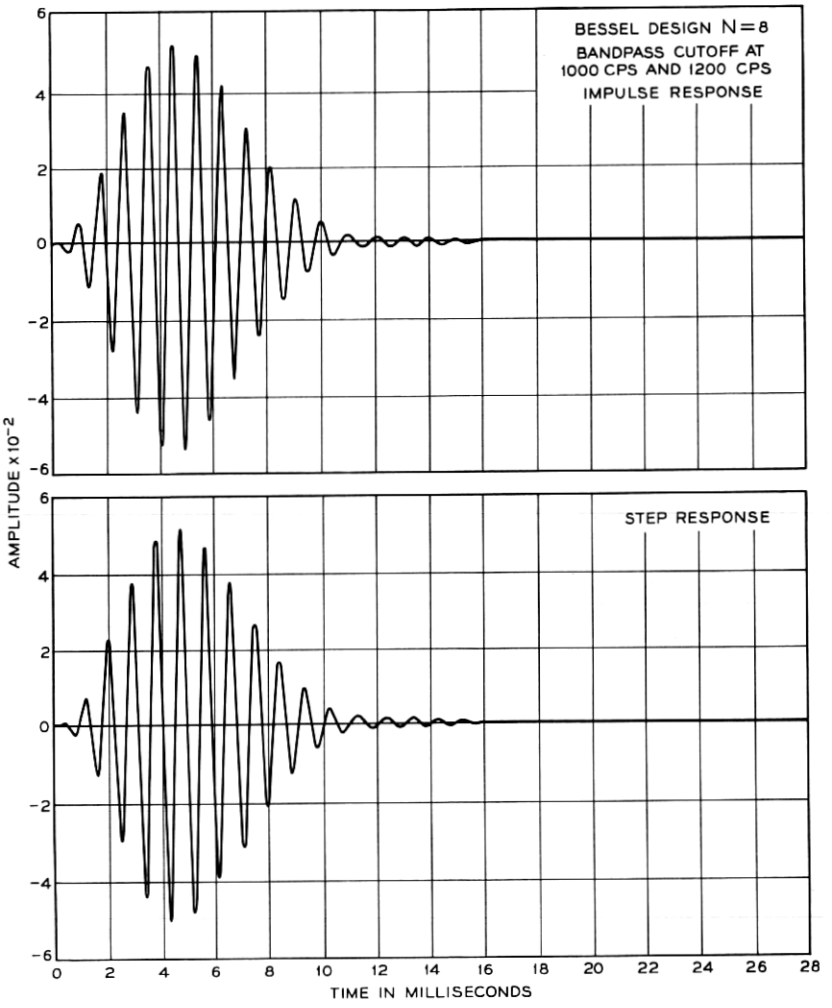

Fig. 5 — Time response for a vocoder-channel bandpass filter.

Having determined the representation for the various filters, the next step required is that of coding each of the 15 channels in the vocoder. From Fig. 1, it is seen that each channel (omitting multiplexing and transmission) consists of a bandpass filter, a full-wave rectifier, a low-pass filter, a "chopper" modulator, and another bandpass filter. In addition, a third bandpass filter is used between the output of the excitation generator and the modulator. A block diagram for one of these

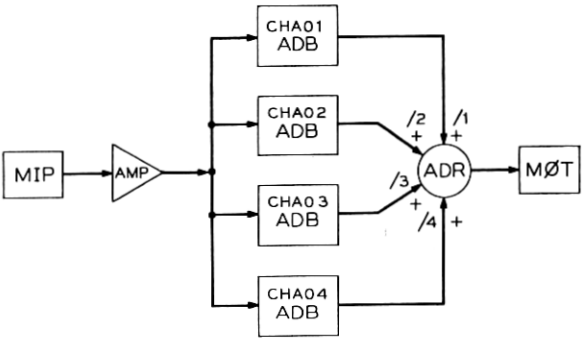

Fig. 6 — BLODIB representation for a vocoder-channel bandpass filter.

channels is shown in Fig. 7 along with the BLODIB coding necessary for simulation. The power of BLODIB with its "SUPER" capability is now readily apparent. Not only is the actual coding for each channel simplified, but any or all of the filters may be changed (by substitution of different SUPERS) without the necessity for recoding the entire system.

Completion of the above second prerequisite is usually all that is required in the programming of an entire system. However, two other features of BLODIB which also are used advantageously will be discussed briefly. These are the modular programming feature called SSUBR and the subroutine feature called SUBR. The SSUBR feature allows a block of BLODIB coding (similar to that contained within a SUPER) to be used as an external "module" to a main BLODIB program. However, unlike SUPERs which must be specified and compiled within the main program, SSUBRs are coded and compiled separately. These programs are then loaded as subroutines for use by the main program.

For example, the MOD (chopper) box shown in Fig. 7 was coded as an SSUBR and used in conjunction with the main vocoder program. This allowed different modulator configurations to be tested by simply "plugging" them into the program deck at run time. Multiplexing systems also were tested by using different external SSUBRs.

The SUBR feature permits coding an entire simulation so that it can be controlled by a main or "executive" program. (Such a program may be coded in FORTRAN or FAP.) This permits changing system parameters or using intermediate analysis programs as an integral part of the simulation.

At present, the simulated vocoder consists of a main FORTRAN program which supplies external parameters and provides the calling

TABLE 2 — BLODIB CODING FOR A VOCODER-CHANNEL BANDPASS FILTER

```
CHA       MACRO    I1, I2, I3, I4
IN        MIP      1, CHAI
CHAI      AMP      1., −4, CHA01, CHA02, CHA03, CHA04
CHA01     ADB      4
                   −448.005981
                     74.456865
                    114.566368
                   −189.968182
                    CHA05/2
CHA02     ADB      4
                    276.439266
                    193.008114
                    113.922181
                   −184.224327
                    CHA05/3
CHA03     ADB      4
                    187.295624
                   −112.815660
                    118.592467
                   −198.106075
                    CHA05/4
CHA04     ADB      4
                    −11.099112
                   −154.649326
                    117.119220
                   −180.637222
                    CHA06/2
CHA05     ADR      CHA06
CHA06     ADR      MOT
MOT       MOT
          END
```

sequences for the following BLODIB subroutines (SUBRs): an input routine which allows either the conventional mode or the voice-excited vocoder mode to be used; the main vocoder program; and, the output routine. The main vocoder program uses BLODIB SSUBRs for the excitation detector and generator and for the multiplexing and transmission networks. Fig. 8 shows how the various programs are interconnected. Variation of parameter or network configuration is accomplished by changing either parameters within a given block or changing the block completely. In this manner, optimization of system performance was achieved by optimizing the appropriate "modules" in the simulation. Hence, the programming of a complete vocoder system is achieved in an extraordinarily short amount of time using BLODIB.

Having completed the first two important prerequisites for simulation — namely sampled-data representation of the vocoder and its translation into the BLODIB programming language — there remains only the final step of preparing input speech data to be processed by the com-

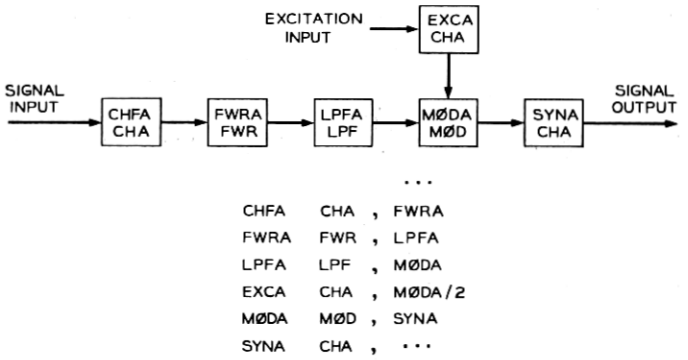

Fig. 7 — BLODIB representation and coding for a complete vocoder channel.

puter. This is accomplished by an analog-digital translator as indicated in Fig. 9. The translator, which can sample prerecorded input speech at rates up to 20 kcps, quantizes the input speech signal to one of 4096 levels (including sign). The sampled-and-quantized data is recorded at 800 bits/inch on standard digital magnetic tape in a format used by the BLODIB input/output routines. Running time on the IBM 7094$^{\text{Mod II}}$ for typical vocoder simulations is about 100 times real time (i.e., 1 second of speech requires 100 seconds for processing). The results from the simulation programs are similarly recorded on digital tape and then

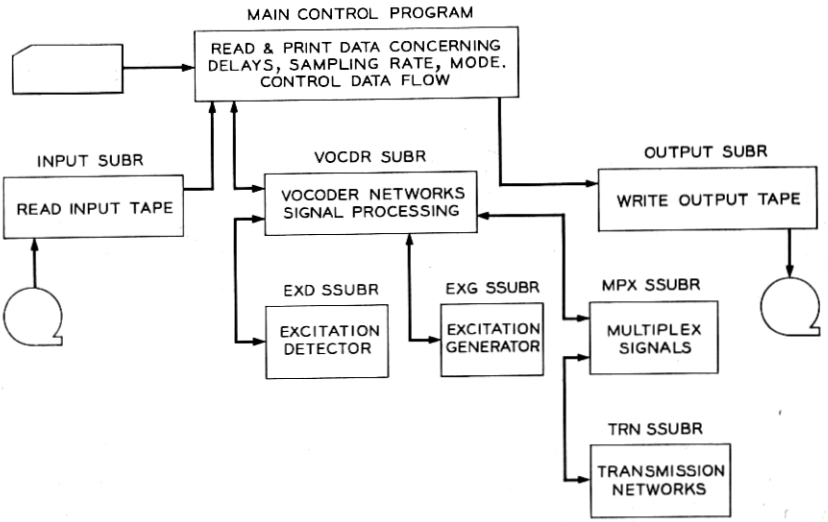

Fig. 8 — Flow diagram for vocoder simulation program.

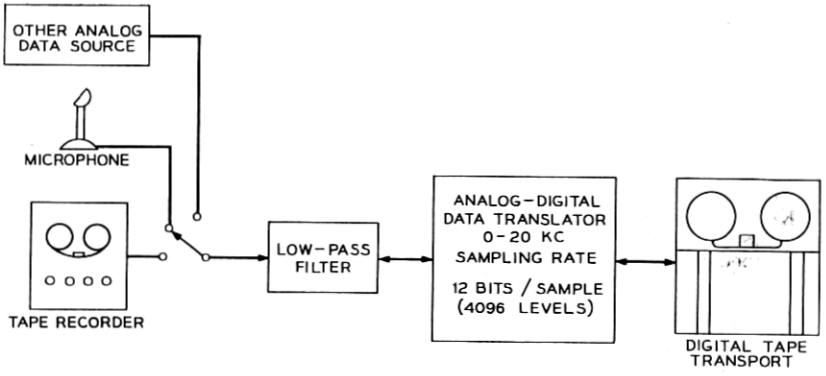

Fig. 9 — Preparation of digital input data for use with computer simulation of speech communication systems.

played back through the data translator. Of course, digital input tapes can be used over and over again, thereby removing this third prerequisite from succeeding simulations.

Thus far, BLODIB computer simulations have indicated: what filter designs yield good synthesized speech, what type of multiplexing and transmission systems might be satisfactorily used on the spectrum channel signals, how many spectrum channels are required, etc. As a result of these findings, improvement of system performance has been accomplished in a relatively short time and without the inherent delay and cost of building complex equipment.

VI. SUMMARY

Digital computer simulation of communication systems has been simplified and made more flexible by use of the BLODIB programming language.

Three prerequisites to actual simulation have been presented. They are:

(*i*) the finding of an appropriate sampled-data representation for the particular communication system,

(*ii*) the preparation of a BLODIB computer program for the above sampled-data representation, and

(*iii*) the preparation of input signals for processing by the program.

Details with respect to the BLODIB programming of the second step above were illustrated by their applications in the simulation of a vocoder system. Hence, new and complex communication systems can

be evaluated, optimized, and redesigned quickly and efficiently before final construction in hardware.

REFERENCES

1. Karafin, B. J., The New Block Diagram Compiler for Simulation of Sampled-Data Systems, AFIPS. Conference Proceedings, *27*, pt. 1, 1965, Fall Joint Computer Conference, Spartan Books, Washington, D.C., pp. 55–61.
2. Kelly, J. L., Jr., Lochbaum, C., Vyssotsky, V. A., A Block Diagram Compiler, B.S.T.J., *40*, May, 1961, pp. 669–676.
3. Schroeder, M. R., Logan, B. F., Prestigiacomo, A. J., New Methods for Speech Analysis-Synthesis and Bandwidth Compression, Fourth International Congress on Accoustics, Copenhagen, August 21–28, 1962.
4. Golden, R. M., Digital Computer Simulation of a Sampled-Data Voice-Excited Vocoder, J. Acoust. Soc. Amer., *35*, Sept., 1963, pp. 1358–1366.
5. Dudley, H., Automatic Synthesis of Speech, Proc. Nat. Acad. Sci., *25*, July, 1939, pp. 377–383.
6. Kaiser, J. F., Design Methods for Sampled-Data Filters, Proc. First Allerten Conference on Circuit and System Theory, Monticello, Illinois, Nov., 1963.
7. Golden, R. M. and Kaiser, J. F., Design of Wideband Sampled-Data Filters, B.S.T.J., *43*, July, 1964, Part 2, pp. 1533–1546.