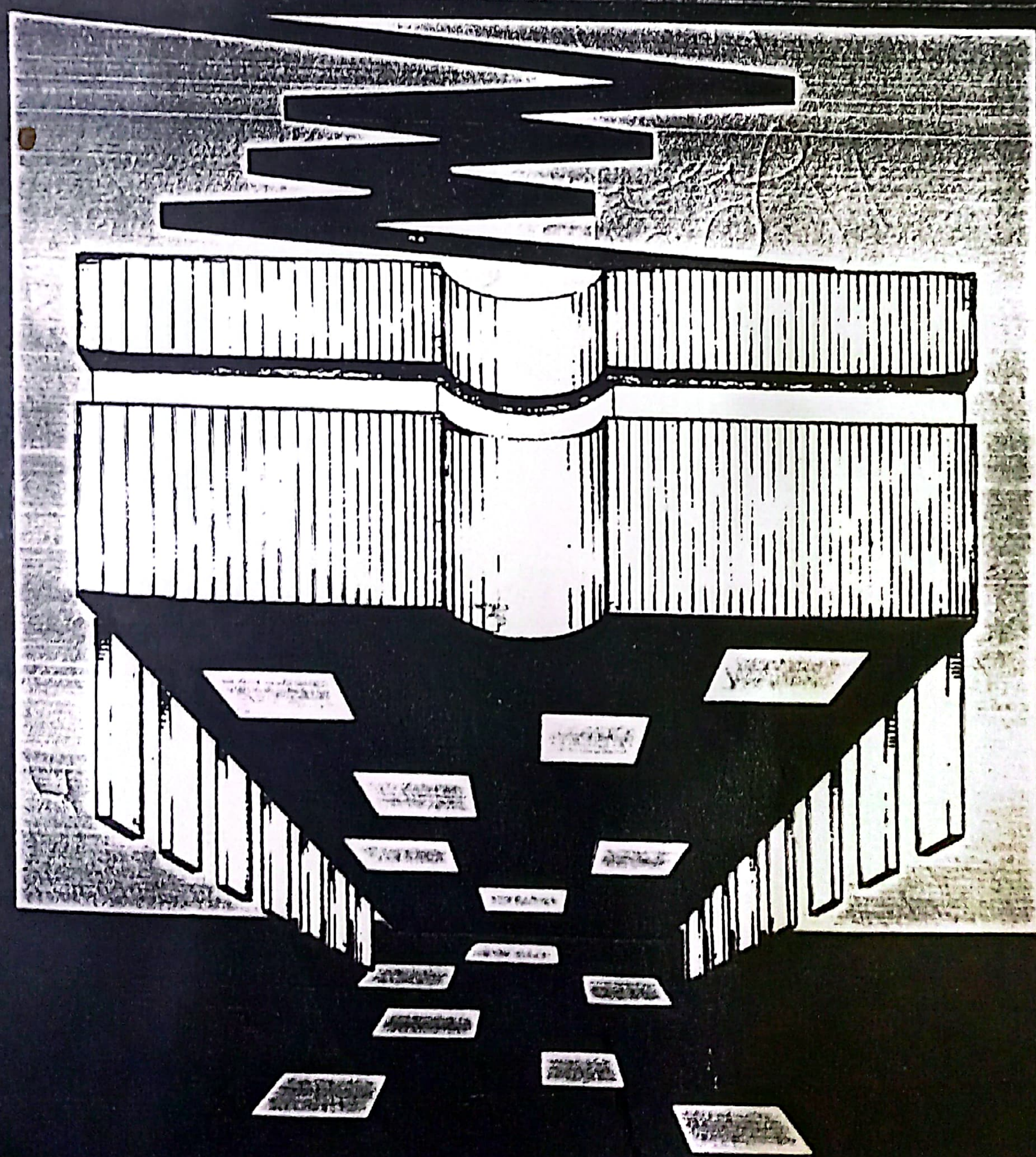
SAMS

22173

Electronically Hearing: Computer Speech Recognition

John P. Cater



SAMS

Electronically Hearing: Computer Speech Recognition

Cater

22173

CHAPTER 4

Speech Signal-Acquisition Techniques

It's time to take a break. The chapters of this book, so far, have discussed sciences that seem to lack a certain amount of precision. Each time we felt we had a firm grasp of an idea or subject, our illusion was shattered by ambiguities and interpretational difficulties. As we move our consideration toward electronic speech processing, things become more precise. In this chapter, we will begin our examination of electronic speech processing with the first element of its implementation: speech acquisition. The techniques discussed herein are those used by speech recognition systems to convert the acoustic speech signal to appropriate electronic data for eventual intelligence extraction and processing.

Two methods stand out as primary candidates for optimal acquisition techniques. The first, direct signal digitization or waveform encoding, as it is also known, is accomplished by simply digitizing the analog voltage from a microphone. The second technique, which we will call spectral acquisition, requires more complexity in the processing of analog information. Now, taken one at a time, here's how they work.

Direct Waveform Acquisition

Two methods of direct waveform acquisition are waveform encoding and delta modulation encoding.

Waveform Encoding

The speech acquisition technique known as direct waveform encoding is a relatively simple process where an analog signal is converted to a

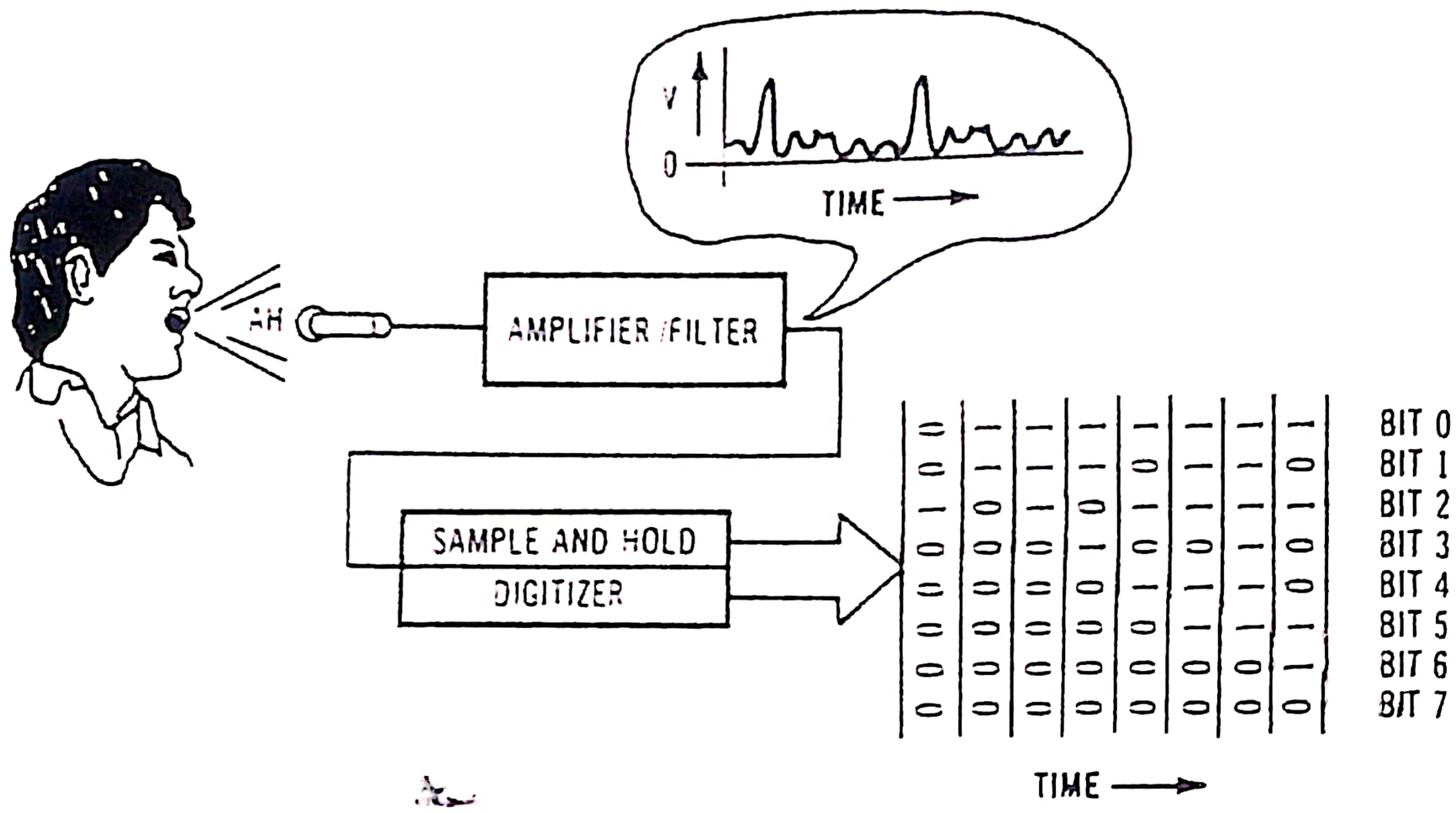


Figure 4-1.
Speech digitization.

serial string of binary data. The binary data may be either in single bit form or parallel byte form. Fig. 4-1 illustrates the speech digitization process in which the speech analog waveform is converted to a serial string of 8-bit bytes for each data sample. What is really occurring in this sequence of events is that at periodic intervals a *sample-and-hold* circuit captures a snapshot of the analog incoming voltage. which is immediately converted to a voltage-equivalent parallel data byte by an analog-to-digital (a/d) converter.

The same process may be viewed in terms of electronic blocks in the diagram in Fig. 4-2. Not only does this diagram illustrate the individual

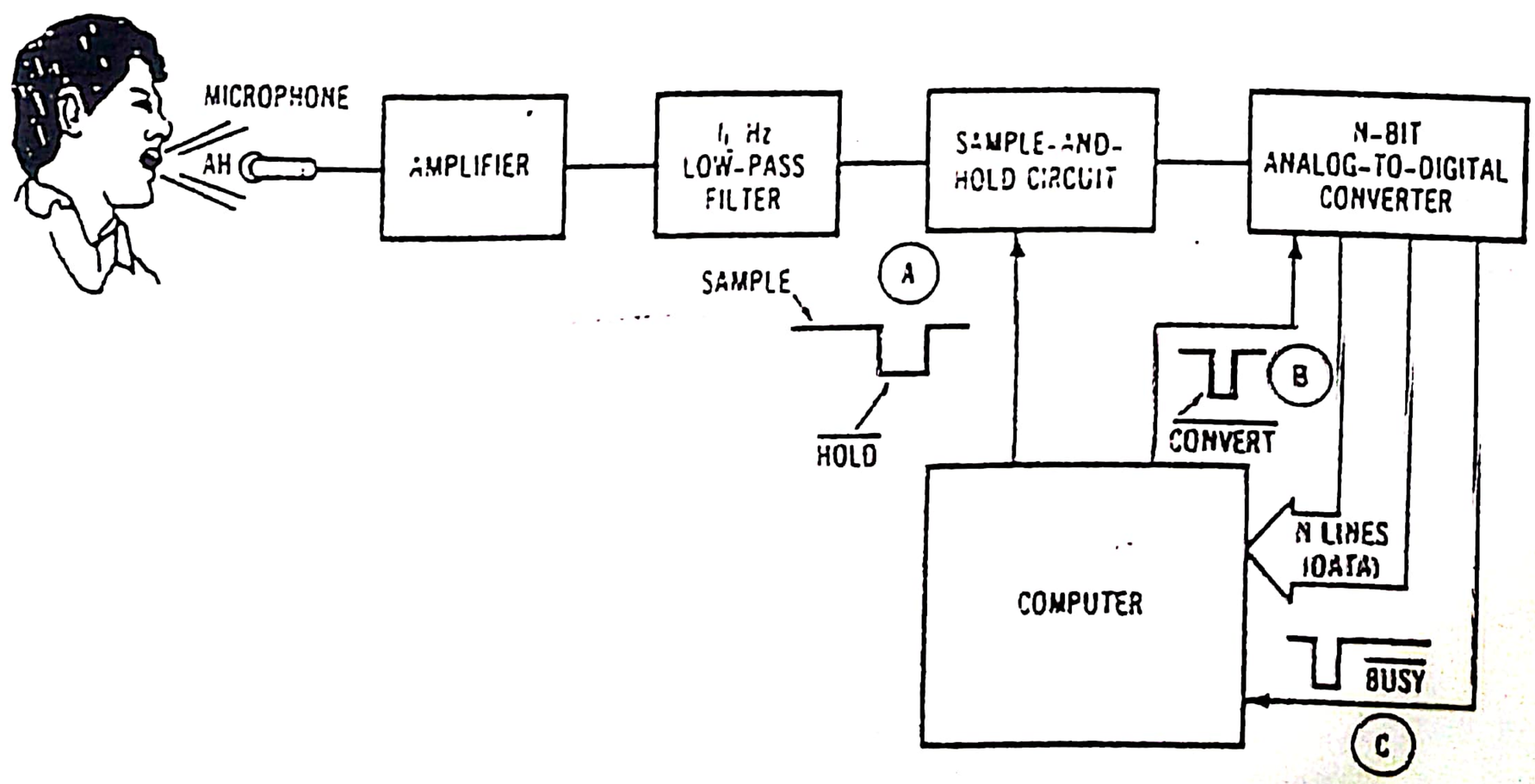


Figure 4-2.
Representation of speech waveform encoding.

blocks within the waveform encoding process, it also represents the control signals to and from the computer which govern the digital encoding process. The signal flow in this diagram is from the microphone to an amplifier that boosts the speech voltage levels to usable values for subsequent digitization. Immediately following the amplifier is a low pass filter to remove high frequencies that might interfere with the sampling process. (This will be discussed in more detail momentarily.) The output of the low pass filter (which is still an analog voltage) is processed by the sample-and-hold circuit exactly as its name implies. Upon command from the computer, the circuit that has been continuously *sampling* the input waveform briefly *holds* the current analog voltage during the digitization process. Following the release of the hold command, the sample-and-hold circuit returns to the sample or *tracking* mode.

During the time that the sample-and-hold circuit is holding the analog voltage, the N-bit analog-to-digital converter performs the actual digitization of the "held" speech waveform. Conversion times for normal speech digitizers range anywhere from 2 to 100 microseconds. Following the conversion to a digital byte, the a/d converter signals the computer that it has completed the assigned task. The controlling computer may now input the digital byte corresponding to the temporary voltage level and repeat the entire cycle until the full desired waveform has been captured.

The sequence of events is best put into perspective by observing the control signals and their time relationship during the process. Fig. 4-3 presents the three waveforms plotted in the same time frame reference to illustrate their temporal relationships. In the figure, notice that the sample-and-hold command (A) is the first control line to change levels. Following a brief delay (on the order of a few microseconds), which allows the sample-and-hold amplifier to stabilize. the computer commands the a/d converter to convert by briefly taking the *convert* command line (B) low. As the a/d converter recognizes that it has been given a command to convert, it immediately takes its *busy* response line low indicating to the computer that it is, in fact, "busy" converting. The important feature of the three waveforms is that the *hold* signal (A) stays low until the signal conversion has completed. The computer must do this through logic to ensure that the digitizer does not try to digitize a changing input signal. (If the input signal to the a/d converter were allowed to change during conversion, errors in quantization will occur which the computer will view as audio distortion.) Finally, after the a/d converter has taken its *busy* line back high again, the computer may reset the sample-and-hold circuit to its sample mode, and then delay for the next periodic conversion time out.

A few paragraphs ago, I mentioned that we would come back to the effect of low-pass filtering upon sample time. The actual need for the low-pass filter arises because speech frequencies can easily extend up to and beyond the sampling frequency. In most systems, speech fre-

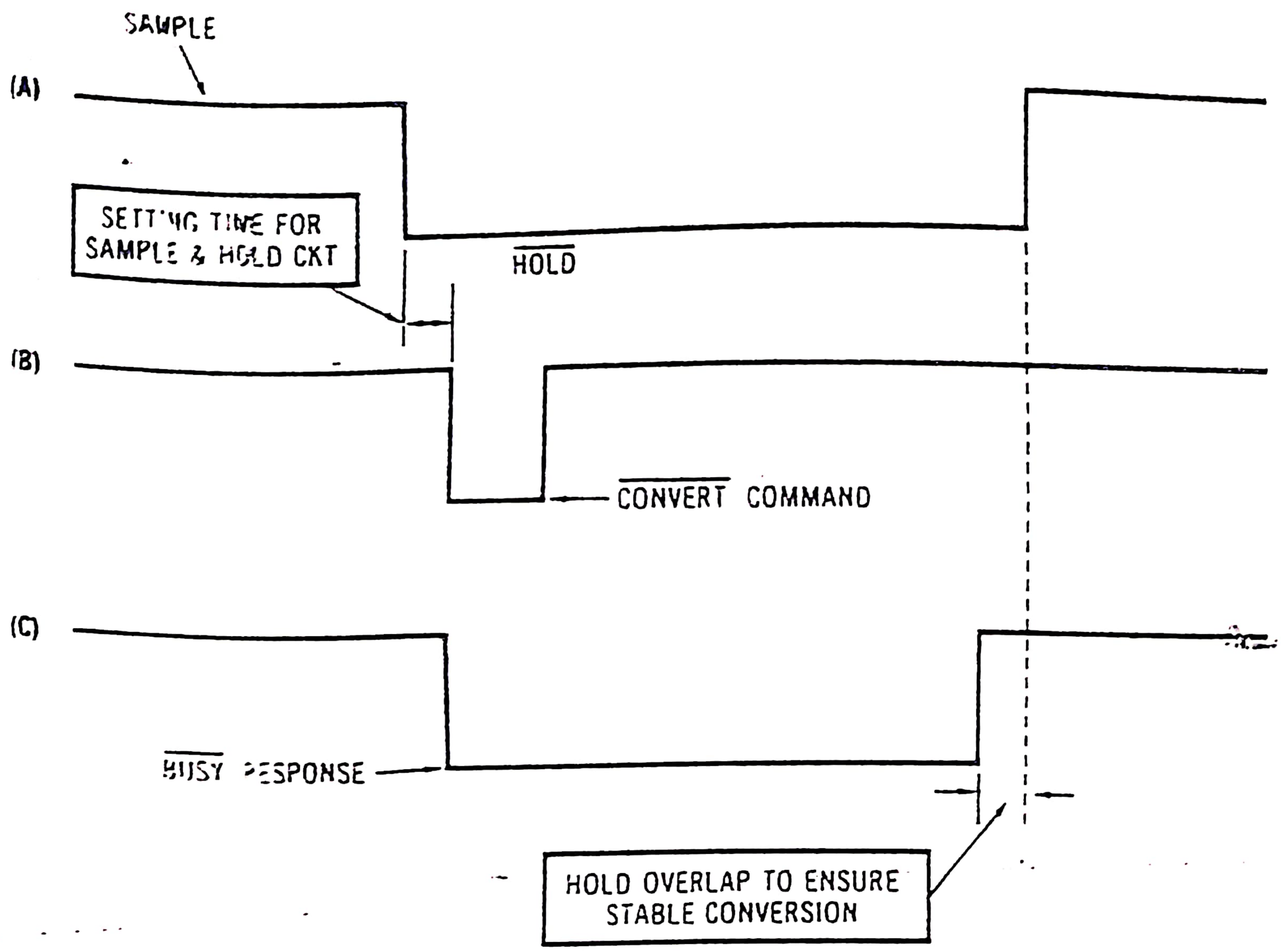


Figure 4-3.
Digitization control signals.

quencies beyond 3 to 4 kHz are redundant and provide little more information than those below that range. Now, if we can find some reason for lowering the sample rate to a minimum value, then we are putting less bytes per second of data into memory (the net effect is to allow the digitization of longer words or phrases for a given fixed size memory).

To determine a sample rate for adequate speech input digitization, we must keep in mind a powerful theory of sampled data discovered by a fellow named Henry Nyquist of Bell Laboratories many years ago. His perceptive observations of sampled analog waveforms detailed a phenomenon that occurs when the frequency of the incoming waveform reaches and extends beyond one-half of the sampling frequency. When described in words, the manifestations of the sampling theorem are quite complex. However, if we view the physical process of a changing sampling rate in Fig. 4-4, we can easily see the artifacts of an increasing sample rate.

The original signal shown in Fig. 4-4A corresponds to any incoming waveform which is—for sake of clarity—a cosine waveform at frequency f_o. The waveform in Fig. 4-4B is the sampled voltage equivalent when sampled at a frequency equal to eight times the original fre-

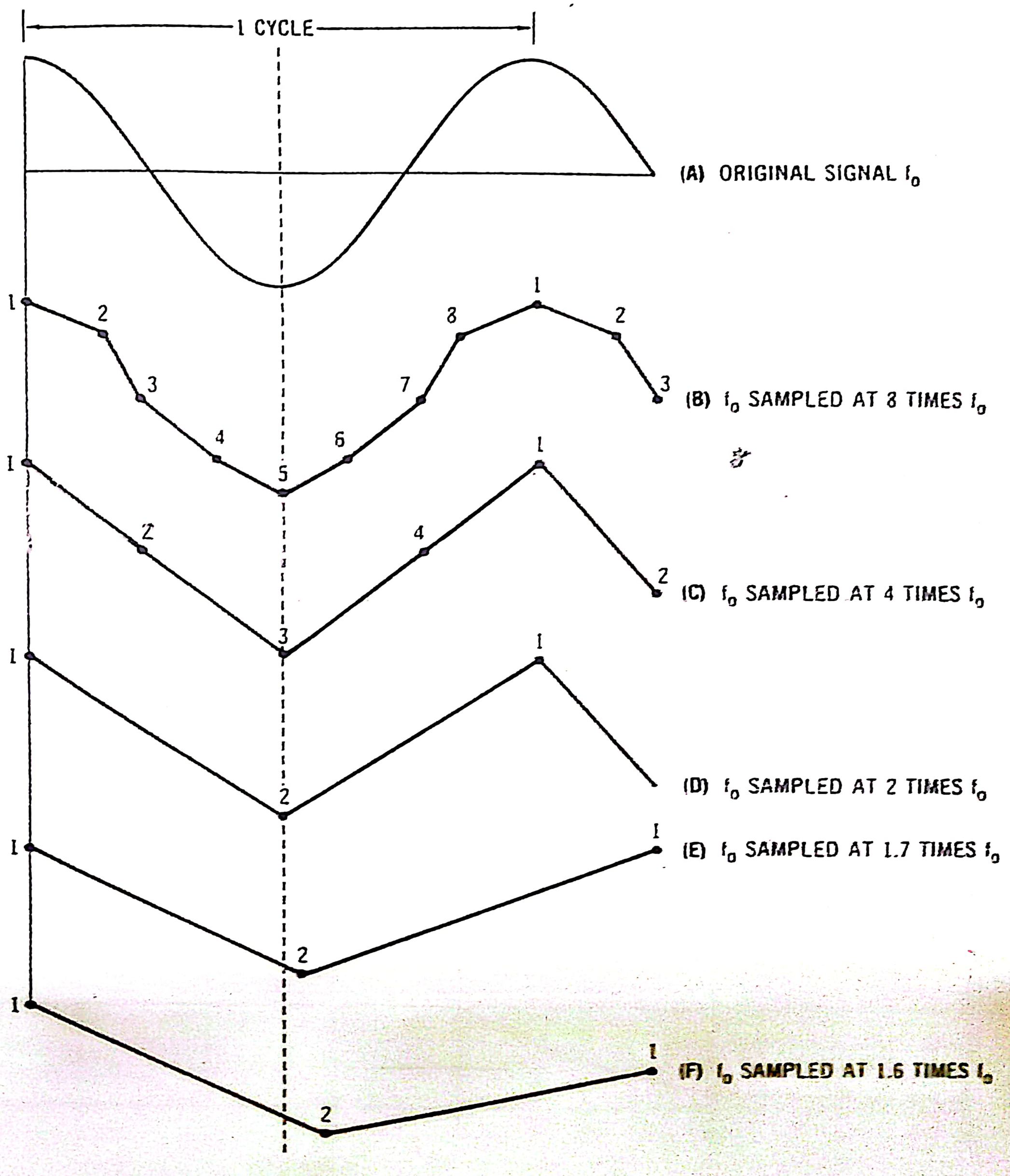


Figure 4-4.
The sampling frequency problem.

quency. It closely follows the original shape and thus yields considerable information about the top waveform. The next two waveforms in Figs. 4-4C and 4-4D are the sampled representations of the original signal at four times and two times the frequency f_0, respectively. Each of these triangular waves still contains the original fundamental frequency as part

of the sampled information. The problem, which arises when sampling at less than twice the original frequency, is shown in the remaining waveforms of the figure. The original waveform when sampled at 1.7 times the original frequency (Fig. 4-4E) is the same triangular waveform as in Fig. 4-4D *but* the period of the sampled data is beginning to change. The same frequency distortion occurs to a greater extent in Fig. 4-4F as the sample rate drops to 1.6 times the original frequency. As the sampling frequency and original frequency move closer to each other, the resultant sampled data continually moves down in frequency until they are equal (f_o = the sampling frequency). At this condition, the sampled output is a dc value sampled once each period, which produces a steady nonchanging output.

The resultant effect of the sampling problem described by Nyquist is that as the original signal exceeds one-half of the sampling frequency, an artifact frequency is generated equal to the sampling frequency (f_s) minus the original frequency (f_o), which lies *in the passband* of the original frequency. Once it has been generated within the same passband as the desired signal, it may no longer be filtered out.

As a precaution to prevent the previously described undesirable condition from occurring, the sampling rate should always be held at a frequency at least twice that expected within the analog data to be sampled. For instance, if a speech spectrum cutoff frequency of 3 kHz is desired, then the sampling frequency should at least be 6.5 to 7 kHz. The higher that the sampling frequency is placed above the original frequency, the fewer *image* frequencies will be generated within the desired speech spectrum. It is also necessary to impose rather strict conditions upon the speech spectrum to ensure that it never reaches more than one-half the sampling frequency. This may be ensured by the use of a rather fast drop off low-pass filter with the passband edge near the upper desired speech frequencies (in the example: 3 kHz).

A physical realization of one possible low-pass filter circuit is given in Fig. 4-5. The 3-pole filter is a Chebyshev design having a 0.5-dB passband ripple. The rolloff rate at 3 kHz is 18 dB per octave (one octave is a doubling of frequency). Thus, if the sample rate is 6 or 7 kHz, then the amplitude response of this filter is down at least 13 to 20 dB. The frequency response of the filter may be visualized by plotting the output of the filter for a fixed input voltage versus frequency. Fig. 4-6 illustrates the performance of the filter in Fig. 4-5 versus frequency. Notice in this plot that if the speech signal is above 3 kHz, then the resultant image amplitudes within the speech passband will be reduced by the filter response level. Ideally, the cutoff slope should be almost vertical at 3 kHz; however, this is very difficult and expensive to achieve with physical filtering systems. Switched-capacitor type filters may approximate a steep cutoff slope with a simple filter design but their cost is rather high for the increased advantages.

Another consideration in the speech digitization process is the reso-

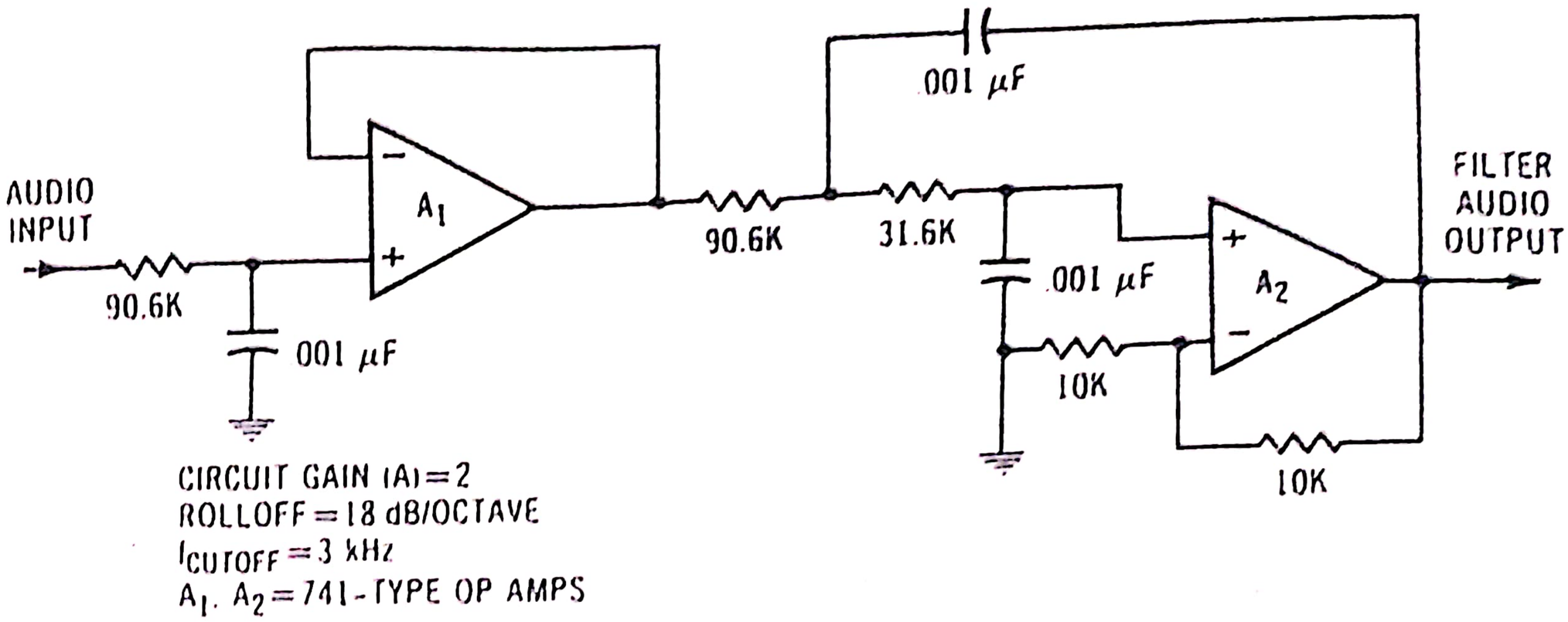


Figure 4-5.
A 3-pole Chebychev low-pass filter with 3 kHz cutoff.

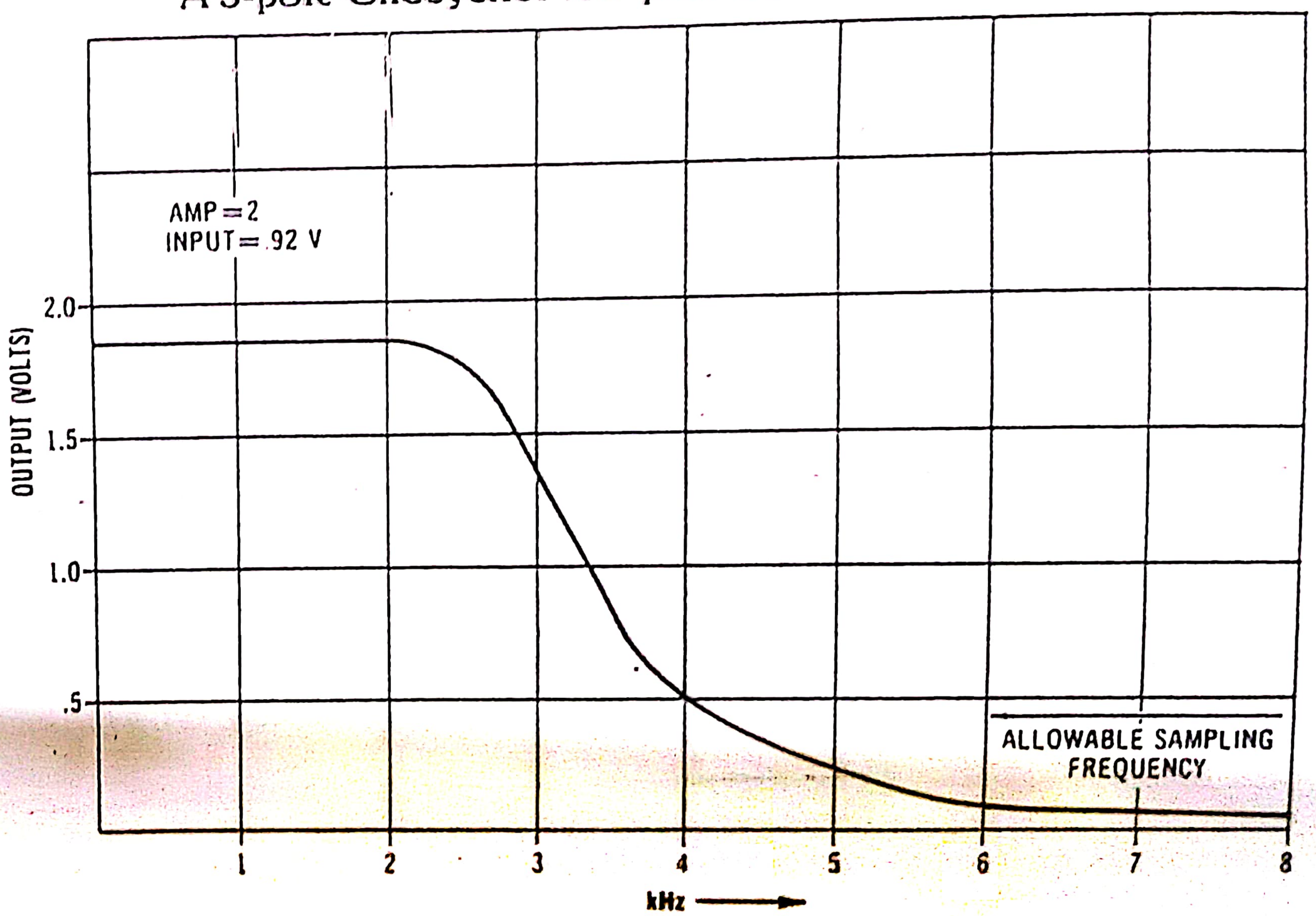


Figure 4-6.
A 3-pole 3 kHz low-pass filter response.

lution of the analog-to-digital converter. The previous figures show an N-bit converter being used to digitize the analog values with no indication of exactly how many bits are being converted. The total number is really left up to you; however, 8 bits is probably the most convenient. This adds a feeling of comfort with 8-bit microprocessors because each

sample corresponds to one byte. Let me issue a *caveat* here about this normal inclination. While an 8-bit digitized signal will represent 255 levels of possible sampled signal amplitudes—and should be perfectly acceptable for normal speech—the dynamic range of the speech may present problems. Think back to the formant frequency tables in Chapter 2. In Table 2-4 from that chapter, the ratios of the formant frequencies are given, with some third formants being approximately 40 dB below the first formant frequency. This is 100 times less than the maximum formant amplitude. What this means, in the sampling and digitizing process, is that the third formant component of speech will be digitized only with a 2.5-bit resolution. This is relatively crude to say the least! Some of the fricative sounds which are very soft and weak in amplitude will have the same problem when digitized with an 8-bit resolution. For this reason, you should consider a digitization resolution of at least 12 bits to encompass the entire dynamic range of speech. Although it means that you must carry an extra 4 bits through an 8-bit computer (no problem for a 16-bit processor), you have at your disposal 4,096 levels of digitization. Accordingly, even the smallest levels of speech that may be expected will be digitized with a 40- to 50-level resolution. This relatively small concession in cost (the 12-bit over the 8-bit a/d converter) will yield considerable improvement in the digitization of the softer sounds of speech.

Although it may sound rather complicated, the process of acquiring the digital equivalent of a speech waveform is relatively simple. If the concepts of sampling rate and presampling filters are closely observed, there should be no problems. Low-cost sample-and-hold amplifiers like the National LM-398N sell for around $4.00 each and provide perfectly adequate sampling capabilities for speech recognition. Analog-to-digital converters with 12 bits of resolution are rather expensive; however, National Semiconductor also makes a relatively low-cost 12-bit converter with a conversion rate allowing up to 10-kHz sample frequencies. This converter, the ADC1210, sells for around $50.00 but provides very high-resolution speech input. Remember, the cost of the components that you put into the system normally determine the quality and accuracy of the resultant speech recognition capabilities.

Delta Modulation Encoding

This is another form of directly digitizing speech input, but its use in speech recognition systems is rare. It is mentioned here primarily to cover the field of direct signal acquisition and because there may be some creative ways of using this simplified speech input method for speech recognition.

Basically, delta modulation encoding is a form of digitization of an analog signal which retains only the information about the *change* in the signal over time. That is where the process got its name. The *delta*

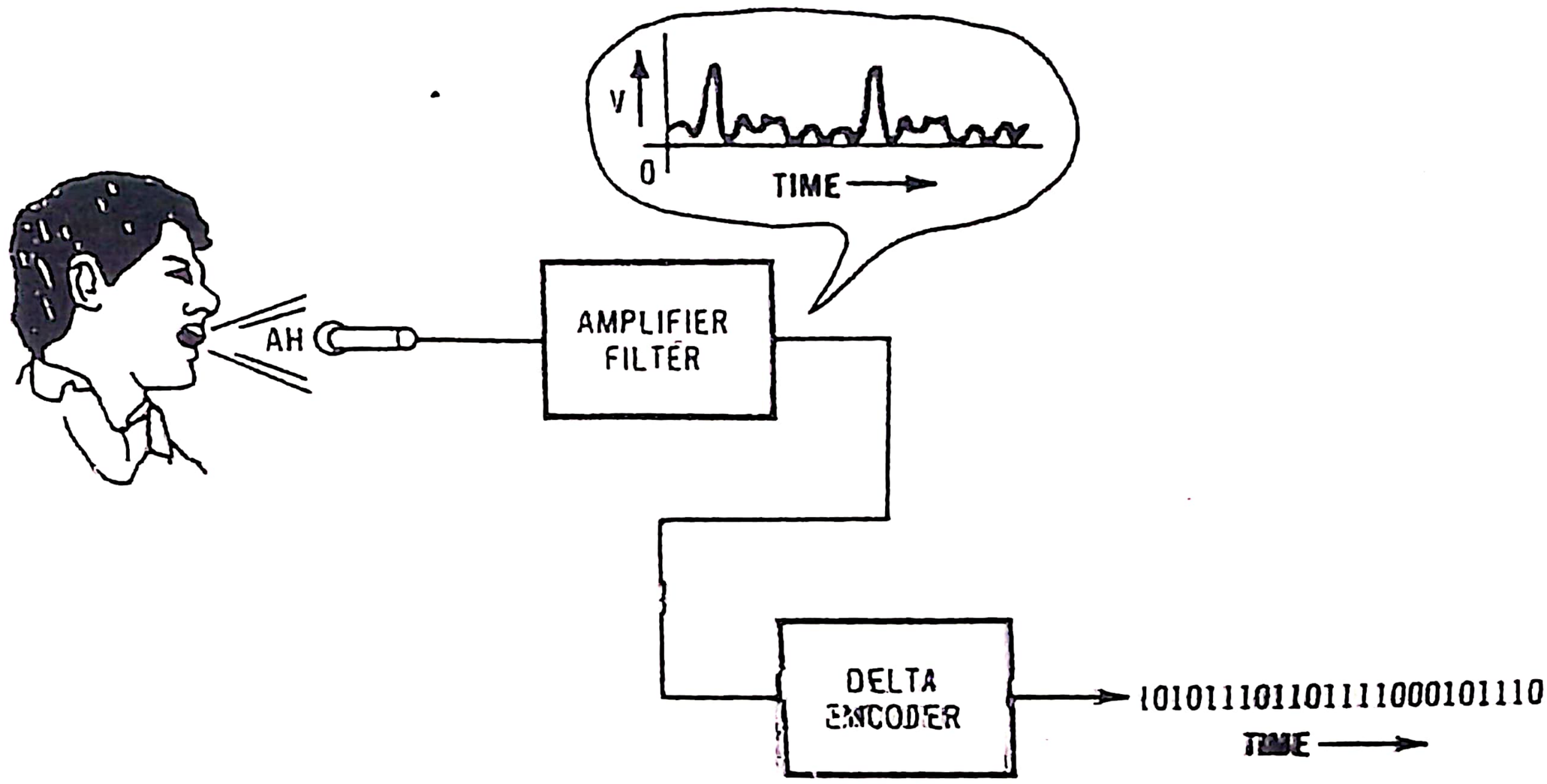


Figure 4-7.
A delta encoding system.

refers to a difference between samples. As a signal acquisition process, it is quite a bit simpler than the previous waveform encoding method, particularly when it comes to circuitry. The block diagram of a delta encoding circuit is shown in Fig. 4-7. The system is relatively simple in that it consists of an amplifier/filter to limit the speech response, followed by a delta encoding circuit. At each sample time, the encoder outputs a *single bit* to indicate whether the signal has increased, decreased, or stayed the same since the last sample. Given any point of the sample process, the computer cannot tell what actual value is being digitized. It only knows that the signal is changing from a previous value. As it turns out, this is a perfectly adequate procedure for digitizing speech. Since we are concerned primarily with the amplitude changes in time, we have no particular need to know the absolute voltage level at any sample.

The advantage to this type of encoding in addition to simplified circuitry is the need for only a single input bit to the computer. The disadvantages lie in the increased sample rate needed to adequately digitize high frequency signals. An example of how a speech signal can be digitized using only one bit is shown in Fig. 4-8. The waveform at the top represents an input signal that might come from a microphone. The center waveform is the output of a delta modulation encoder representing only the change information within the incomming signal. Notice that at the peaks of the cosine wave (where the signal goes through a relatively slow change), the delta data must alternate to produce the leveled output shown at the bottom of the figure. The correspondence between the "one" bit and the "zero" bit is very simple and direct. When the delta data is high, it corresponds to an increasing slope of

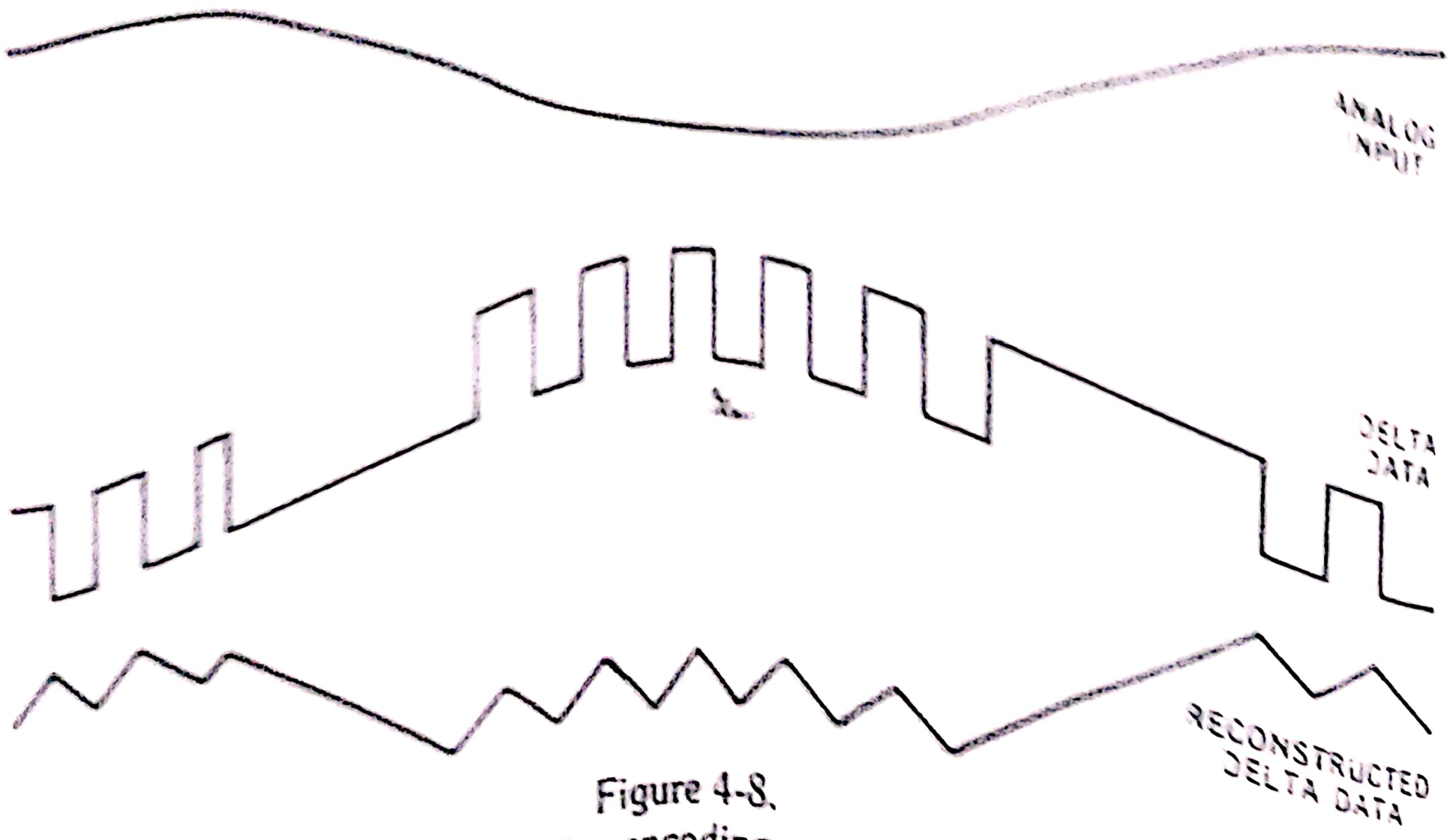


Figure 4-8.
The delta encoding process.

constant value. When it is low, it indicates a decreasing constant slope equal to the magnitude of the rising value. Thus, when the delta signal alternates, it represents a stable nonchanging incoming signal level.

As I mentioned earlier, the delta encoding process yields no real advantages and possibly even has disadvantages for use within speech recognition systems. However, the absolute waveform value may be recreated within the computer by simply adding or subtracting a fixed number from some arbitrary starting point for each clocked bit change. For example, if each clocked delta bit represents an incoming voltage change of 0.1 volt per millisecond, then a computer monitoring the delta line may internally add to a starting number (say 128) the value of 1 for each high delta data bit. Each low delta data bit would therefore correspond to a subtraction in the original 1. The resultant internal computer number after each addition or subtraction would be a relative value corresponding to the incoming signal amplitude.

As you can see, the idea of delta encoding is mechanicallly simple, while its concept of operation is somewhat more difficult than direct waveform digitization. There is a slight advantage in transmission bandwidth in using this method (approximately 3 to 6 dB), but since it requires more diddling with the numbers once they are in the computer, you will be spending more time working with the details of acquisition and less with those of actual recognition methods. There is a way that

allows much simpler signal acquisition and even simplifies the computer analysis of acquired speech. Its hardware realization is more complicated, but as you will see in the next section, it takes a large burden off of the computer in processing the speech signal.

Spectral Signal Acquisition

Spectral signal acquisition can involve analog, digital, and mathematical filters.

Analog Filtering

Having seen the complexities of direct waveform encoding of speech, you might wonder if there aren't some simpler methods that involve acquiring the frequencies of speech directly, before digitization. This is certainly an acceptable method of acquiring speech for speech recognition systems, and, in fact, simplifies the computer's task of processing spectral data. How does it work? Well, assume that a fairly large bank of selective filters is placed between the microphone and the computer to limit the frequency range in each digitized signal. The diagram of such a spectral acquisition system that might be used in front of a speech processor is shown in Fig. 4-9. Does it look familiar? It should. It is very similar to the spectrograph described in Chapter 2. Using the same technique of segmenting speech into frequency bands, the computer may digitize incoming signals more slowly but with more information carried during each sample.

In the example, there are 12 frequency bands which subdivide the vocal spectrum. Following each bandpass filter is a rectifier circuit which does nothing more than produce a dc voltage corresponding to the amplitude in that frequency band. Following each rectifier there is a smoothing filter capacitor which ensures that the output changes from each rectifier vary slowly with time. This allows a slower signal sample rate because we are no longer trying to directly digitize speech, but instead we are capturing the frequency variations in speech. The smoothing filter has to be no faster than our vocal tracts can change their shapes, thus giving a time window on the order of 10 to 20 milliseconds for each complete digitization sequence.

During the acquisition sequence, the computer must sequentially select, using the analog multiplexer, each filter output and in turn digitize that value for input. Then, internal to the computer are a bank of 12 registers used to store the frequency components of speech as they are digitized. The process is really quite simple. It gives the computer information that may be used to directly access possible phoneme tables or stored speech patterns of sounds and frequencies.

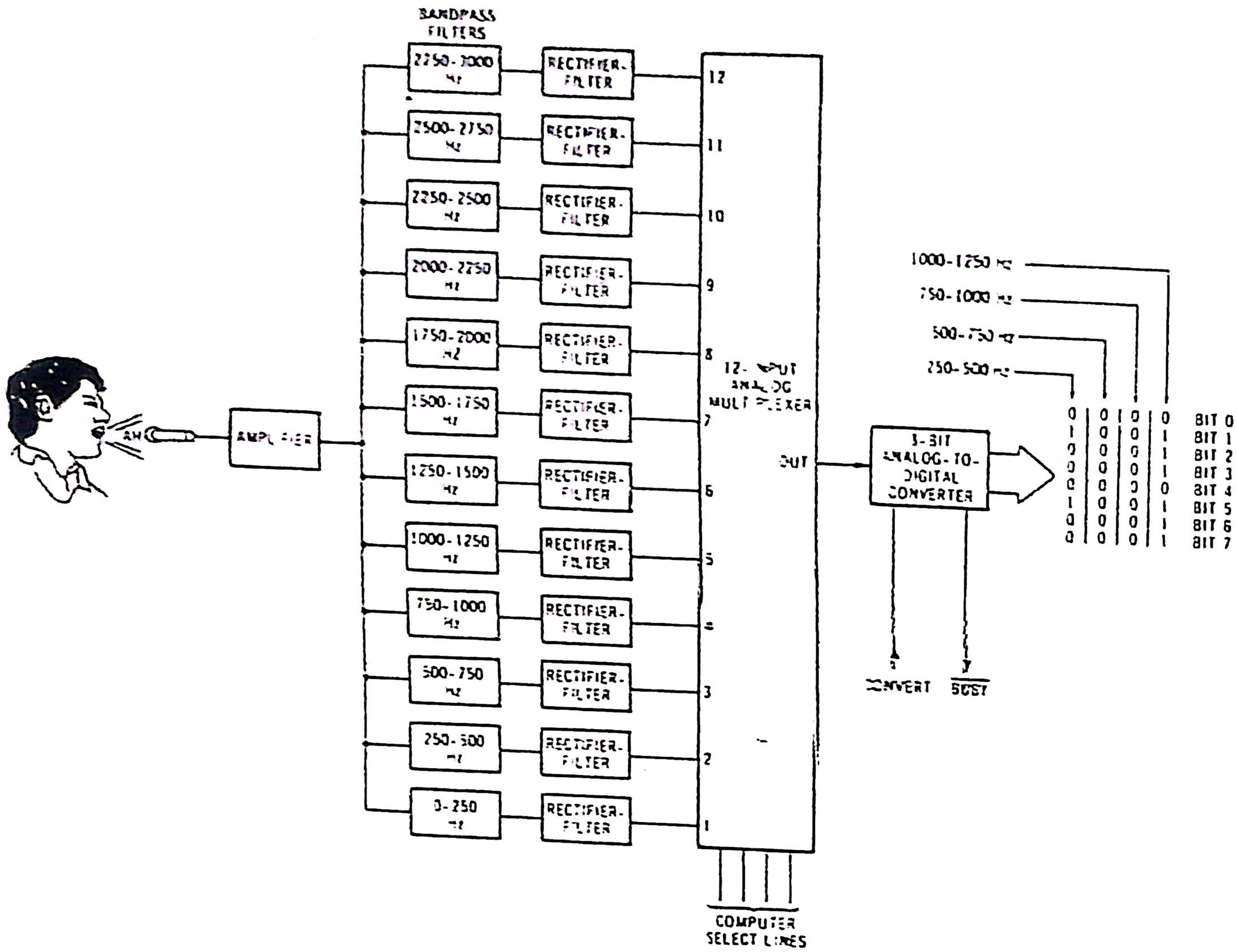


Figure 4-9.
Spectral acquisition of speech.

Since the task of digitizing all the filter outputs must be accomplished in a 10- to 20-ms window, the 8-bit analog-to-digital converter has considerable time to spend with each rectifier-filter output. Of course, the computer software is significantly simplified because the speech spectral information is digitized directly. The trade-off in hardware costs versus software complexity and running time must be considered depending on your own desires and pocketbook. For a look at the type of hardware being discussed, Fig. 4-10 shows a typical filter-rectifier combination. Remember that this circuitry has to be duplicated, for the example, *12 times*. Even though the amplifiers may not be expensive, the resistor-capacitor combinations must be highly precise to ensure nonoverlapping frequency coverage. Actually, if the filter system is to be accurate, the cut-off bands of each filter should also be very sharp and nonoverlapping. This would require extremely costly and complex bandpass filters compared to the one shown in Fig. 4-10. If you would like to experiment with different bandwidths or design your own bandpass filters, the BASIC program in Listing 4-1 gives the values for the filter circuit in Fig. 4-10 over the speech band.

FIG. 4-9 FILTER FREQUENCY RANGE	R_X VALUE
0-250 Hz	96.6K
250-500 Hz	14.3K
500-750 Hz	4.56K
750-1000 Hz	2.27K
1000-1250 Hz	1.36K
1250-1500 Hz	904 Ω
1500-1750 Hz	645 Ω
1750-2000 Hz	483 Ω
2000-2250 Hz	375 Ω
2250-2500 Hz	300 Ω
2500-2750 Hz	246 Ω
2750-3000 Hz	205 Ω

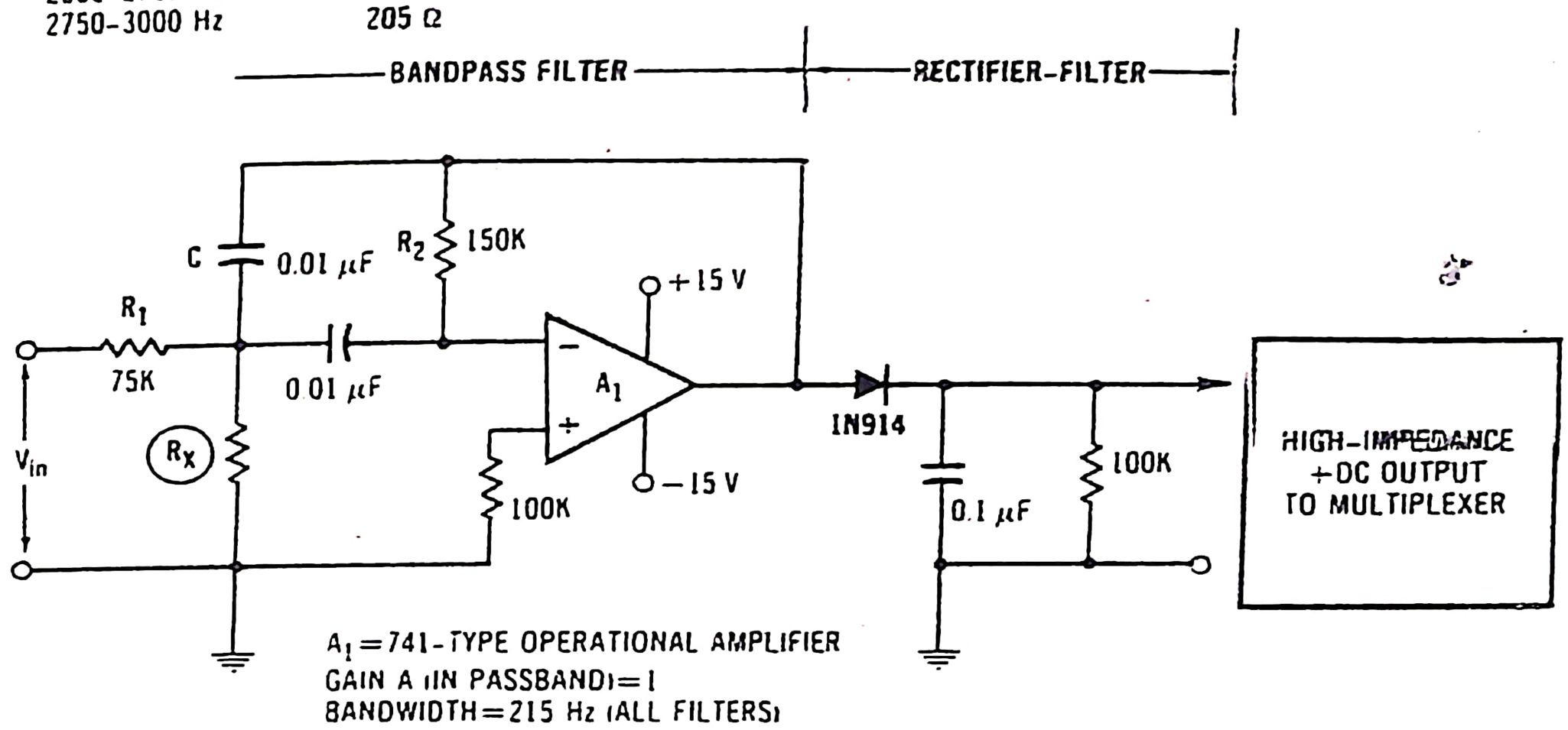


Figure 4-10.
Filter circuit for implementing spectral acquisition system.

Listing 4-1.
Bandpass Filter Values.

```
5 REM THIS PROGRAM COMPUTES VALUES FOR A BANDPASS FILTER
7 REM FROM FIG. 4-10 IN ELECTRONICALLY HEARING #22173
9 REM AND FIG. D-11 IN ELECTRONICALLY SPEAKING #21947
10 REM
11 REM CHANGE VALUES FOR C, R1, AND R2 TO ADJUST
12 REM FILTER BANDWIDTH AND GAIN.
13 REM
14 REM C=.01 UF, R1= 75K OHMS, R2= 150K OHMS
15 C=.01E-6:R1=75000:R2=150000
17 REM
18 PRINT!"FOR C=";C,"R1=";R1,"R2="R2
19 PRINT!:PRINT!
20 FOR FRQ=125 TO 3000 STEP 250
25 IFFRQ<200 THEN FRQ=200
30 RX=1/(R2*4*(3.14159^2)*(C^2)*(FRQ^2)-(1/R1))
40 PRINT!"GAIN=";R2/(2*R1),"BW=";INT(1/(3.14159*C*R2));"HZ","FR
   Q=";FRQ,"RX=";RX;"OHMS"
45 IF FRQ=200 THEN FRQ=125
50 NEXT FRQ
60 PRINT!:PRINT!:PRINT!:PRINT!"END":PRINT!:PRINT!:PRINT!
```

There is an alternative method that may be used if the analog hardware route is undesirable. This alternative utilizes the relatively new technology of digital switched-capacitor filters.

Digital Filtering

This nomenclature may be slightly misleading because the filters to be described within this section are of the switched-capacitor type, and are basically analog filters internally. However, because these devices require an external clock frequency input that sets the filter center frequency, they should be classed as truly digital filters. They are a result of a fairly recent major innovation in semiconductor technology to construct physically realizable monolithic floating capacitors with closely controlled capacitance ratios. They appear to have been first developed in the laboratories at the University of California at Berkeley and were exactly that: laboratory curiosities. Once the monolithic technology and integrated circuit construction techniques were compatible with the circuit designs of switched-capacitor filters, they became economically feasible.

The operating principle behind a switched-capacitor filter relies primarily on a technique called charge distribution. An example of a very simple section of a switched capacitor filter is shown in Fig. 4-11. In this diagram, the switch is used to connect the capacitor C_s between either voltage V1 or voltage V2. If the switch moves from position 1 to position 2, the capacitor voltage will change from V1 to V2. The change in the charge on the capacitor (ΔQ) is described by $C_s * (V1 - V2)$. As the switch begins to move faster and faster and the same amount of charge is moved for each repetition, a net current flows in the direction from V1 to V2. The current in this case may be mathematically represented by an equation that combines the two voltage values, the capacitance value, and the switching frequency F_s:

$$I_s = \Delta Q * F_s = C * (V1 - V2) * F_s$$

If we now allow the switching frequency F_s to increase beyond any possible variation in time of either V1 or V2, then the circuit may be

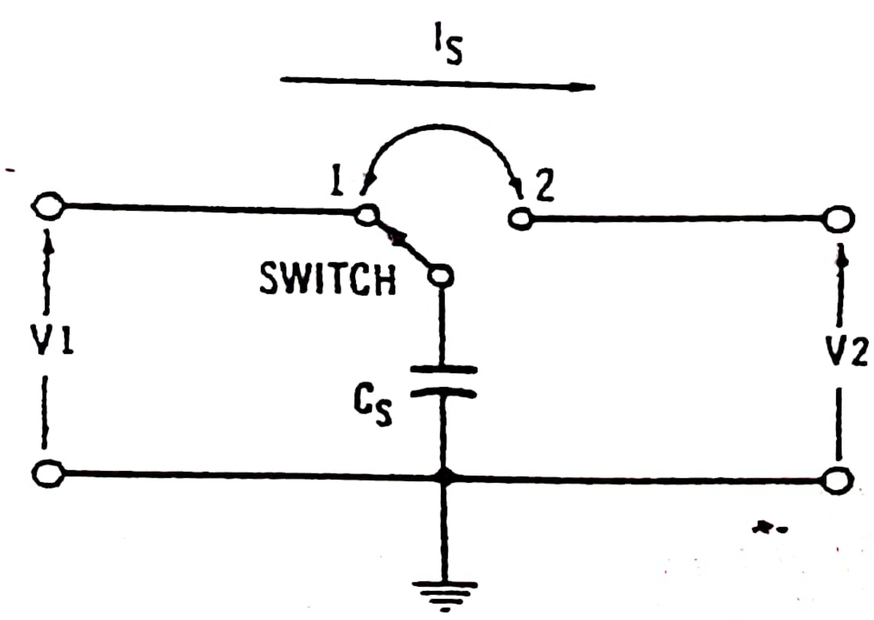


Figure 4-11.
A simple switched-capacitor circuit.

viewed as a simple resistance having a given current flow for a given voltage difference. If we take the equation:

$$I = (V1 - V2)/R$$

and compare it with the previous equation, we see that we can also then equate the two terms:

$$1/R = C_s * F_s$$

The analogy can be carried further to the circuits of Fig. 4-12 to illustrate how the switched capacitor may be used to replace a resistance in a single-stage RC filter. The two parts of this figure are identical in function, *if* the switching frequency F_s is selected to equal an identical current flow that occurs in resistance R for the same input voltage. What is significant in this circuit is that the resistance no longer has to be placed on the monolithic integrated circuit. Since the capacitance ratio determines the RC constant of Fig. 4-12B, the circuit is much easier to produce. The mechanism also allows the filter to be tuned by simply changing the switching frequency F_s. Thus, if a large group of these filters is utilized in a complex filter structure, the entire passband of the filter system may be readily changed by simply varying the input switching frequency. Remember, though, that the key to the operation of this circuit is that the switching frequency F_s must be much higher (on the order of 50 to 100 times) than any expected input signal variation.

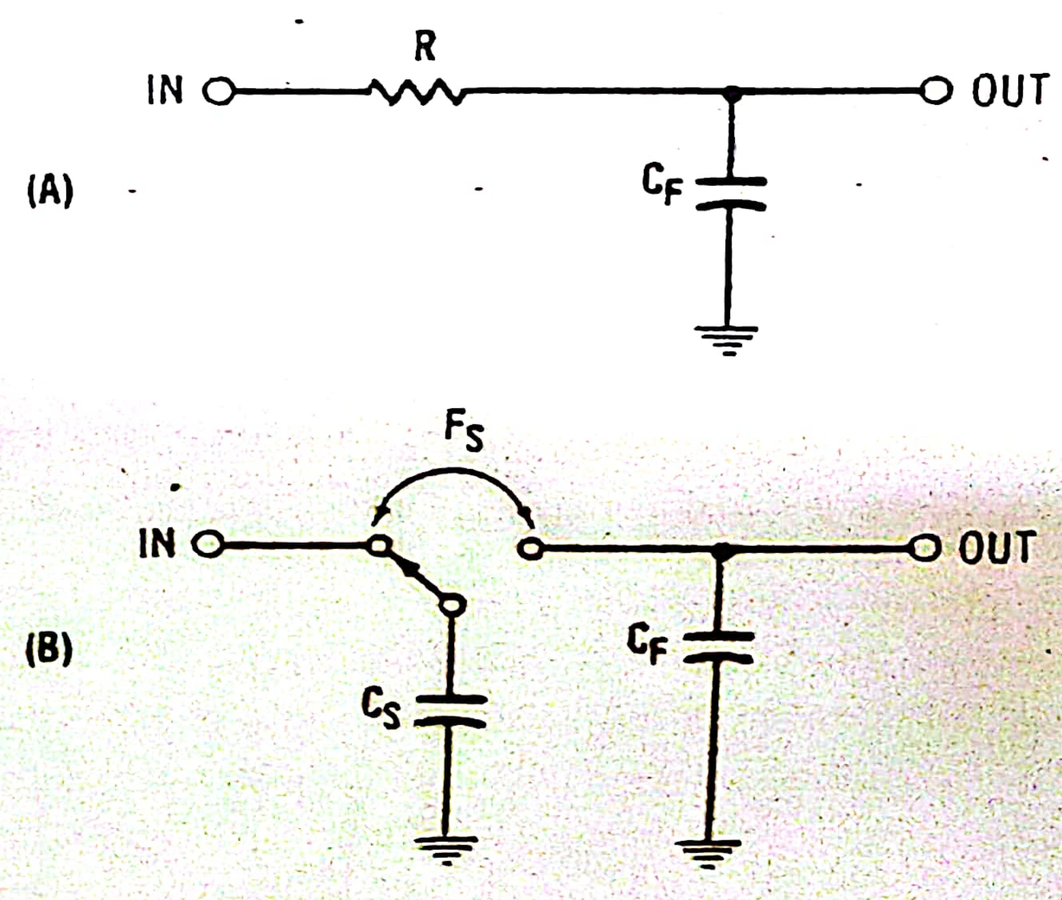


Figure 4-12.
A single-section filter comparison.

The net results of the switched capacitor filter development is the rather inexpensive availability of extremely complex filters in very small integrated circuit packages. Several companies are now producing these filters, which may be used for either bandpass filtering of spectral input, or low-pass, presampling filtering. Motorola produces a fifth-order elliptic low-pass switched-capacitor filter fifth called the MC-145414. Each of these filter chips contains two fifth-order filters, which may be cascaded to produce a tenth-order elliptic low-pass filter with extremely

sharp cutoff at the filter frequency. In the Motorola circuit, the sampling frequency is approximately 40 times greater than the signal frequency.

Another manufacturer, American Microsystems, Inc., produces a seventh-order elliptical filter that is programmable through either the adjustment of the clock frequency or the input of a binary code. The S3528 switched-capacitor filter provides 50-dB attenuation of the input signal at 1.3 times the cutoff frequency of the filter.

Another important manufacturer of switched-capacitor filters is Reticon, Inc., which produces a family of filter chips, including low pass, bandpass, and high pass. These may be used to construct an entire spectrum analysis system and can simplify the task of building a spectral acquisition system similar to that shown in Fig. 4-9. Reference to these manufacturers is given in the appendix.

Mathematical Filtering

This type of signal filtering is more commonly referred to as true digital filtering. However, because in most cases these systems first digitize the analog input and then mathematically manipulate the data according to sampled data equations, they may also be referred to as mathematical filters.

The technique of mathematical filtering is not limited to integrated circuit technology. With proper equations and data manipulations, digitized data in any computer may be mathematically filtered to present band-limited information for further speech processing. The general theory behind digital filters is somewhat complex. For instance, a representation of a first-order mathematical filter is shown in Fig. 4-13. The method of operation is based upon three basic computations. For a

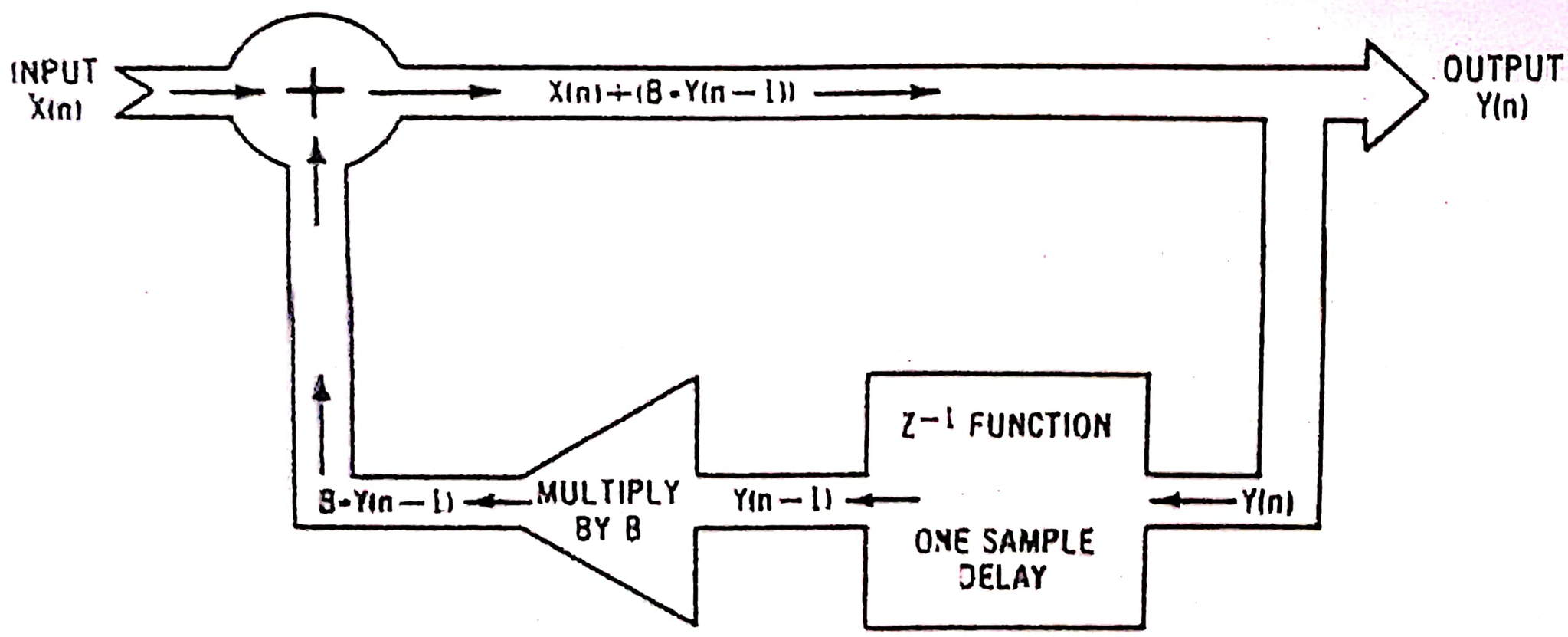


Figure 4-13.
A first-order mathematical recursive filter.

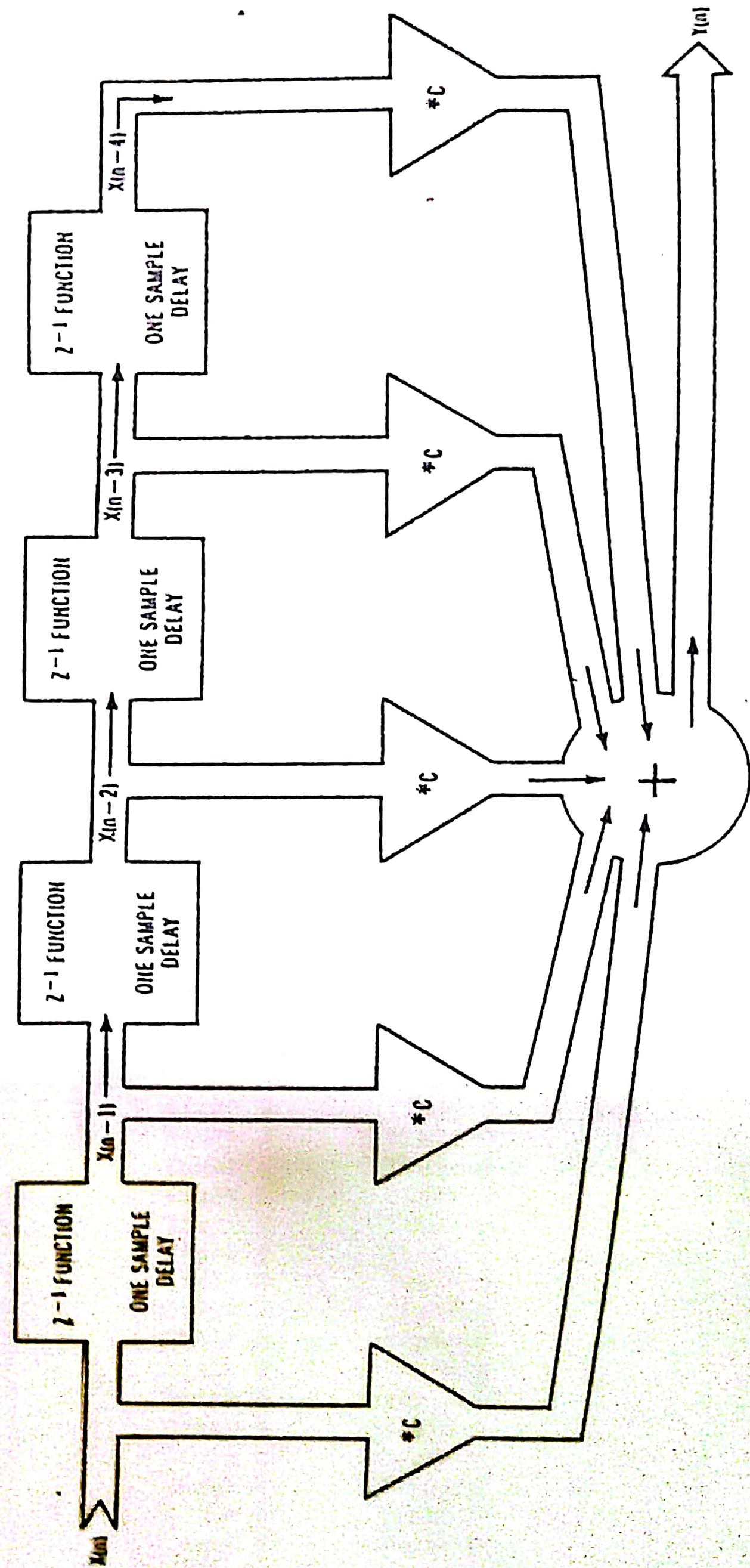


Figure 4-14.
A nonrecursive mathematical filter.

given input, X(n), the filter's output is Y(n). What occurs between the input and output is a mathematical filter process of the *first order*. This is the *simplest* type of filter in the mathematical filter family.

There are basically two types of computational filters in digital filter theory: the recursive and the nonrecursive filter. If a filter has the property of recursion, then its output is based upon previous samples of its own output. In other words, there is some type of feedback within the system. The filter in Fig. 4-13 is recursive. It is also called an IIR (Infinite Impulse Response) filter. A nonrecursive filter, on the other hand, generates an output that is based solely upon the input samples and *not* on previous values of the filter output. An example of a nonrecursive filter is shown in Fig. 4-14. This one is an FIR (Finite Impulse Response) filter. Its operation is very similar to the previous recursive filter; however, it is unconditionally stable (it has finite response) and cannot go into a mode of oscillation like the recursive filter. (That is where the name "Infinite Impulse" originates.)

The operation of these filters is based upon a series of sampled data taken in order beginning with the first through the nth. The square boxes in each of the two previous figures represent a one-sample delay. This is nothing more than backing up one previous sample from the one currently under consideration. If a series of samples is stored in memory, then the computer performs the Z^{-1} function by simply indexing one memory location backwards to pickup the previous Z^{-1} sample for each block. The multiplication triangle represents a process where the incoming number is simply multiplied by the constant value within

Listing 4-2.
First-order Digital Filter Simulation.

```
5 REM BASIC PROGRAM TO SIMULATE FIRST-ORDER DIGITAL FILTER
6 REM -- RECURSIVE TYPE --
10 DIM XM(30),YM(30)
15 INPUT "B=",B
20 FOR I=0 TO 26
30 READ XM(I)
40 NEXT I
45 PRINT!"1st-Order Recursive Filter Response":PRINT!"for B=";B
   ;" in Fig. 4-13."
46 PRINT!:PRINT!
47 PRINT!" Input","Output"
48 PRINT!"----------------------------"
50 REM NOW FILTER XM
60 FOR N=0 TO 30
65 IF N=0 THEN YM(N)=XM(N):GOTO80
70 YM(N)=XM(N)+B*YM(N-1)
80 PRINT!"X(n)=";XM(N),"Y(n)=";INT(YM(N))
90 NEXT N
100 PRINT!:PRINT!:PRINT!
150 DATA 0,0,10,10,10,10,10,10,10,10,10,10,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0
```

that block (B in the case of Fig. 4-13, C in the case of Fig. 4-14). Finally, the "round" addition shapes add the incoming numbers together and then produce a single summed output.

The mathematical functions of Fig. 4-13 may be simulated in a computer using a BASIC program as shown in Listing 4-2. The rather simple mechanism of the computation should indicate that the filtering calculations are relatively straightforward and understandable. If we allow this program to run with varying values for B, we have the resultant outputs given in Figs. 4-15 through 4-18. Each of the response values Y(n) represents the filter response to a square pulse input of amplitude 10 with a length of 10 sample periods. In each of the figures, the filter architecture remains the same and only the multiplying coefficient B changes. Notice that as B increases from 0 through 0.8, the response time of the filter slows considerably. If we allow B to increase to a value of 1 or greater, then the filter becomes unstable and has a continually increasing output whether the input is present or not. Aha! The Infinite Response!

Input	Output
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 10
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0

Figure 4-15.
First-order recursive filter response for B = 0 in Fig. 4-13.

Input	Output
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 13
X(n)= 10	Y(n)= 13
X(n)= 10	Y(n)= 14
X(n)= 10	Y(n)= 14
X(n)= 10	Y(n)= 14
X(n)= 10	Y(n)= 14
X(n)= 10	Y(n)= 14
X(n)= 10	Y(n)= 14
X(n)= 0	Y(n)= 4
X(n)= 0	Y(n)= 1
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0

Figure 4-16.
First-order recursive filter response for B = 0.3 in Fig. 4-13.

The second type of filter shown in Fig. 4-14, being nonrecursive, does not have the same problem. A BASIC program simulation of this filter is given in Listing 4-3. If we allow this program to run with a C value of 0.2, then we have a filter response computed as shown in Fig. 4-19. This computed filter response like the previous filter shows a time-delayed response in the Y(n) output for an X(n) input equal to a 10-sample square wave of amplitude 10. This type of filtering is very closely related to a sliding average, and the filter of Fig. 4-14 represents a 5-point sliding average along the incoming digitized data.

The cutoff frequencies for these types of filters is directly dependent upon the sampling frequency and the amount of delay induced by the filter. Many books have been written on digital and mathematical filtering methods with all of their associated complexities. The references at the end of this book include some excellent advance readings in mathematical filtering methods.

As a final word on speech signal-acquisition techniques, what you have seen in this chapter are the primary types of computer acquisition

Input	Output
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 17
X(n)= 10	Y(n)= 21
X(n)= 10	Y(n)= 25
X(n)= 10	Y(n)= 27
X(n)= 10	Y(n)= 29
X(n)= 10	Y(n)= 30
X(n)= 10	Y(n)= 31
X(n)= 10	Y(n)= 31
X(n)= 10	Y(n)= 32
X(n)= 0	Y(n)= 22
X(n)= 0	Y(n)= 15
X(n)= 0	Y(n)= 11
X(n)= 0	Y(n)= 7
X(n)= 0	Y(n)= 5
X(n)= 0	Y(n)= 3
X(n)= 0	Y(n)= 2
X(n)= 0	Y(n)= 1
X(n)= 0	Y(n)= 1
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0

Figure 4-17.
First-order recursive filter response for B = 0.7 in Fig. 4-13.

of speech. The variant types that exist are usually based upon some form of those given here. In the next chapter, we see how a variant of mathematical filtering called the frequency-domain transform technique can produce the same results as multiple mathematical filter systems. Thus, the mathematical process can simulate a spectral signal-acquisition circuit like that described in this chapter, based purely upon computational processing of the directly digitized input waveform.

Input	Output
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0
X(n)= 10	Y(n)= 10
X(n)= 10	Y(n)= 18
X(n)= 10	Y(n)= 24
X(n)= 10	Y(n)= 29
X(n)= 10	Y(n)= 33
X(n)= 10	Y(n)= 36
X(n)= 10	Y(n)= 39
X(n)= 10	Y(n)= 41
X(n)= 10	Y(n)= 43
X(n)= 10	Y(n)= 44
X(n)= 0	Y(n)= 35
X(n)= 0	Y(n)= 28
X(n)= 0	Y(n)= 22
X(n)= 0	Y(n)= 18
X(n)= 0	Y(n)= 14
X(n)= 0	Y(n)= 11
X(n)= 0	Y(n)= 9
X(n)= 0	Y(n)= 7
X(n)= 0	Y(n)= 5
X(n)= 0	Y(n)= 4
X(n)= 0	Y(n)= 3
X(n)= 0	Y(n)= 3
X(n)= 0	Y(n)= 2
X(n)= 0	Y(n)= 1
X(n)= 0	Y(n)= 1
X(n)= 0	Y(n)= 1
X(n)= 0	Y(n)= 1
X(n)= 0	Y(n)= 0
X(n)= 0	Y(n)= 0

Figure 4-18:
First-order recursive filter response for B = 0.8 in Fig. 4-13.

Listing 4-3.
Five-Stage Transversal Simulation.

```
5 REM BASIC PROGRAM TO SIMULATE 5 STAGE TRANSVERSAL
6 REM FILTER -- NON-RECURSIVE TYPE --
10 DIM XN(40),YN(40)
15 INPUT "C=",C
18 REM OFFSET DATA TO KEEP POSITIVE SUBSCRIPTS
19 REM XN(0) THROUGH XN(3)=0
20 FOR I=4 TO 33
30 READ XN(I)
40 NEXT I
45 PRINT!"Filter Response in Fig. 4-14 ":PRINT!"     for C=";C
46 PRINT!:PRINT!
47 PRINT!" Input","Output"
48 PRINT!"----------------------------"
50 REM NOW FILTER XN
59 REM SINCE SUBSCRIPTS CAN'T BE NEGATIVE OFFSET XN AND YN BY 4
60 FOR N=4 TO 33
63 YN(N)=(XN(N)*C)+(XN(N-1)*C)+(XN(N-2)*C)+(XN(N-3)*C)+(XN(N-4)
   *C)
80 PRINT!"X(n)=";XN(N),"Y(n)=";INT(YN(N))
90 NEXT N
100 PRINT!:PRINT!:PRINT!
150 DATA 0,0,0,0,0,10,10,10,10,10,10,10,10,10,10,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0
```