"THE DESIGN OF A PROGRAMMABLE REAL TIME ADM VOICE PROCESSOR

AND

THE FORMULATION OF THE AUTOCORRELATION FUNCTION FOR THE LDM ALGORITHM"

by

TUVIA APELEWICZ

A dissertation submitted to the Graduate Faculty in Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1978

This manuscript has been read and accepted for the Graduate Faculty in Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

9/14/78
date

Chairman of Examining Committee

9/14/78
date

Executive Office

Prof. D. L. Schilling (mentor)

Dean R. B. Marsten

Prof. R. Mekel

Dr. J. Salz

Prof. H. Taub

Supervisory Committee

Abstract

This dissertation concerns itself with the design of a programmable real time adaptive delta modulation voice processor and the formulation of the autocorrelation function of the linear delta modulation algorithm.

Design requirements for the voice processor are established. A specific processor configuration is chosen and an instruction set is developed. Several delta modulation encoder algorithms and a programmable decoder capable of receiving them are implemented. This, establishes compatability and enables communication between different delta modulation systems.

A Markov chain approach is used in obtaining the autocorrelation function of the linear delta modulation system. An algorithm, to obtain the transition probabilities matrix and the steady state level probabilities, is developed for the zero-order and first-order Markov cases. This enables the formulation of the autocorrelation function of the LDM estimate signal.

Acknowledgment

I wish to express my deep gratitude to Professor Donald L. Schilling for his interest, encouragement and valuable guidance during the progress of this research. The many technical discussions and his constructive suggestions are most appreciated.

I would like to thank all the members of my Doctoral committee; Dean Richard B. Marsten, Dr. Jack Salz (Bell Laboratories), Professors Herbert Taub and Ralph Mekel for the time and effort each has taken to read and constructively critisize this dissertation.

In addition, I would like to thank all my colleagues in room T502F, especially Professor Chaim Ziegler, Mr. Donald Ucci and Mr. Norman Schienberg for their continuous support and friendship, and Mr. Dave Pressman for a job well done in the physical layout of the Voice Processor.

The excellent typing of this dissertation was performed by Ms. Jackie Murdock. Her efforts and professionalism in helping me meet my deadline is deeply appreciated.

Last but not least, I would like to thank my wife, Ariela, who helped me see through the difficult periods that arose from time to time.

The research contained in this dissertation was partially supported by the National Aeronautics and Space Administration under Grants NAS 9-13940, NSG-5013 and NAS-5038.

Table of contents

Chapter 1

1.1 Statement of Problem

This dissertation presents the design of a programmable real time voice processor and the formulation of the auto-correlation function of the estimated signal when the Linear Delta Modulation (LDM) algorithm is employed. The problem that motivated the design of the processor was the lack of compatability and communication between different DM algorithms as well as the need for a single programmable device capable of real time testing and implementation of various DM algorithms suitable for different applications.

The philosophy behind all digital Adaptive Delta Modulation (ADM) schemes known, in particular the two shown in Figs. 1.2-5 and 1.2-10, is conceptually the same. All employ a comparator and a two level quantizer. In each system the estimated signal, $\hat{m}(t)$, is obtained by adding the computed step size to the previous estimate and the information sent at sampling time k, is always the sign of the error between the input and estimated signals at time k. The step size generator, or the algorithm controlling the formation of the new step size, on the other hand, is unique to each scheme.

However, in spite of the similarity between the different Delta Modulation (DM) schemes, one cannot use a delta modulator transmitter employing a continuous companding scheme, and a delta modulator receiver employing a syllabic companding scheme, for example, and expect a reasonable quality of the received signal. As a matter of fact one would do just as well by processing the error bit information by a band pass filter which passes speech frequencies. On the other hand, since different people may use different delta modulation schemes, there is a need for a device that will interface between the various algorithms, or equivalently, a programmable device that can be adapted to the different delta modulation systems employed.

Adaptive delta modulation systems are extremely difficult to analyze mathematically. Therefore quantitative performance of a proposed system must be measured in one of two ways: a] simulation on a computer, (not a real time operation) where the input is either a sine wave or stored speech. b] hard wired implementation of a proposed system.

Since simulations on a computer are usually done at very low sampling frequencies, to introduce some sense of reality the input is slowed down. When determining the adaptive delta modulator performance one is required to decide whether distortion is due to this non real time operation or due to the system under investigation.

The three basic quantitative performance measurements used to test delta modulation systems and to obtain their performance are: a] Linearity - that is given two independent sources, will the response of the delta modulator to the composite signal be the same as the sum of responses to the individual sources? Also, when connecting few delta modulator systems in tandem is the overall system linear? b] Idle channel noise - what is the ratio of the average power of the response when no input is present, to average power of the response to an input with maximum allowable input power? c] Sinewave SNR - what is the ratio of the average power of the response to a sine wave, to the average power of the quantization noise?

The only mathematical performance criteria applicable is the Signal-to quantization Noise Ratio (SNR). However, adaptive delta modulators when following a sine wave of a single frequency reduce to a LDM. Hence, the SNR criteria is an invalid or meaningless performance criteria. The only real way to test a system designed for voice-type input signals is to hard wire a proposed system and then perform all possible quantitative and subjective tests. However, should the proposed system turn out to be invalid then the time spent on the design, implementation and debugging is futile. It is, therefore, desirable to have a system which will simulate and interface all different adaptive delta modulation systems in real time and be programmable such that new algorithms can be tested and debugged easily.

1.2 Introduction and Historical Background

1.2-1 Linear Delta Modulation

Delta Modulation is a technique by which an analog signal can be encoded into binary digits (bits) and transmitted across a communication channel. Delta modulation encoding has been utilized for two decades and numerous contributions to its analysis and performance have been made by De Jager [1], Van de Weg [2], Zetterberg [3], and others.

A DM system is shown in Fig. 1.2-1a. When an input, $m(t)$, is applied to the DM encoder, it is compared to $\hat{m}(t)$, where $\hat{m}(t)$ is the estimate of $m(t)$. The difference, or error, signal,

$$\Delta(t) = m(t)-\hat{m}(t) \qquad (1.2\text{-}1)$$

is examined at fixed time intervals $k\tau$, where τ is the sampling period of the DM and k is a positive integer. If $\Delta(t=k\tau)$ is positive, the output of the quantizer $e(t=k\tau)$, will be 1, and if $\Delta(t=k\tau)$ is negative, $e(t=k\tau)$ will be -1. The estimator utilizes the output signal $e(t)$ to form the estimate.

A LDM system is shown in Fig. 1.2-1b. When an input $m(t)$ is applied to the LDM encoder, it is compared to $\hat{m}(t)$. The output of the one bit quantizer, or hard limiter, is 1 if $\Delta(t=k\tau)$ is greater than or equal to zero, and -1 if

$\Delta(t=k\tau)$ is less than zero. The estimate, $\hat{m}(t=k\tau)$, will be incremented (decremented) by a step of value S, if $e(t=k\tau)$ is 1 (-1). Figure 1.2-2 shows a typical DM response to an arbitrary input. At the receiver, the estimate signal is reconstructed from the transmitted bit stream e(t) in the same manner $\hat{m}(t)$ was constructed at the transmitter.

The difference wave form $\Delta(t)$ may be viewed as noise due to the quantization process and is called the quantization noise or error. The quantization error can be considered to consist of two parts: granular noise and slope overload noise. Granular noise is due to the finite step size S and slope overload noise is due to the fact that the LDM can not follow a signal whose rate of change, or slope, is larger than S/τ.

A method of overcoming the slope overload deficiency in the LDM is to compress the large amplitude levels in the input signal relative to the smaller ones prior to encoding and to expand after decoding as shown in Fig. 1.2-3. The companding - expanding action can be integrated into the DM scheme by the introduction of a non-linear network in the feedback loop. A DM with a non-linear network is known as an Adaptive Delta Modulator (ADM).

The adaptation can be "Syllabic" [4-6] with an adaptation time constant equal to several samples or several tens of samples, in the order of msec, or instantaneous [7-9] with significant step size modification

over one sampling period. In the various adaptive schemes that have been proposed, the step size is made to increase when following rapidly varying inputs, and to decrease otherwise. This reduces slope overload noise, but increases the granular noise in the form of overshoots and oscillations if a rapidly varying input suddenly becomes idle as shown in Fig. 1.2-4. However, simple forms of delayed encoding can provide very useful stabilizations. The stabilizing action results because signal anticipation provides an interpolative capability which can usefully complement the extrapolative action of predictive coding. In either case adaptation is based only on the immediate history of the ADM bit stream. The step size information can be automatically recovered at a receiver by observing the received bit information.

1.2-2 Instantaneously Companded Delta Modulation

In an Instantaneously Companded DM (ICDM), the step size varies, or adapts, according to instantaneous changes in the transmitted bit stream. An example of an ICDM is the one proposed by SONG [9], where an optimum (in the mean square sense) encoder-decoder combination, based on immediate two-bit history and a first order Markov input, is derived. To simplify the hardware implementation (the original performance equations were very complex) a high sampling rate was

assumed (i.e. correlation of 1 between adjacent samples). The resulting equations were found to be:

$$\hat{m}(k+1) = \hat{m}(k) + s(k+1)$$

$$s(k+1) = \begin{cases} 1.5|s(k)|e(k) & e(k)=e(k-1) \\ 0.5|s(k)|e(k) & e(k)\neq e(k-1) \end{cases}$$

$$e(k) = \text{sgn}[m(k)-\hat{m}(k)] \qquad (1.2\text{-}2)$$

where $\hat{m}(k+1)$ and $\hat{m}(k)$ are the next and present estimate values respectively, $s(k+1)$ and $s(k)$, are the stepsizes used to generate $\hat{m}(k+1)$ and $\hat{m}(k)$ respectively and $e(k)$ is the sign of the error at sampling time k. (Note: for simplicity of notation we have assumed τ is unity).

The ADM is shown in Fig. 1.2-5. A clock signal at time k will sample the instantaneous error $\Delta(t)$ and the sign of the error just prior to the positive transition of the clock will be stored as $e(k)$. The present sign bit $e(k)$ together with the past one $e(k-1)$ and the previous step size will form the next step size $s(k+1)$ according to the rules of Eq. (1.2-2). The approximation to the input signal at time k+1 is formed by adding the present estimate and the newly generated step size.

The response of the ADM to a step input is shown in Fig. 1.2-6. It is quite clear that the step size is increasing in an exponential manner as shown in Fig. 1.2-7. This feature makes the above coder-decoder

scheme very attractive for video coding applications since video information changes in steps. The mean square error will be a minimum if the ADM estimate level can reach the input signal level as fast as possible. On the other hand, since the envelope of voice changes slowly compared to the sampling frequency, the fast increase causes the above algorithm to be inadequate in encoding voice-like signals.

Experimental results [10] have indeed shown that video information can be encloded quite satisfactorly (20 Mbits/sec compared to 40 Mbits/sec of PCM encoding) whereas voice encoding was quite poor for same sampling frequency to signal bandwidth ratio.

To improve the voice encoding performance of the SONG ICDM, Eq. (1.2-2) was modified such that the step size did not rise as rapidly as it did. The desired algorithm was chosen to be:

$$\begin{aligned} \hat{m}(k+1) &= \hat{m}(k)+S(k+1) \\ S(k+1) &= |S(k)|e(k)+S_o e(k-1) \\ e(k) &= SGN[m(k)-\hat{m}(k)] \end{aligned} \qquad (1.2\text{-}3)$$

where S_o is a constant usually between 10 and 30 millivolts. The response of Eq.(1.2-2) to a step input is redrawn in Fig. 1.2-8 and on it superimposed the response to a step input of Eq.(1.2-3). Note that

although initially Eq. (1.2-3) rises faster, the exponential rise of Eq. (1.2-2) takes over quickly whereas Eq. (1.2-3) rises much more gracefully. Equation (1.2-2) is known as the SONG Video Mode ADM (SVMADM) and Eq. (1.2-3) as the SONG Voice Mode ADM (SVADM).

1.2-3 Syllabically Companded Delta Modulation

Incorporating the companding feature into the DM action can be achieved yet in another way. An audio waveform band limited to 2.5 kHz and amplitude limited to V_o volts peak-to-peak (usually in the order of 5 volts) has an envelope with a frequency between 25Hz and 100 Hz. In a Syllabic Companded DM (SCDM) the increase or decrease in step size follows this envelope.

It was found that the best results are achieved when changes in the step size follow an exponential decay rather than an exponential or linear increase as is the case with the ICDM. Step size generation is controlled by a single pole RC network known as the syllabic filter with a time constant between 1 and 10 msec. When the step size increases in magnitude it follows the voltage across the capacitor and when the step size decreases in magnitude it follows the voltage across the resistor.

An example of a SCDM is the Continuous Variable Slope DM (CVSD). Figures 1.2-9 and 1.2-10 illustrate the analog and digital versions of the CVSD. An error signal is formed by comparing the input signal with the latest approximation to it. The error signal is sampled at a fixed rate and the sign of the error at time t=k is recorded as e(k). Based upon the present and the past two e(k)'s slope overload is detected when all three bits are the same. Therefore the step size increases in the same manner as a voltage changes across a charging capacitor in a single pole R-C network (hence the term charge command). When either one of the three bits is not the same overshoot condition exists and thus the step size decreases in the same manner as a voltage change across a resistor in a single pole R-C network. (hence the term discharge command). The charge and discharge of the step size are shown in Fig. 1.2-11.

The step size generator can best be explained by first considering a R-C network where the input is the maximum step size when slope overload was just detected and the output is the step size added to the estimated signal as long as slope overload or charge command is still true. If the maximum step size is labeled S_i and the next step size is labeled So(t) then:

$$So(t) = S_i(1-e^{-t/RC}) \qquad (1.2\text{-}4)$$

Since the step size varies discretely let us redefine $So(t)$ as $S(k)$ and $e^{-t/RC}$ as $e^{-k/RC}$ where k is the current sampling instance.

Thus:

$$S(k) = S_i(1-e^{-k/RC}) \qquad (1.2\text{-}5)$$

$$S(k+1) = S_i(1-e^{-1/RC}e^{-k/RC}) \qquad (1.2\text{-}6)$$

Therefore

$$S(k+1) = \{e^{-1/RC}|S(k)|+S_i(1-e^{-1/RC})\}e(k) \quad e(k)=e(k-1)=e(k-2) \qquad (1.2\text{-}7)$$

When the discharge command is true the step size will decrease, and:

$$S(k+1) = e^{-1/RC}|S(k)|e(k) \qquad (1.2\text{-}8)$$

The absolute value is necessary to insure proper charging and discharging. When no input is present $S(k+1)$ must not discharge to zero but rather to a minimum value usually 10 to 30 millivolts.

Let $e^{-1/RC} = \alpha$ then the CVSD algorithm can be summarized as:

$$\hat{m}(k+1)=\hat{m}(k)+S(k+1)$$

$$S(k+1)=\begin{cases} [\alpha|S(k)|+S_i(1-\alpha)]e(k) & e(k)=e(k-1)=e(k-2) \\ \begin{cases}\alpha|S(k)|e(k) & |S(k)|>So \\ S_o e(k) & |S(k)|\leq So\end{cases} & \text{otherwise}\end{cases} \qquad (1.2\text{-}9)$$

$$e(k) = SGN[m(k)-\hat{m}(k)]$$

A comparison between Eqs. (1.2-3) and (1.2-9) reveals the inherent difference between the SVADM and the CVSD algorithms. In Eq. (1.2-3) a constant positive value, S_o, is always added or subtracted from the current step size, S(k), to form the next step size S(k+1). On the otherhand, in Eq. (1.2-9) this is not the case. A constant positive value, $S_i(1-\alpha)$, is added to the current step size if and only if a certain condition exists, otherwise the step size remains unchanged.

Since the CVSD was designed to follow speech-like signals, it cannot follow square wave type signals (i.e. data), whereas the SVADM is more of a waveform tracker. Since speech is being anticipated in the CVSD it will filter out transmission errors which do not have speech characteristics, whereas the SVADM will try to follow those errors. The CVSD was found to have a 0.1 channel error rate performance and the SVADM a 0.01 channel error rate performance. However, the robustness to channel errors of the CVSD is penalized by a high sensitivity to sudden changes in input level. When the input signal level was reduced by 30 dB, the performance of the CVSD was found to be unsatisfactory compared to that of the SVADM.

1.3 Summary of prior work

In analyzing the performance of the LDM, two methods were used. In one case, granular noise and slope overload noise were considered statistically independent and were treated as such. Taub and Schilling [11] found an approximate formula for the average signal to granular noise power ratio. It was assumed that there is no slope overload and that the input waveform was sinusoidal. Although the approach employed as a heuristic one the result obtained is very close to the one obtained by Van de Weg [2] who used a more rigorous analysis. The second method is the exact study of the DM without the distinction between granular and slope overload noise. Fine [12] obtains a recursive relation for the joint distribution of the error and estimated signals for the case of input with independent statistics. Slepian [13] showed how to exactly compute the steady-state distribution and the mean square error of a LDM with an ideal integrator excited by a stationary Gaussian process with a rational power spectral density. However, complexity of the computations rapidly becomes untractable. Mills [14] and O'Neal [15], develop a method for determining quantizing noise power for a first order Markov-Gauss input. Although their procedure provides accurate calculations for quantization noise at all values of step sizes it is computationally difficult to apply.

On the other hand Zetterberg [16-17], Cutler [18] and Schindler [19] have proposed schemes to improve SNR performance of DM. Zetterberg and Cutler use the principle of delayed encoding. They conclude that delayed encoding will primarily affect the stability of a second order system in the form of sustained oscillations and thus by using a stronger adaptive scheme, higher SNR can be obtained. Schindler found that performance was improved by shaping the spectrum of the quantization error with the introduction of pre - and post - filters.

1.4 Summary of results obtained

In this section, we outline the contents of the dissertation that follows. In addition, we shall qualitatively summarize the main results that we have obtained.

Chapter 2 begins with the design requirements for the programmable real time voice processor, and the requirements for instruction memory size, instruction execution time and order of operation are established. To meet design goals available central processing units are examined, in particular the class of bit slices. A specific processor is chosen, and an instruction set is developed, and the system architecture is outlined with a detailed explanation of the system operation.

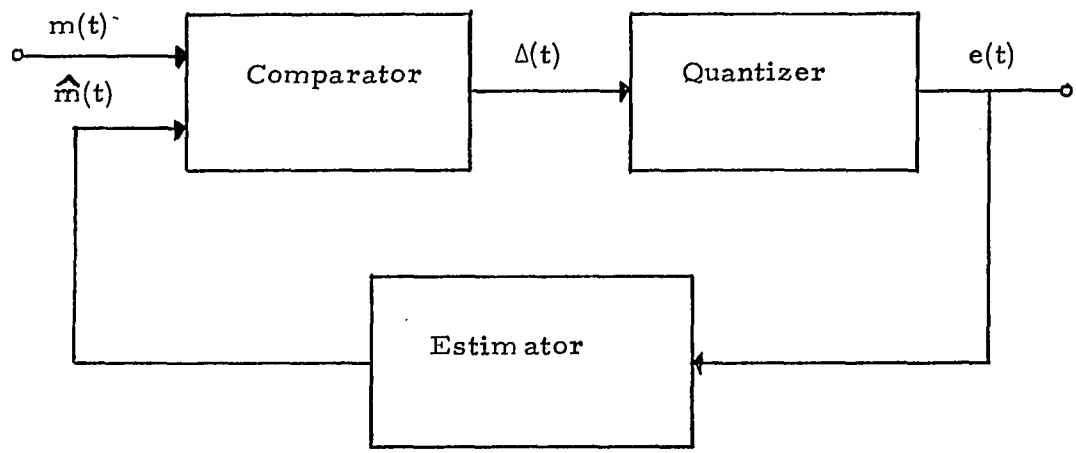
In Sec. 2.3 a SVADM transmitter-receiver algorithm is implemented and shown in the format of a flow chart in Figs. 2.3-1 and 2.3-2. Recognizing the problem of transmitter-receiver synchronization, the leaky integrator solution is studied and implemented. To minimize effects of channel errors on receiver operation, saturation logic is incorporated in the receiver design. Response of a SVADM receiver employing a leaky integrator and saturation logic is demonstrated in Fig. 2.3-14. Incorporating an averaging filter in the SVADM receiver improved voice quality at low sampling rates. However,

using look ahead with the SVADM algorithm did not seem to improve voice quality.

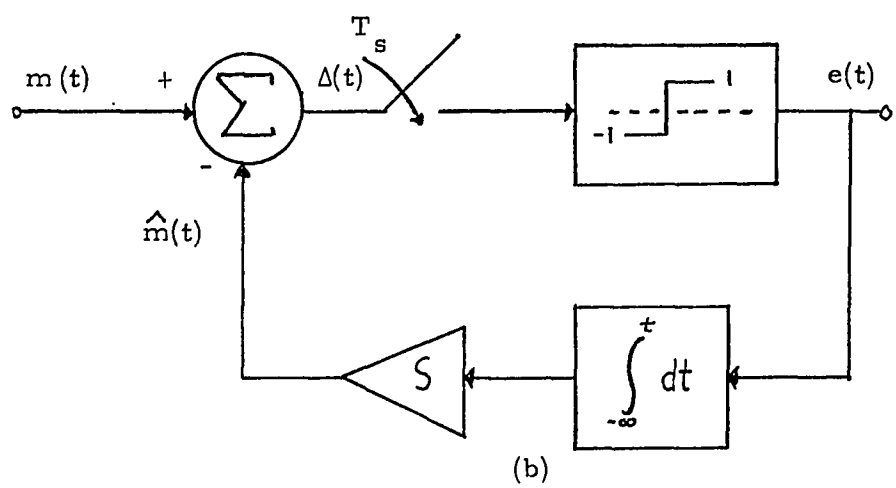
In Sec. 2.4 a CVSD transmitter-receiver algorithm is implemented, as an example of a syllabic delta modulator, with flow chart implementation shown in Fig. 2.4-1 and 2.4-2. Response of the CVSD algorithm to voice input is shown in Fig. 2.4-3 for different sampling rates. In Sec. 2.5 a switching receiver is implemented. Since 50% of speech is silence (i.e. D. C.), the e(k) pattern, which is unique to the CVSD and the SVADM schemes, can be used to detect and identify one of the two algorithms.

In chapter 3 we model the LDM probabilistically to obtain the autocorrelation and power spectral density functions of the estimate signal without assuming that granular noise and slope overload noise are independent. In Sec. 3.2 the LDM is modeled probabilistically for the case of a zero order Markov input. A Markov chain approach is used and transitional probabilities are shown to be a function of the input statistics alone. Also, an algorithm for computing the transition probabilities is obtained. Knowing the transition probabilities, a steady state transition matrix and the steady state level probabilities are obtained thus enabling the calculation of the autocorrelation and power spectral

density functions of the estimate signal. In Sec. 3.4 two examples are used. For a zero order Markov input with a uniform pdf, the autocorrelation function of the LDM estimate is found to be an exponential decay which indicates that the LDM acts as a low pass filter. For a white Gaussian input the autocorrelation is found to be an impulse which indicates that the LDM acts as an all pass filter. In Sec. 3.5 the LDM is modeled probabilistically for the case of a first order Markov input and an algorithm for computing the transition probabilities is obtained. An example of a first order Markov-Gauss input is shown and results agree with the ones found for the zero order Markov cases, as shown in Figs. 3.5-2a, 3.5-2b and 3.5-2c.



(a)



(b)

Fig. 1.2-1 : (a) DM Encoder Block Diagram,
(b) Linear DM Encoder.

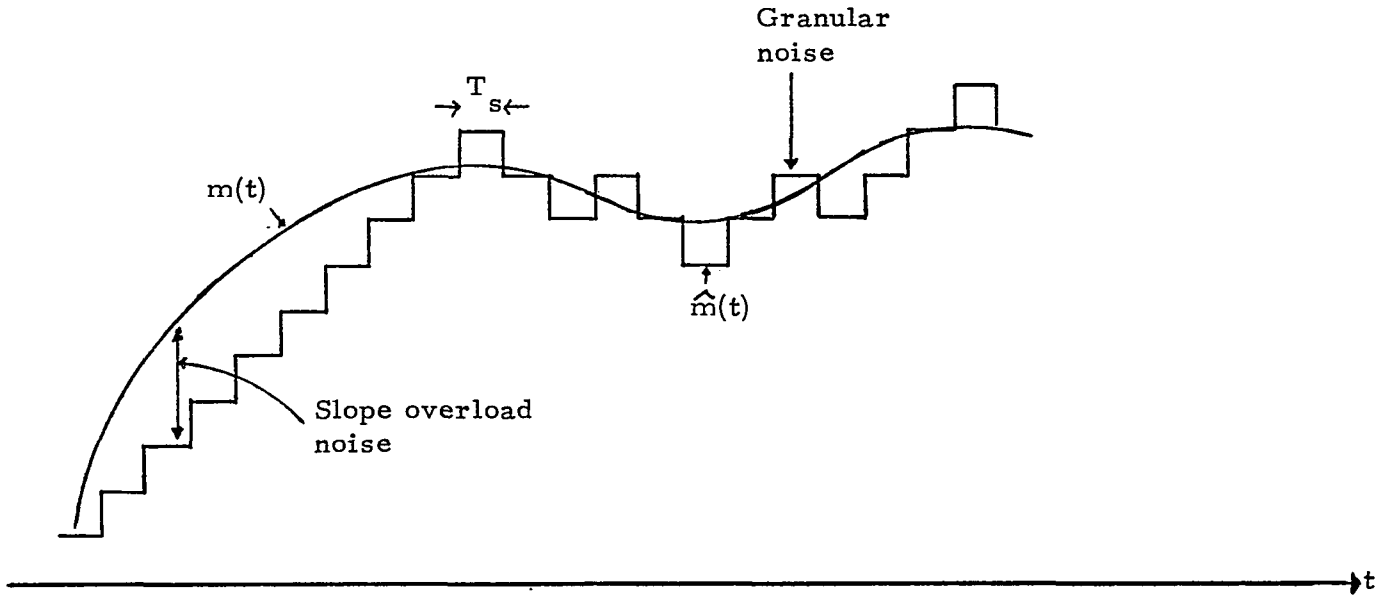


Fig. 1.2-2: LDM Response to an Arbitrary Input.

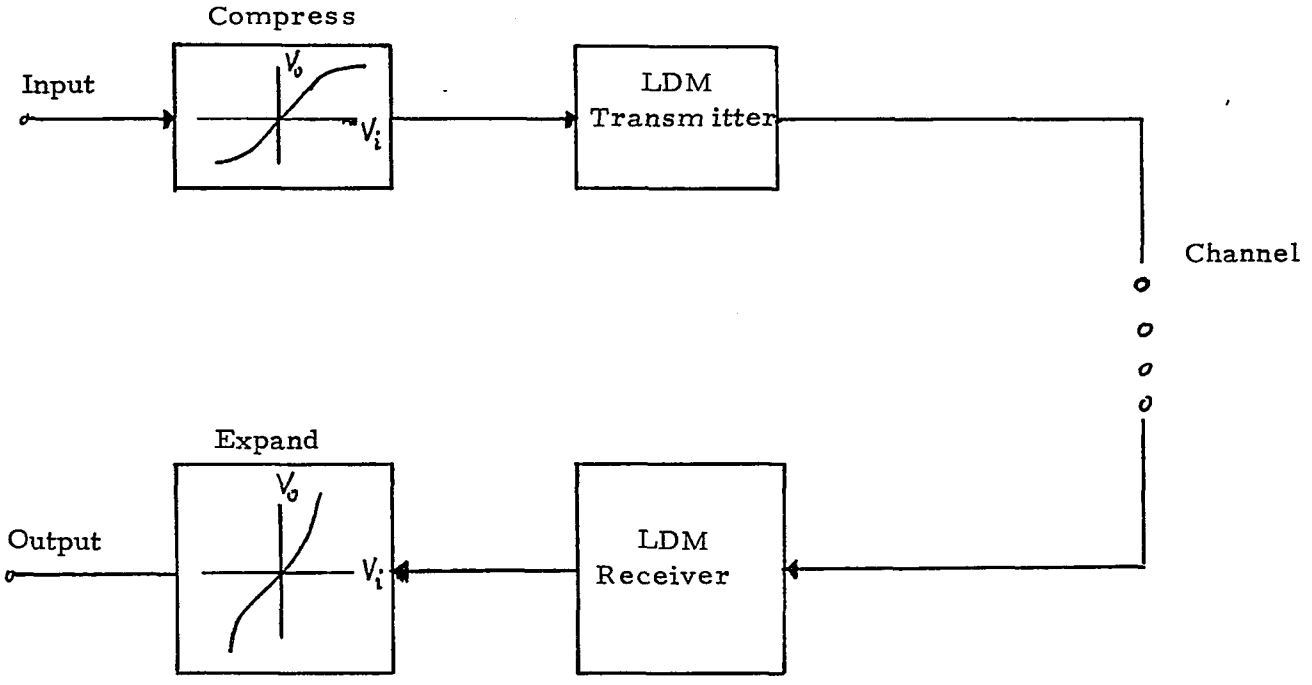


Fig. 1.2-3: The Compress-Expand Method to Improve LDM performance.

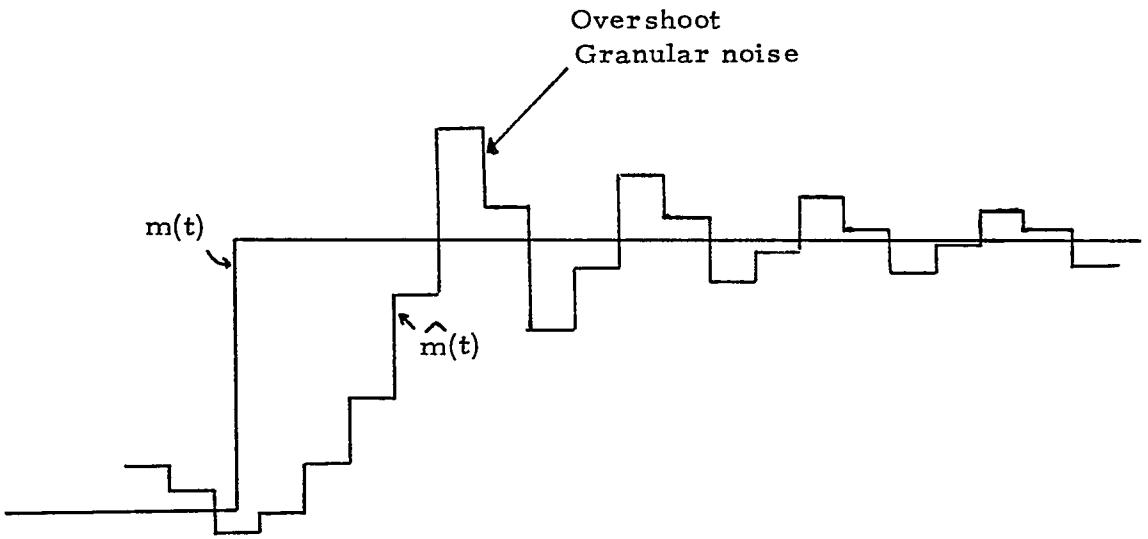
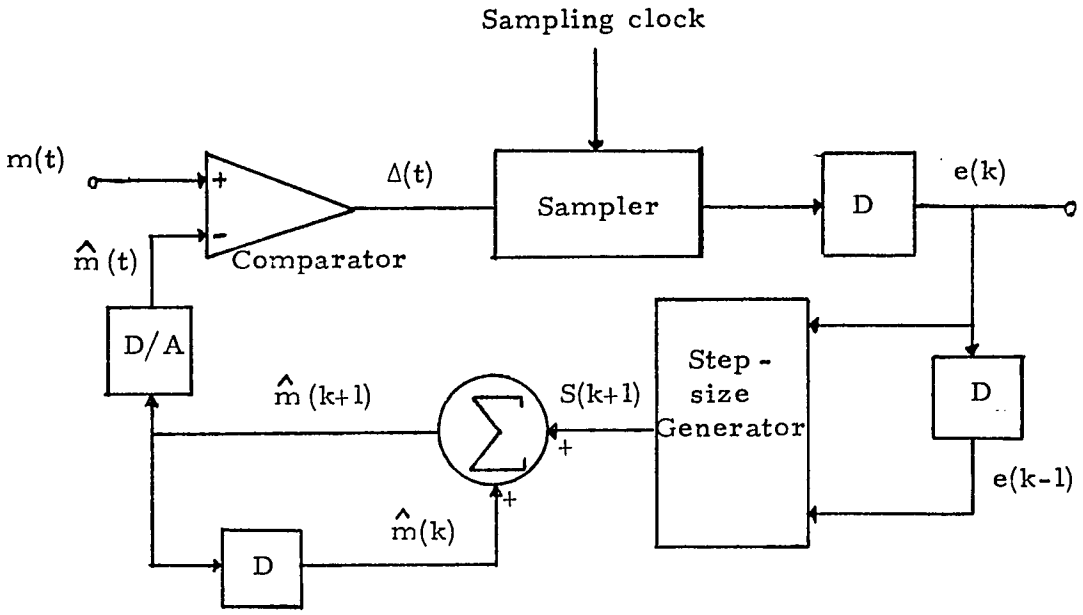
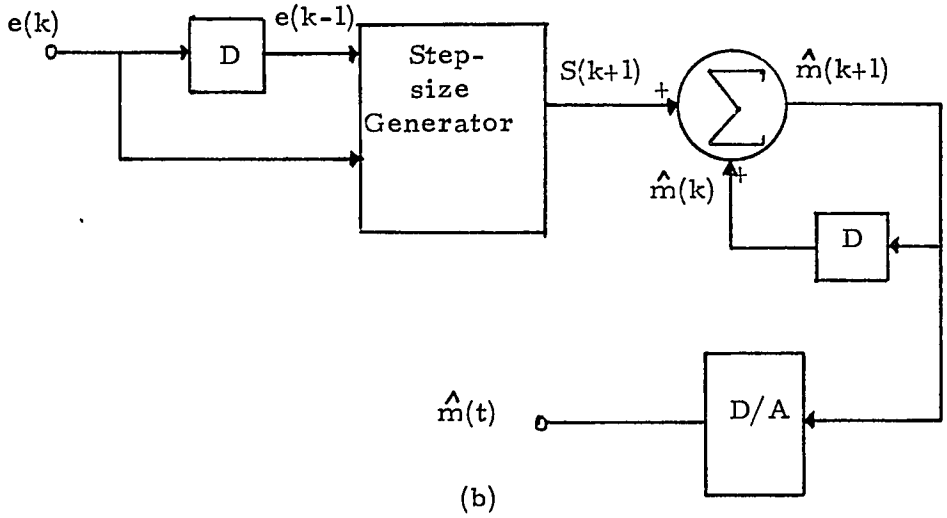


Fig. 1.2-4: Adaptive DM Step Response Exhibiting Overshoots and Oscillations.



(a)



(b)

Fig. 1.2-5 : Instantaneously Companded DM (a)Encoder, (b)Decoder.

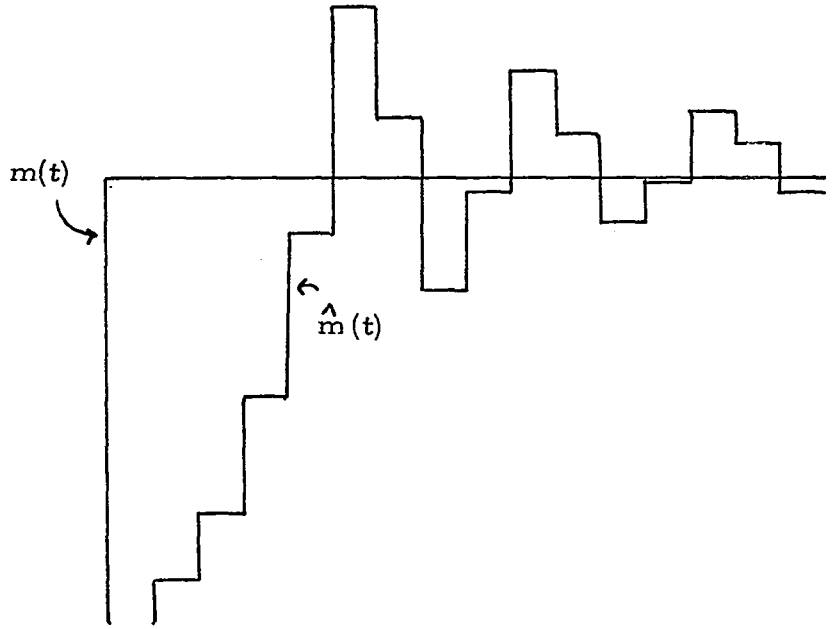


Fig. 1.2-6: Response of ICDM to a Step Input.

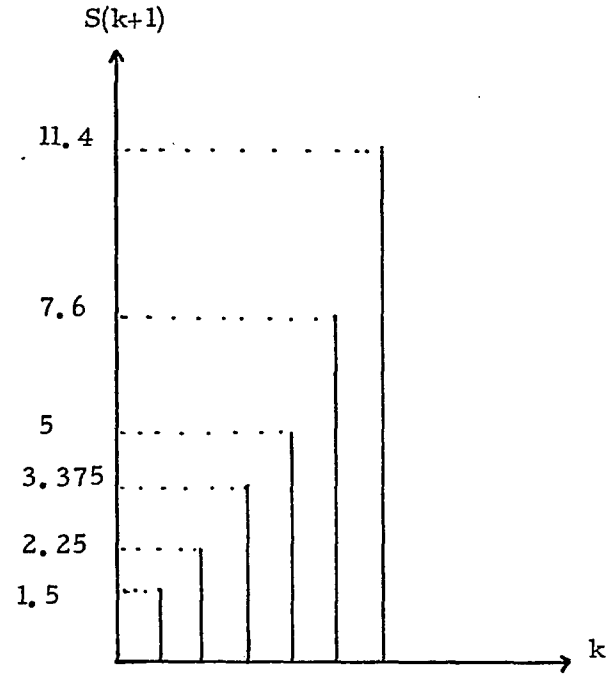


Fig. 1.2-7: Illustrating the Exponential Increase in the Step-Size S(k+1).

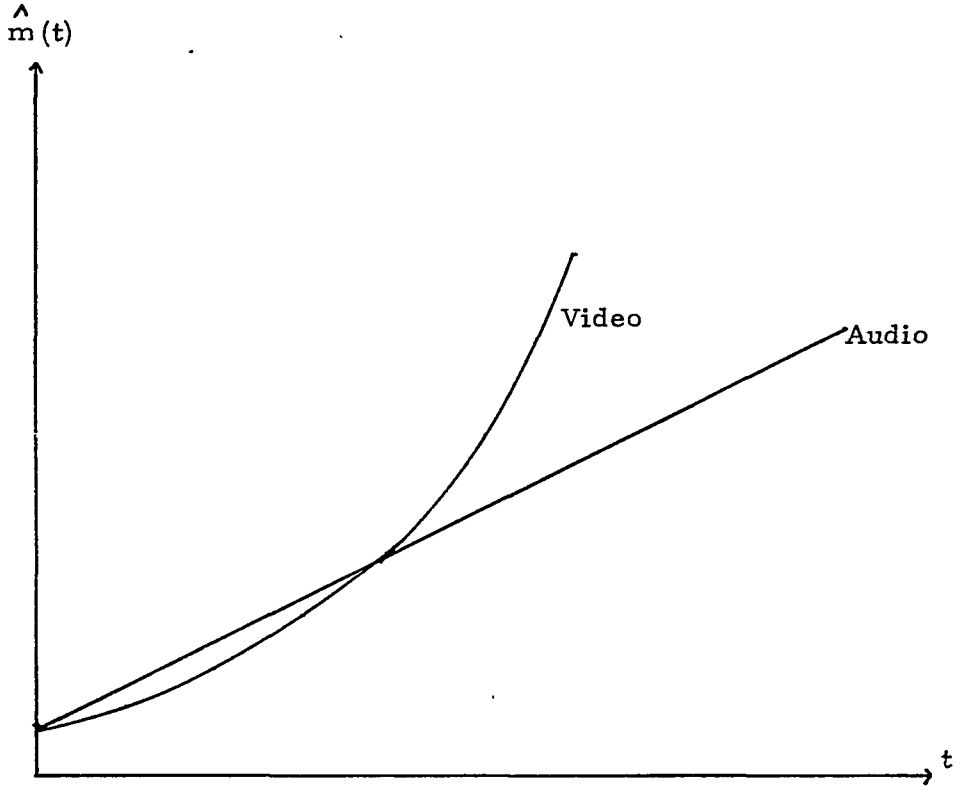


Fig. 1.2-8: Step Response of Audio and Video DM's.

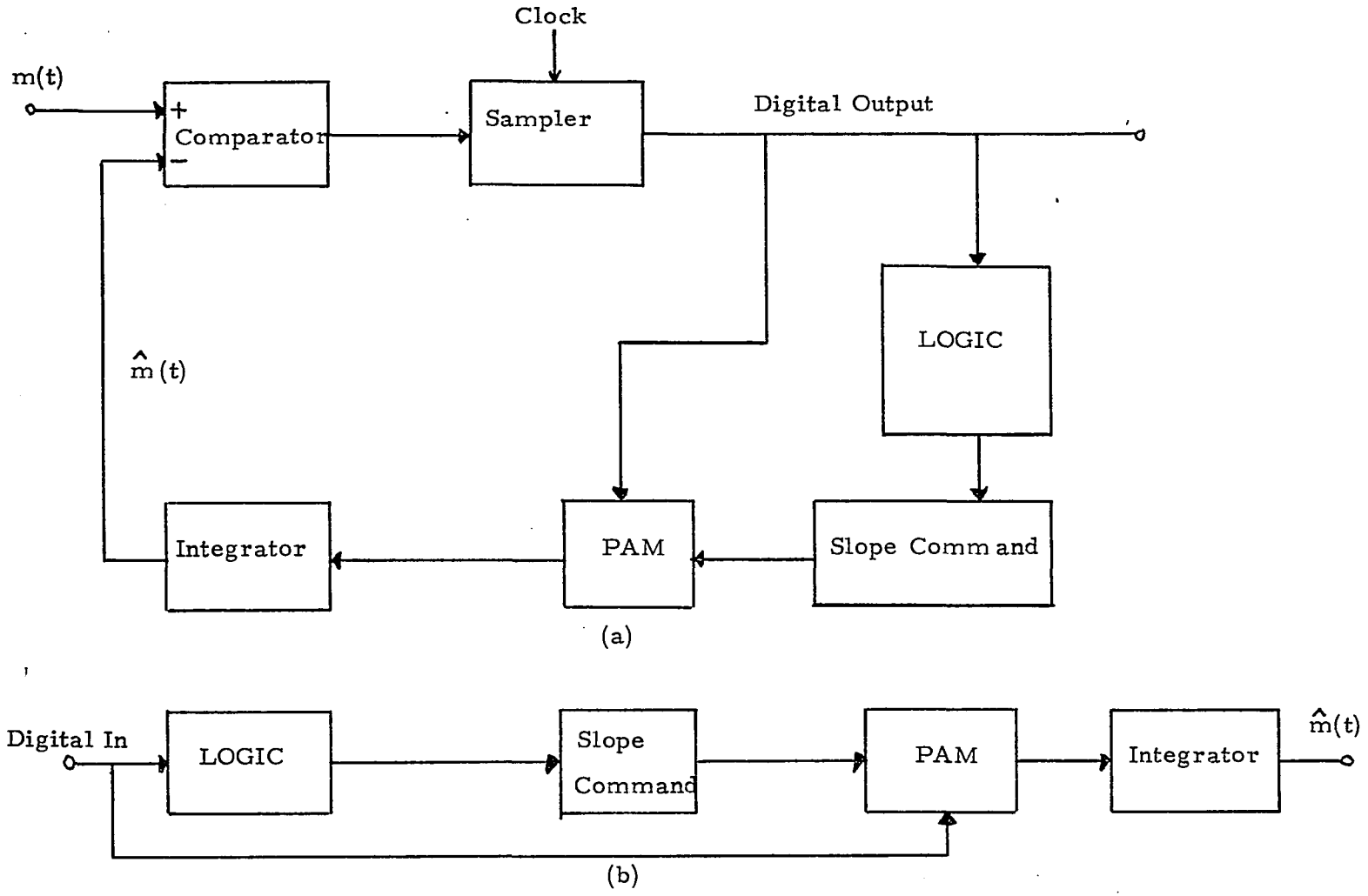


Fig. 1.2-9: Analog CVSD (a) Encoder, (b) Decoder.

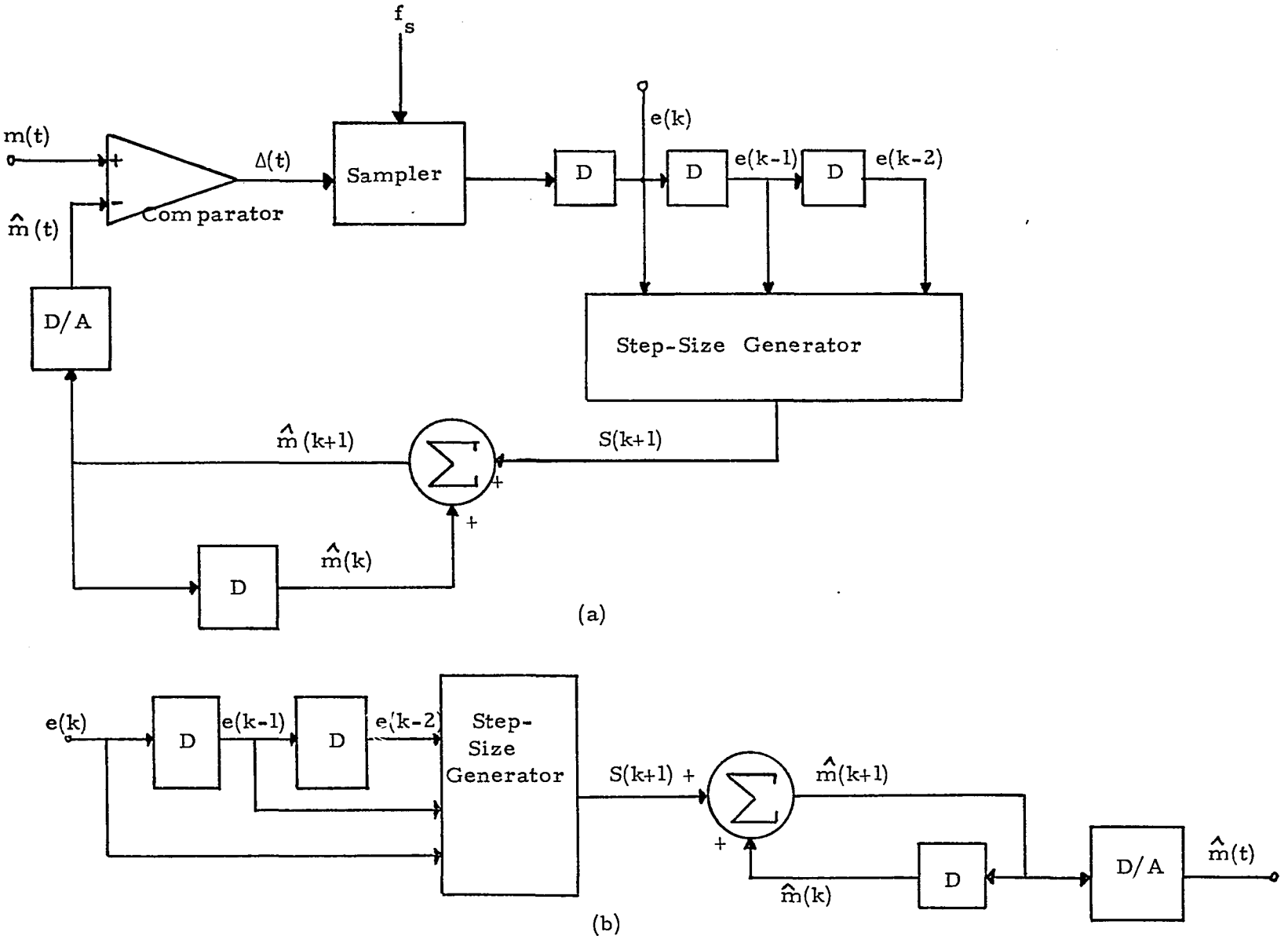


Fig. 1.2-10: Digital CVSD (a)Encoder, (b)Decoder.

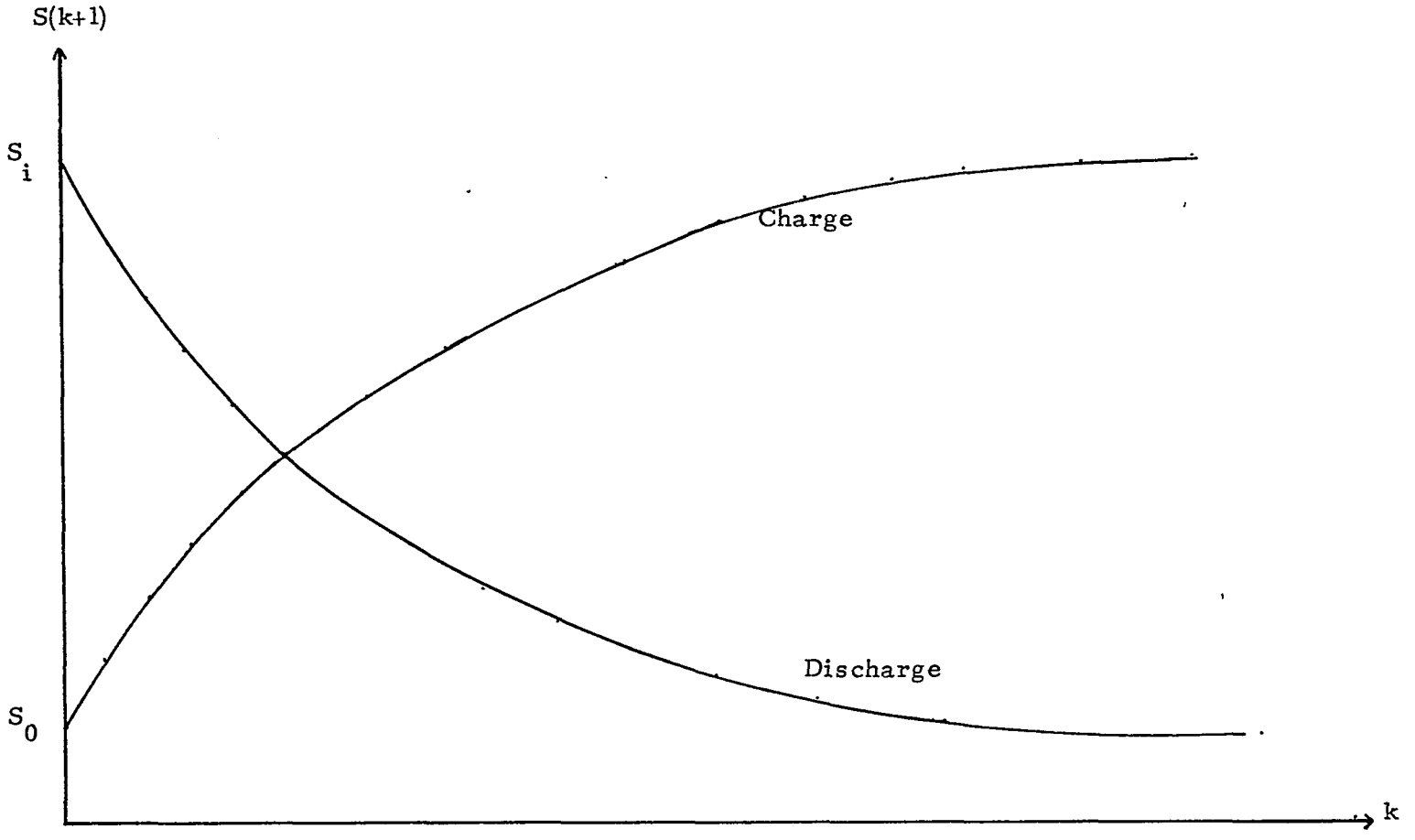


Fig. 1.2-11: Variation in CVSD Step-Size.

Chapter 2

2.1 Design Requirements for the Voice Processor

An intuitive way, of implementing Delta Modulation algorithms [i.e. Eqs. (1.2-3) and (1.2-9)] is to create a table look up using a ROM for the different step size algorithms, and then switch from table to table when different schemes are desired. However, since the tables are fixed, they cannot be modified, or changed, without creating new tables. Also, since the switching between tables is done manually the implementation is not truly programmable. Further communication between different algorithms is still not readily available.

The first requirement of a programmable device is that it be able to execute instructions stored in memory. Memory has to be volatile, or read/write Random Access Memory (RAM), so that instructions can be modified at will. A nonvolatile, or Read Only Memory (ROM), option can be included for permanent instruction sets that do not have to be changed.

Since instructions will contain logic and arithmetic commands, the device has to be able to execute arithmetic and logic functions. In other words, it has to contain, in some form, an Arithmetic Logic Unit (ALU).

The order, or sequence, of instruction execution is of importance too. Since the next step-size, S(k+1), is a function of $\underline{e(k)}$ (i.e. e(k),e(k-1),.... etc.), the instruction set will contain subsets which will not be executed at the same time. Instructions executed at sampling instant k, will not necessarily be the same as those executed at sampling time k+τ. Therefore, it is important to control the order by which instructions are executed.

Execution of instructions at sampling instant k must be completed before sampling time k+1. If T_s is the sampling interval, N_ℓ is the largest instruction subset and τ_e is the execution time of an instruction, than for proper operation:

$$T_s \geq N_\ell \tau_e \qquad (2.1\text{-}1)$$

It was found, experimentally, that as far as speech is concerned, sampling rates above 50kbs do not contribute any appreciable improvement to Delta Modulation performance. Thus $T_s \geq 20$ μsec, and if 100 instructions/sample were to be executed, instruction execution time would have to be limited to 200 nsec. The goal thus set is a 160 nsec instruction cycle time which would enable the execution of approximately 125 instructions at 20 μsec sampling intervals.

Furthermore, once instructions have been executed, there will be a time gap before the next sample is ready to be processed. It is therefore required, of the device, to identify the last instruction to be executed and halt further operation until the next sampling instance.

In addition, a full duplex operation is desired, since transmitter and receiver have to be implemented on each side of a communication channel. The requirement for full duplex operation is the capability of simultaneously dealing with analog signals as well as with digital information.

Finally, since instructions will be stored in memory, the capability of addressing and entering information as well as examining information stored has to be incorporated in the design. Instruction address and content can be entered via switches whereas information stored can be displayed via Light Emitting Diodes (LED).

2.2 Real Time Programmable Microprocessor

2.2-1 The bit slice

Since look-up tables are not considered to be a good solution, using the ALU seems the way to go.

An ALU, shown in Fig. 2.2-1, is a device which can perform sixteen logical and sixteen arithmetic operations on two 4-bit variables. The choice between an arithmetic operation and a logic operation is performed by the mask input bit labeled M. When M="0" an arithmetic operation will be performed on variables A and B with result indicated by function output F. When M="1", a logic function will be executed. The instruction command is available at input $\underline{S} = S_0S_1S_2S_3$. Table 2.2-1 shows the relationship between A B and F for all possible instructions.

Using the ALU as the heart of the processor, the following is still needed to meet design requirements:

a) Instruction memory - can be either a ROM or a RAM or both.

b) System controller - to supervise instruction execution sequence.

c) Temporary storage memory - To store intermediate results (also known as a scratch-pad memory).

d) Input-Output devices - To convert analog signals into digital format and vice versa.

The implementation of the processor as just outlined seems to require too many components. The introduction of microprocessors, on the other hand, seems to minimize the number of components to be used. The rule of thumb is that if a system contains more than 30 Integrated Circuits (I C's) it is advantageous to replace the system by a microprocessor. Since the instruction execution time has to be less than 200 nsec, MOS microprocessors, such as the Intel 8085, which have an instruction cycle time in the order of microseconds, are not applicable. The only choice thus left is the bipolar programmable microprocessor slice.

A microprocessor slice is a device, capable of arithmetic and logic operations, having n-bits, usually two or four, which is expandable. K-slices of the same device, can be connected in series to form a k·n bits processor. It is also microprogrammable in the sense that the user has the option of constructing his own instruction set. At this time there are available, commercially, three bipolar microprogrammable microprocessor slices: a) The Advance Micro Devices (AMD) 4-bit Am 2901 slice, b) The Intel 3002 two bit slice, and c) The Motorolla MC 10800 4-bit slice.

The Intel 3002 shown in Fig. 2.2-2 is a 2-bit Central Processing Unit [CPU] employing bipolar Transistor Transistor Logic (TTL) technology.

It contains the following: Two input terminals and two output terminals allowing full duplex operation and an ALU, which is controlled by the microfunction decoder and the k-bus, capable of a variety of arithmetic and logic operations. In addition, each slice contains eleven general purpose registers. This extra feature is very attractive, for if one does not need more than eleven registers for temporary storage, then one does not have to access external memory and can, consequently, achieve a faster instruction cycle time. The basic instruction cycle time for each slice is 100 nsec, with a minimum of 8-bits per microinstruction (F_0-F_6 and k).

The AMD Am 2901 shown in Fig. 2.2-3 is a 4-bit CPU slice employing TTL technology. It contains a 16-word by 4-bit 2-port RAM, a high speed ALU and the associated shifting, decoding and multiplexing circuitry. Eight bits are needed to address two of sixteen registers, and with a basic microinstruction word of 9-bits, the minimum instruction word contains 17-bits with a minimum cycle time of 105 nsec. The microprocessor is expandable with full look ahead or with ripple carry, it has two inputs and one output, and provides four status flag output from the ALU.

The MC10800 (Emitter Coupled logic Technology) 4-bit ALU slice, as seen in Fig. 2.2-4, is nothing more than a modified ALU. It does not contain a microfunction decoder and it does not contain general purpose registers. Data enters and exits the MC10800 slice through the A bus, output (ϕ) bus, and input (I) bus. The ϕ and I buses are bidirectional, that is data can enter and exit on the same bus, while the A-bus is unidirectional and an input port only. Seventeen select lines control the function performed by the circuit and determine the source and destination for data with a 60-nsec typical instruction cycle time. Five status outputs are also available for condition codes and branch test.

The MC10800 4-bit ALU slice seems to be the fastest slice considered. However, the need for a microfunction decoder (17 control lines), a register file and various multiplexing circuitry makes the overall instruction cycle time comparable to the other two slices, with the disadvantage of added components. Of the three slices available, the Intel 3002 2-bit slice, lends itself almost naturally, to a full duplex mode of operation with an instruction set which is versatile, yet simple, enough to allow such operation. Since the instruction cycle time is comparable to the other two slices, and since the smaller size of the internal register file

(11 general purpose registers compared to 16 in the Am 2901) is offset by the smaller instruction size (8 bit per instruction compared to 17-bit per instruction in the Am 2901), the Intel 3002 2-bit slice was chosen.

2.2-2 System architecture

To meet design goals the voice processor as seen in Fig. 2.2-5 has four major components:

a) Central Processing Unit (CPU)
b) Instruction memory
c) Program controller (PC)
d) Input-Output (I/O) devices.

The CPU consists of five 2-bit slices allowing for a 10-bit dynamic resolution. The instruction memory, consist of RAMs or PROMs, and wholly contains the instruction set. Read/write memories are used for temporary storage and once the program is working satisfactorly it can be transferred to a non volatile memory or PROM. Thus, there is an option of selecting between RAMs and PROMs. The 74S200 (S stands for shottkey TTL technology), a 256x1 bit read/write memory, having a maximum access time of 50 nsec, was chosen as a temporary storage unit for the instruction set and the 82S114 PROM with a maximum read time of 60 nsec was chosen as the permanent storage, non volatile, unit of the instruction set.

Each instruction is a twelve-bit word as shown in Fig. 2.2-6. The first nine bits contain the OPeration Code [OP-code, i.e. $F_0,F_1,F_2,\overline{K}$], the source and destination registers address (i.e. F_3,F_4,F_5 and F_6)

and the carry input information. The tenth and twelfth bits are used when a "JUMP" instruction is desired, and the eleventh bit, called $\overline{ED}$, is used as a command to load the output of the CPU array into an output device.

The read-write memory outputs are multiplexed with read only memory outputs to allow a convenient way of choosing either one. Further more, the OP-code bits are multiplexed with a NO OPeration (NOP) code such that whenever a "Jump to Location", "conditional" or "unconditional", or "skip" instruction is desired, a NOP instruction will be loaded into the instruction register, so that normal CPU operation and the state of the internal registers will not be affected.

The micro-instruction set, shown in tables 2.2-2 and 2.2-3 is divided into two columns controlled by the k-bus. When the k-bus is at logic level "zero", the left column is chosen, and when the k-bus is at logic level "one", the right column is used. In other words, the k-bus masks the participation of the accumulator in the execution of a specific instruction. A k-bus equal to logic level "zero" means the accumulator does not participate, whereas a k-bus equal to logic level "one" means the accumulator will participate in the execution of a certain instruction.

Each column is broken into eight groups and the first three bits of the instruction word contain the group information. Each group is divided into three subgroups, and bits five, six, seven and eight contain that information. As an example, suppose that the instruction word is:

Instruction word = 1001 1011 1010 (2.2-1)

The first three bits indicate the decimal equivalent "4" which means that the fifth group is chosen (0,1,2,3,4). The next bit is a "1" which in turn mean that the fifth group of the left column is chosen ($\bar{k}$=1 .: k=0). Since the next four bits are 1011 it is deduced, from table (2.2-3), that the instruction is Clear Accumulator (CLA).

The address inputs of the memory are controled by the Program Controller (PC). The PC is basically a counter as shown in Fig. 2.2-7. The counter is a synchronous eight bit binary fully programmable counter; that is, the outputs may be preset to any level. As presetting is synchronous, setting up a low level at the load input, L, disables the counter and causes the outputs to agree with the setup data after the next clock pulse regardless of levels of the enable input P. Otherwise, the enable count P

has to be at logic level "one" in order to enable the count. The counter chosen was the 74S161 4-bit binary counter and thus two were used to obtain 8 bits. Eight external switches and eight bits of the instruction word are multiplexed to form the set-up data. To implement a halt state, the count 255 was used to disable count until the sampling clock loads the address of the first instruction to be executed which is indicated by the external switches.

The input device consists of a 10-bit Analog to Digital (A/D) converter employing a modified successive approximation technique. The output is in two's complement format with a 16 μsec conversion time. To avoid timing problems, the output of the A/D is latched into a register which is called the input register. On the leading edge of the A/D clock, which is the sampling clock, the output of the A/D is transferred into the input register. On the trailing edge of the clock, a new input sample is processed. To minimize the dependency of the processor operation on the sampling clock pulse width, the input sampling clock is passed through a pulse shaper circuit shown in Fig. 2.2-8a. The relationship between the input and output pulses is shown in Fig. 2.2-8b. The output of the CPU array

loads a register, called the output register, which transfers data to a 10-bit Digital to Analog (D/A). Information is loaded into the output register on the trailing edge of the clock. Outputs of the CPU have to be latched since after a high-to-low transition of the clock, the outputs of the CPU enter a high impedance state (outputs are floating). The circuitry which supervise proper loading of the output register at the proper time is shown in Fig. 2.2-9.

The basic instruction cycle time for each slice is 100 nsec. Due to the expansion of five slices, the instruction cycle time is increased to 120 nsec. Also, an additional 40 nsec must be included as a result of other hardware delays. Thus the total instruction cycle-time was set at 160 nsec. With "carry look ahead" circuitry, the total instruction cycle time can be reduced to 130 nsec. (However, we did not employ carry look ahead in our design). To maximize throughput, the clock was made unsymmetrical with 120 nsec on time and 40 nsec off time. The crystal controlled clock circuitry is shown in Fig. 2.2-10.

2.2-3 System operation

The voice processor is controlled by two clocks. One is the A/D clock, or the input sampling clock. This clock loads the digital representation of the analog input into the input register and presets the counter to a value set by the external switches. This, in turn, starts a new processing cycle. The second clock is the instruction execution control clock, or the system clock, operating at 6.25 MHZ. The counter counts at the system clock rate which in turn implies that the microinstructions are executed at the rate of 160 nsec per instruction.

At the leading edge of the sampling clock, the counter is preset to a predetermined value set by the external switches. At the next system clock pulse, the first instruction is loaded into the instruction register, and the CPU executes that instruction. At the same time, the counter content will be incremented by 1 such that the next instruction is available at the inputs of the instruction multiplexers as shown in Fig. 2.2-11. The next instruction to be executed is either a NOP or the stored instruction. This occurs at the leading edge of the next system clock pulse, and

depends upon the tenth bit of the previous instruction, the ripple carry out of the CPU array from the previous arithmetic or logic operation and the twelfth bit of the present instruction. At the end of a program, control will pass to the last available location, which in turn will disable the counter. The system is then ready for the next input sample. At a system clock rate of 6.25 MHz and a sampling clock rate of 32 KHz, 195 instructions can be executed between samples.

Program loading is done manually via switches. The address of the instruction to be loaded is inserted into the counter via the external switches, shown in Fig. 2.2-7, when a control switch is moved to load address position. Since the content of the counter now points to the desired location, an instruction word can be written into that location when a command write is given (also manually). The counter outputs and the content of the corresponding memory location are displayed on Light Emitting Diodes (LED's) to ensure proper program loading.

A mnemonic (symbolic) listing of the modified instruction set is shown in appendix A with a complete circuit schematic of the microprocessor shown in appendix B.

We turn now to the implementation of two sample algorithms:

1. Song Voice Mode DM (SVADM) representing an ICDM algorithm.

2. Continuous Variable Slope Deltamodulator (CVSD) representing a SCDM algorithm.

2.3 Implementation of a Song Voice Mode DM Transmitter-Receiver algorithm

2.3-1 Simple transmitter-receiver

It is convenient at this point to repeat the SVADM algorithm:

$$\hat{m}(k+1) = \hat{m}(k)+s(k+1) \qquad (2.3\text{-}1a)$$

$$s(k+1) = |s(k)|e(k)+s_o e(k-1) \qquad (2.3\text{-}1b)$$

$$e(k) = \text{SGN}\,[m(k)-\hat{m}(k)] \qquad (2.3\text{-}1c)$$

Where S(k+1) is the computed step-size value such that when added to the current estimate, $\hat{m}(k)$, produces the next estimate $\hat{m}(k+1)$. Since e(k) is the data transmitted over the channel, the first two equations (i.e. (2.3-1a) and (2.3-1b)) also constitute the basic receiver equations.

The SVADM algorithm shown above is a recursive one. At every sampling instant, the transmitter will determine the sign of the error signal and will use that information in the generation of the next step-size, in the same manner it did for all previous samples. In other words the DM equations will be executed the same way every sampling instant. Since the algorithm is recursive, implementation of it can best be shown by a flow chart as shown in Figs. 2.3-1 and 2.3-2.

At the leading edge of each sampling clock, a new input sample is stored in the input register, and is available for comparison with the previous estimate stored in one of the internal registers of the CPU. The absolute value of the previous step-size (i.e. $|s(k)|$) is obtained by checking the Most Significant Bit (MSB) of the step-size register and two's complementing, $s(k)$, if the MSB = "1" which indicates $e(k)$ negative. The previous estimated value is then subtracted from the input sample. If the difference is positive, a "1" (i.e. 0000000001) is stored in the $e(k)$ register, and a "-1" (i.e. 1111111111) is stored when the difference is negative. The $e(k)$ register is then checked again and if $e(k)$ is negative (i.e. a "-1" stored), the absolute value of the previous step-size (i.e. $|s(k)|$) is inverted and $-|s(k)|$ is obtained. If $e(k)$ is positive (i.e. "1" is stored), the absolute value of the previous step-size if left unchanged. The next step-size is obtained by adding the value stored in the $e(k-1)$ register to the previous step-size. The minimum step-size S_o has the magnitude as the Least Significant Bit (LSB). With a 10-bit dynamic resolution and a 10Vp-p input signal the LSB is approximately 10mV. The next estimate is obtained by adding the current estimate to the next step-size.

The delay function is implemented by storing the current e(k) in the e(k-1) register, $\hat{m}(k+1)$ in the $\hat{m}(k)$ register and S(k+1) in the S(k) register. Process is now complete. Control is passed to the last available location, which in turn implies that the voice processor enters a WAIT state. A new cycle will begin on the leading edge of the sampling clock. The list of instructions implementing the SVADM transmitter is listed in appendix C. Receiver operation is similar to the transmitter with the modification that the e(k) is received from a transmitter and not generated by the receiver. Again, the list of instructions implementing the receiver is shown in appendix C.

Having a receiver identical to the feedback loop, or local receiver, does not guarantee proper operation. Should transmitter and receiver not be alligned a run away condition may occur. A receiver is alligned to a transmitter if the current estimate and step-size registers content of receiver and transmitter are the same. However, since transmitter and receiver do not share the same registers special care must be taken to ensure such allignment.

2.3-2 The leaky Integrator

For proper transmitter-receiver operation it is necessary to have the same estimates and step-size values in the transmitter and receiver. However, experimentally the step-size was never found to be larger than $64S_o$ and guaranteeing the same estimates or very close estimates ensures proper receiver tracking operation.

The steady state, or D-C, response of the SVADM can be used efficiently in the allignment procedure. The steady state pattern shown in Fig. 2.3-3 exhibits a repetitive bit pattern (i.e. 110011001100....). Should a constant value be subtracted from a positive valued estimate, or added to a negative valued estimate, then if transmitter exhibits a long enough steady state pattern, the receiver estimate will approach, or leak off to zero.

It is well known that speech has many dead time periods. A dead time period is one in which there is no signal energy (i.e. no speech). Since transmitter follows the input signal, the estimate register content during dead time tracking will be close in value to zero (speech has no D-C component). Hence, a receiver employing a leak factor as shown in Fig. 2.3-4) or a leaky accumulator which is a digital RC filter will allign to transmitter during dead time.

The question now to be answered is what should the leak factor be? Too of a small leakage will not be sufficient in alligning the receiver, and too large of a leak will distort receiver operation. However, if the leakage was made to be a function of the estimate level, as shown in Fig. 2.3-5, then both conditions would have been met. Not too large when signal level is low and not too small when signal level is large. On the other hand, a leakage factor obeying the exponential rule, as discussed above, is not easy to implement.

A linear approximation, as coarse an approximation as it may seem to be, turns out to be effective if some speech characteristics are taken into account. The envelope of speech, for one, is a slowly varying signal with a periodicity in the order of 100 msec. This rate is generally known as the syllabic rate of speech. Also, if speech is separated into voiced and unvoiced components, than the voiced component, also known as pitch, has a period in the order of 10 msec. The unvoiced component is non periodic and resembles noise. A linear approximation is shown in Fig. 2.3-6. L_f is the leak factor, Am is the maximum amplitude of the estimate signal, Ts is the sampling period and T_p is either the syllabic or pitch period of voice. Hence, the leakage factor is found to be:

$$L_f = \frac{AmTs}{T_p} \qquad (2.3\text{-}2)$$

Table 2.3-1 tabulates the leak factor needed for different sampling rates and a 5Vp-p input signal level. It turns out that a leak factor following the pitch rate (i.e. T_p=10 msec) is much too large and performance was degraded considerably even when the leak factor was kept at minimum (i.e. 10 mv). On the other hand how does one implement a leak factor following the syllabic rate (i.e. T_p=100msec) when the smallest value is 10mv. (The LSB has a weight of 10mv). One way of solving the problem is not to leak every sampling instant which will effectively reduce the leak factor. For instance, when the sign of the step-size register content, S(k+1), is different from the sign of the estimate register content, $\hat{m}(k)$, there is no need to leak the estimate. When the step size, S(k+1), is added to the present estimate, $\hat{m}(k)$, to form the next estimate signal, $\hat{m}(k+1)$, the new estimate is closer in magnitude to the value zero than the present estimate. Hence, the estimate will leak only when the step-size register has the same sign as the estimate signal. The flow chart implementing such a leak factor is shown in Fig.2.3-7, and the list of instructions in appendix C. Figure 2.3-8 shows response of the

adaptive delta modulator to a slow varying square wave input without a leak factor and Fig.2.3-9 shows response with a 10 mv leak factor demonstrating the effect of the leak factor on the DM response.

2.3-3 Saturation Logic

One of the major causes for degradation in receiver performance is channel errors. An error will occur if the received e(k) bit is not the same as the transmitted e(k) bit. The digital information, the e(k) bits, control the sign of the step size as well as the value, or content, of the step size register. Hence, the new estimate signal value, which is incorrect due to the error, will have a D-C offset with respect to the correct level as seen in Fig.2.3-10. If the present estimate value of the receiver is near its maximum value, an error might cause it to overflow as seen in Fig.2.3-11a. With multiple errors the probability of overflow increases and system performance degrades considerably. To minimize effects of overflow, saturation logic is employed. The signal is held at its maximum magnitude when overflow is detected. Overflow conditions exist whenever the signs of the step-size register content, S(k+1), and the current estimate register content, $\hat{m}(k)$,

are the same but different from the sign of the new estimate, $\hat{m}(k+1)$. The flow chart of a program implementing saturation logic is shown in Fig.2.3-12 with instruction listing in appendix C. It should be noted that during dead time the leak factor forces DM estimate value to approach zero. Hence, all previous errors are corrected during dead time. Figure 2.3-13 demonstrates DM response, in channel errors environment, to speech. Figure 2.3-14 shows the response of a DM employing a leaky integrator and saturation logic.

2.3-4 The Averaging Filter

One of the characteristics of ADM is the steady state (D.C.) response as seen in Fig. 2.3-3. As can be verified, the estimate signal exhibits a tone at 1/4 of the sampling frequency. The receiver output is passed through a band pass filter to remove out of band noise. Since we are interested in voice frequencies, the input and output filters bandwidth is set to 2500 Hz with cutoff frequencies set at 300 Hz and 2800 Hz. It is evident that as the sampling frequency approaches 12KHz the 1/4 frequency component moves into band, and will introduce distortion. To eliminate such effects a simple, yet effective, linear filter is used in which the estimate

is passed through a running average where the present estimate level is added to the previous three. The sum is then weighted (i.e. divided by four) and result is passed through the band pass filter. It can be shown that such a filter has a zero at the quarter sampling frequency. The flow chart implementing such a filter is shown in Fig.2.3-15. The response of an ADM, employing a running average filter, to a 200 Hz square wave input is shown in Fig.2.3-16. Note the flatness of the response in contrast to the oscillations in a square wave response of an ADM not employing such a filter. Subjective listening has also indicated that a system employing a running average filter has a better response to voice than a system which does not employ such a filter.

2.3-5 Delayed Encoding

Overshoots found in an ADM response to a square wave input as shown in Fig.1.2-4 can yet be eliminated by using a delayed encoding, or look ahead, principle. Look ahead requires a criterion and an algorithm for finding the best or a good sequence of transmitted symbols. A reasonable procedure is to store a number of input signal samples $\underline{M}(k)$ (i.e. $m(k), m(k+1), \ldots, m(k+n-1)$,

and then, for all binary sequences $\underline{e(k)}$ (i.e. $e(k),e(k+1,\ldots,e(k+n-1))$ construct the estimate sequence $\underline{\hat{M}}(k)$ (i.e. $\hat{m}(k),\hat{m}(k+1),\ldots\hat{m}(k+n-1)$. The distance between the sequence $\underline{M}(k)$ and $\underline{\hat{M}}(k)$ is measured with a suitable criterion function $Q(\underline{M}(k)-\underline{\hat{M}}(k))$ for all sequences $\underline{e(k)}$. The binary symbol to be transmitted is given by the first component of the $\underline{e(k)}$ sequence that minimizes Q. The procedure is then repeated with k replaced by k+1. The criterion function used here is

$$Q(\underline{M}(k)-\underline{\hat{M}}(k)) = \sum_{j=o}^{n-1} |\underline{M}(k+j)-\underline{\hat{M}}(k+j)| \qquad (2.3\text{-}3)$$

i.e. the distance is measured by the square root of the noise power over the interval.

Due to limit number of registers available and due to the complexity involved, n=2 was chosen. Since there are four possible paths, as seen in Fig.2.3-17, and since decision has to be made on the first component of the $\underline{e(k)}$ sequence the number of paths to be considered reduces to the two shown in Fig.2.3-17. The flow chart implementing an ADM with delayed encoding is shown in Fig.2.3-18 with instruction implementation shown in appendix D. Response of the 2-bit look ahead to a step input is shown in Fig.2.3-19 with the response of an ADM not employing look-ahead superimposed on it. Note that

the step size in the scheme employing look ahead always approaches the minimum value.

Using look ahead with the SVADM algorithm did not seem to improve voice quality.

2.4 Implementation of a SCDM Transmitter Receiver algorithm

The CVSD algorithm is implemented as an example of a syllabic Delta Modulator. As seen from the CVSD algorithm [eq.(1.2-9)], the step size generation is a function of the attenuation factor α, as well as of past history. The attenuation factor, in turn, is a function of the syllabic filter time constant, RC, and the sampling frequency T_s. Assuming an envelope frequency of 100 Hz, the time constant, RC, is found to be 1.6 msec. with a 9.6 kHz sampling rate, α is found to be:

$$\alpha = e^{-T_s/RC} \tilde{=} 0.94 \qquad (2.4\text{-}1)$$

The minimum step size, S_o, was set at 10 mv and with a compression ratio of 20 (i.e. $S_{max}/S_o=20$) the maximum step size, S_{max}, was set to 200 mv. Hence, the CVSD algorithm using above parameters is:

$$S(k+1) = \begin{cases} \{0.94|S(k)|+0.01\}e(k) & e(k)=e(k-1)=e(k-2) \\ \begin{cases} 0.94|S(k)|e(k) & S(k+1)\geq 10\times10^{-3} \\ & \quad e(k)\text{'s not same} \\ 10\times10^{-3}e(k) & S(k+1)<10\times10^{-3} \end{cases} & \end{cases} \qquad (2.4\text{-}2)$$

The flow chart implementing the CVSD transmitter algorithm is shown in Fig.2.4-1. Figure 2.4-2 shows receiver flow chart implementation and Program instruction listing is shown in appendix E. Note that saturation logic of the step size was incorporated in the design. The response of the CVSD to voice input is shown in Fig.2.4-3 for four different sampling rates; 9.6kHz, 16kHz, 24kHz and 32kHz.

2.5 Switching receiver Design

Different DM algorithms can be stored in memory and the Voice Processor (i.e. the microprocessor) will select the proper receiver based on a code identifying the transmitter. Suppose that a 1010101010 is assigned to the CVSD algorithm and a 1100110011 to the SVADM algorithm. Figure 2.5-1 is a flow chart implementation of a receiver that selects the proper decoding algorithm.

Register R_1 is a 10 bit serial shift left register storing the incoming e(k) information sent by a transmitter. When the content of R_1 [i.e. (R_1)] agrees with either the content of R_8 or R_9, which contain the codes for the receivers, the LSB of R_0 will be set if the CVSD receiver is selected [i.e. $[(R_9)=(R_1)]$ and will reset if the SVADM receiver is selected [i.e. $(R_8)=(R_1)$]. After switching has been performed registers R_8 and R_9 fulfilled their task and they can be used for other purposes.

The problem with such a system is that the transmitter cannot send useful information until the receiver locks on. Hence the receiver has to signal the transmitter when locking has occured. To overcome such a major drawback the idle time pattern can be detected and used. The major restriction will be

that the idle time pattern of various transmitters be different.

The steady state (i.e. D-C response), or e(k), pattern of the CVSD algorithm is 10101010... and 110011001100.. for the SVADM algorithm. Since 50% of speech is silence (in zero DC) that pattern can be used to detect and identify one of the two algorithms.

If the lowest audio frequency is 500Hz, and the maximum sampling frequency 40kHz then the number of bits per half period is 40. This number is an upper limit if the waveform that the DM is tracking is a square wave. Since voice is of a sine wave form 20 e(k)'s will be sufficient to be compared with and used for an identification of a transmitter. The transmitter does not have to send a header, and the receiver will lock on the steady state pattern. A flow chart implementation of such a scheme is shown in Fig.2.5-2 with program listing in appendix F. Registers R_1 and R_2 are 10 bit serial shift left registers storing the incoming e(k) information sent by a transmitter. When the content of R_1 and R_2 agree with the content of R_9 the CVSD receiver will be decided upon. When the content of R_1 agrees with R_8 and R_2 with R_7 then the SVADM receiver will be

selected. The LSB of R_o will be set indicating true detection, and the MSB of R_o will be set if the CVSD receiver is selected and will reset if the SVADM receiver is selected. After decision has been made registers R_2 R_7 R_8 and R_9 can be used for other purposes.

An interesting problem arises now, and will be left for future research, as to whether we must store every conceivable algorithm or whether we can "learn", by receiving some prescribed "test-pattern", the algorithm employed.

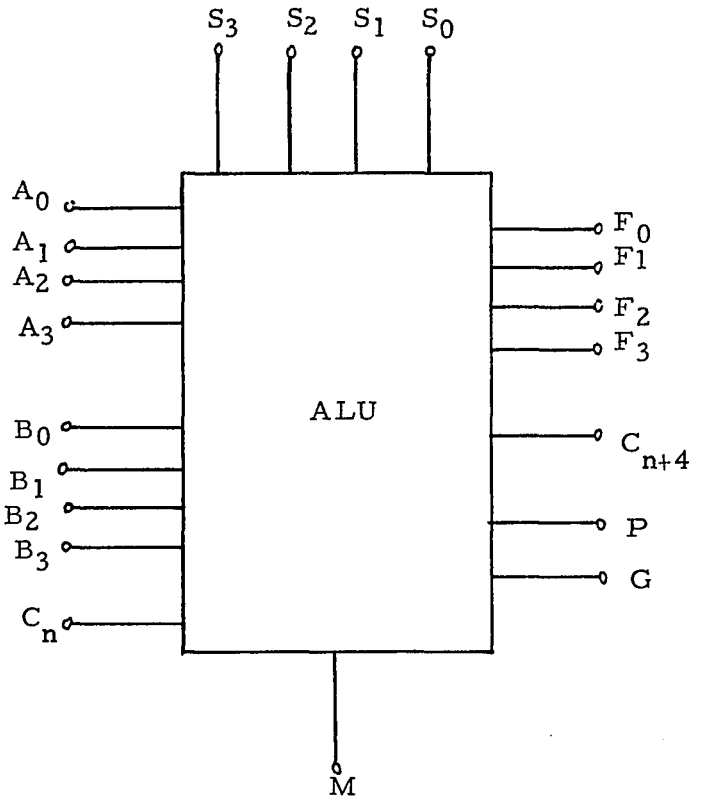


Fig. 2. 2-1: Representation of an Arithmetic Logic Unit.

Function select				Logic functions (M=1)	Arithmetic and logic (M=0)
S_3	S_2	S_1	S_0	F	F
0	0	0	0	A	A minus 1
0	0	0	1	$A \cdot B$	$(A \cdot B)$ minus 1
0	0	1	0	$A \cdot \overline{B}$	$(A \cdot \overline{B})$ minus 1
0	0	1	1	0	minus 1 (in two's complement
0	1	0	0	A+B	A plus (A + B)
0	1	0	1	B	(A + B) plus $(A \cdot B)$
0	1	1	0	A+B	A minus B minus 1
0	1	1	1	$A \cdot B$	A+B
1	0	0	0	A+B	A plus (A+B)
1	0	0	1	$\overline{A+B}$	A plus B
1	0	1	0	$\overline{B}$	(A+B) plus $(A \cdot \overline{B})$
1	0	1	1	$\overline{A} \cdot \overline{B}$	A+B
1	1	0	0	1	A times 2
1	1	0	1	$\overline{A}+B$	A plus $(A \cdot B)$
1	1	1	0	$\overline{A}+\overline{B}$	A plus $(A \cdot \overline{B})$
1	1	1	1	A	A

Table 2.2 : Functional description of ALU operations

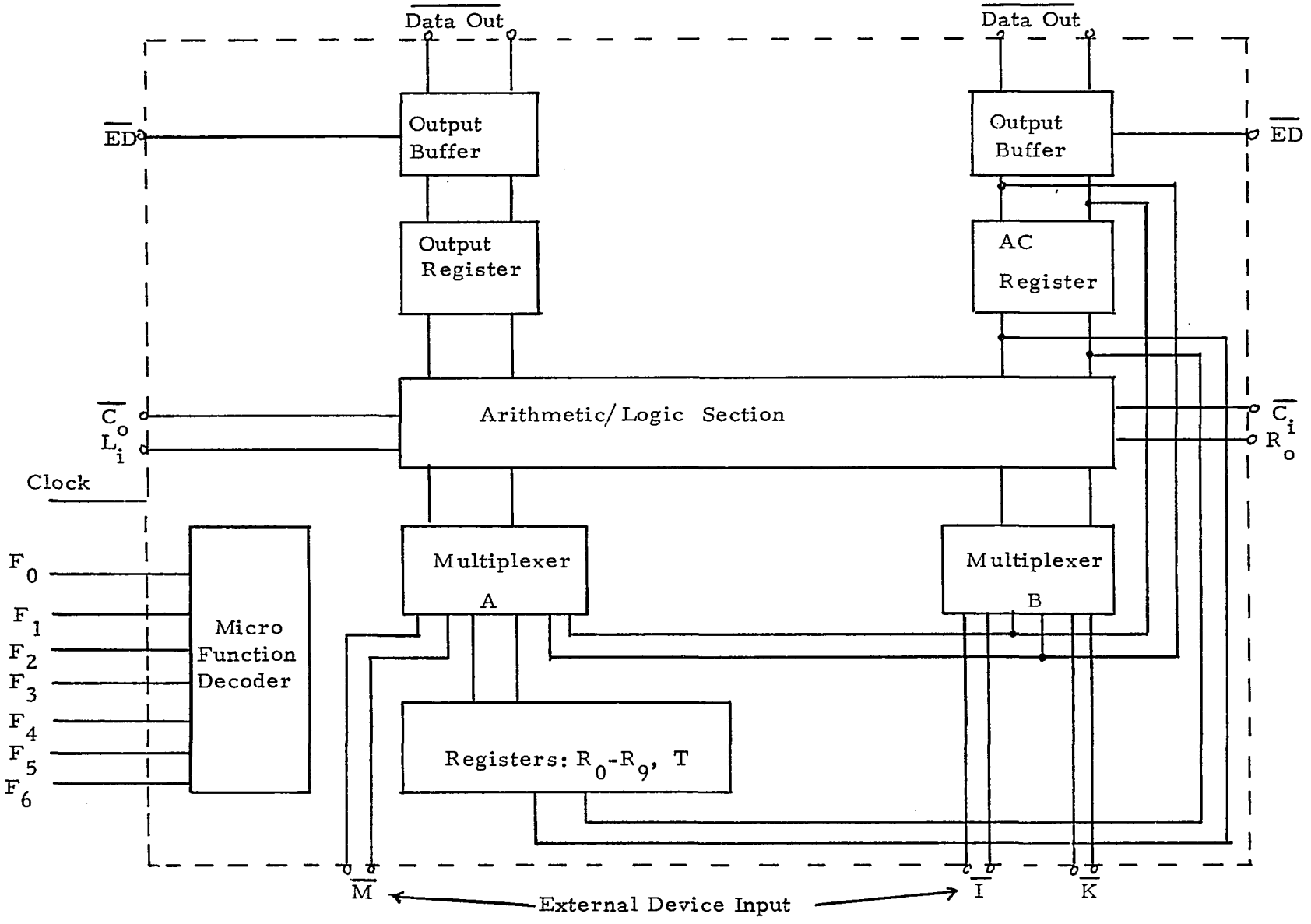


Fig. 2.2-2: Intel 3002 2-Bit CPU Slice.

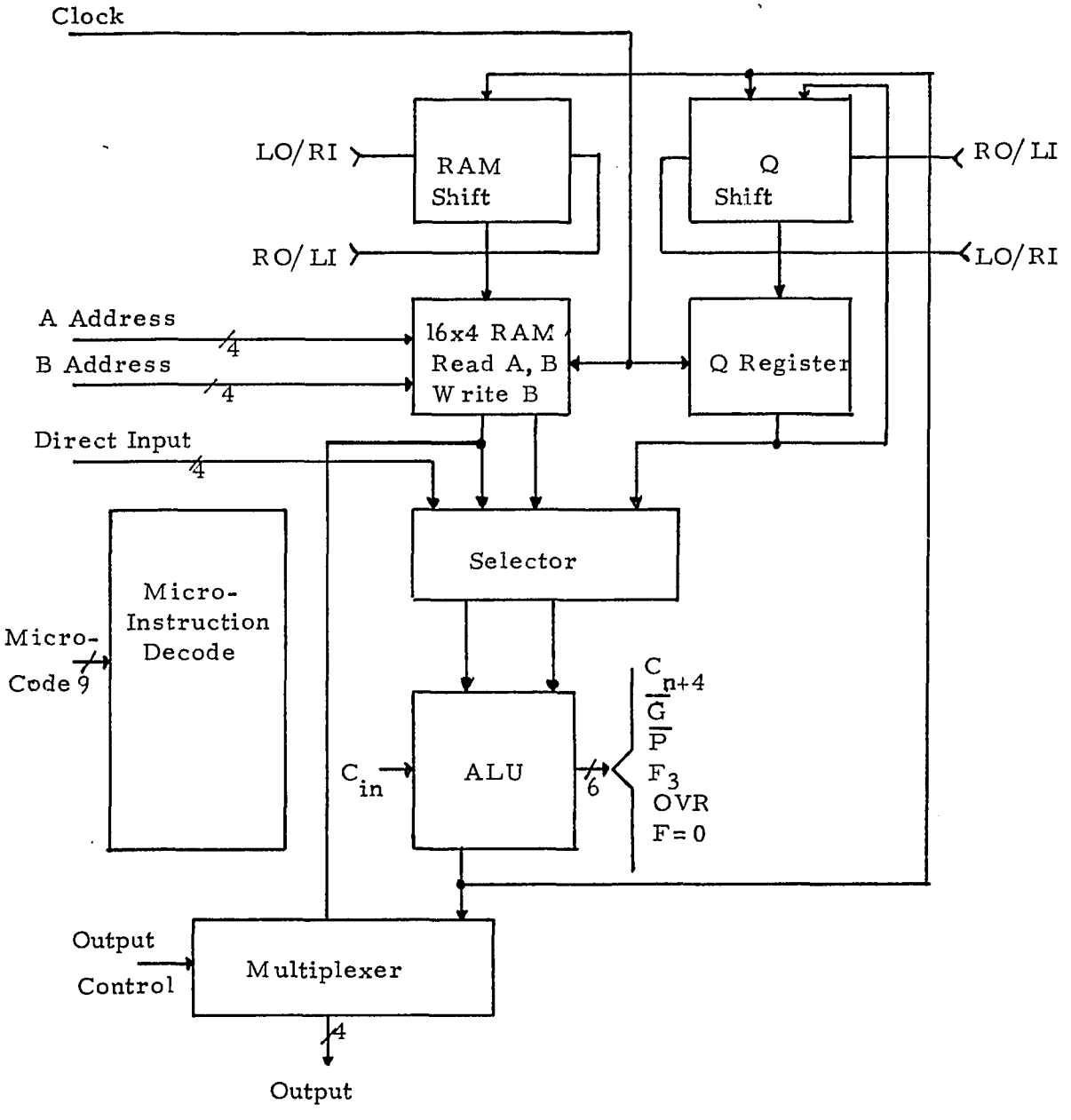


Fig. 2.2-3: AMD 2901 4-Bit CPU Slice.

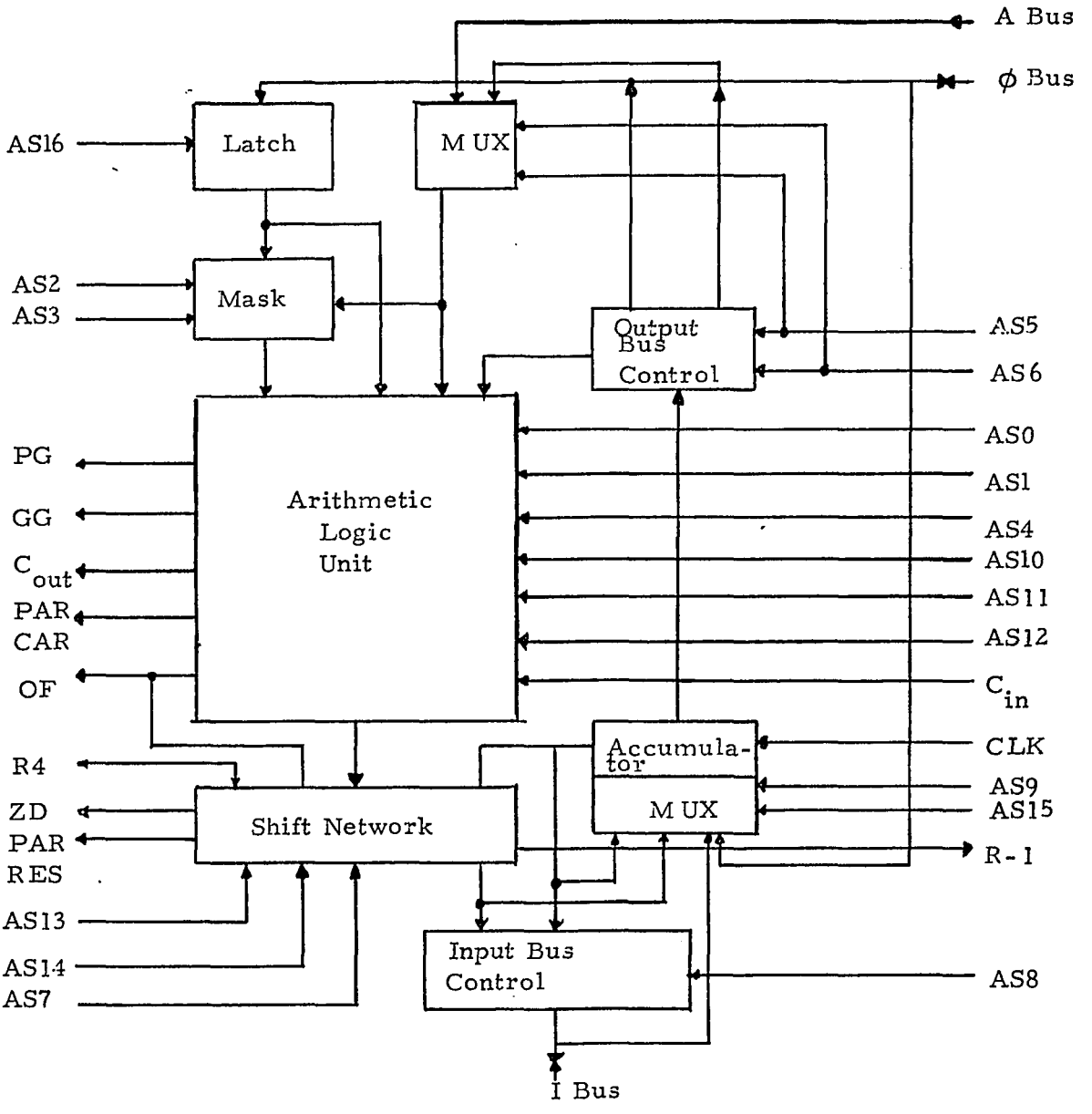


Fig. 2.2-4: MC10800 4-Bit ALU Slice.

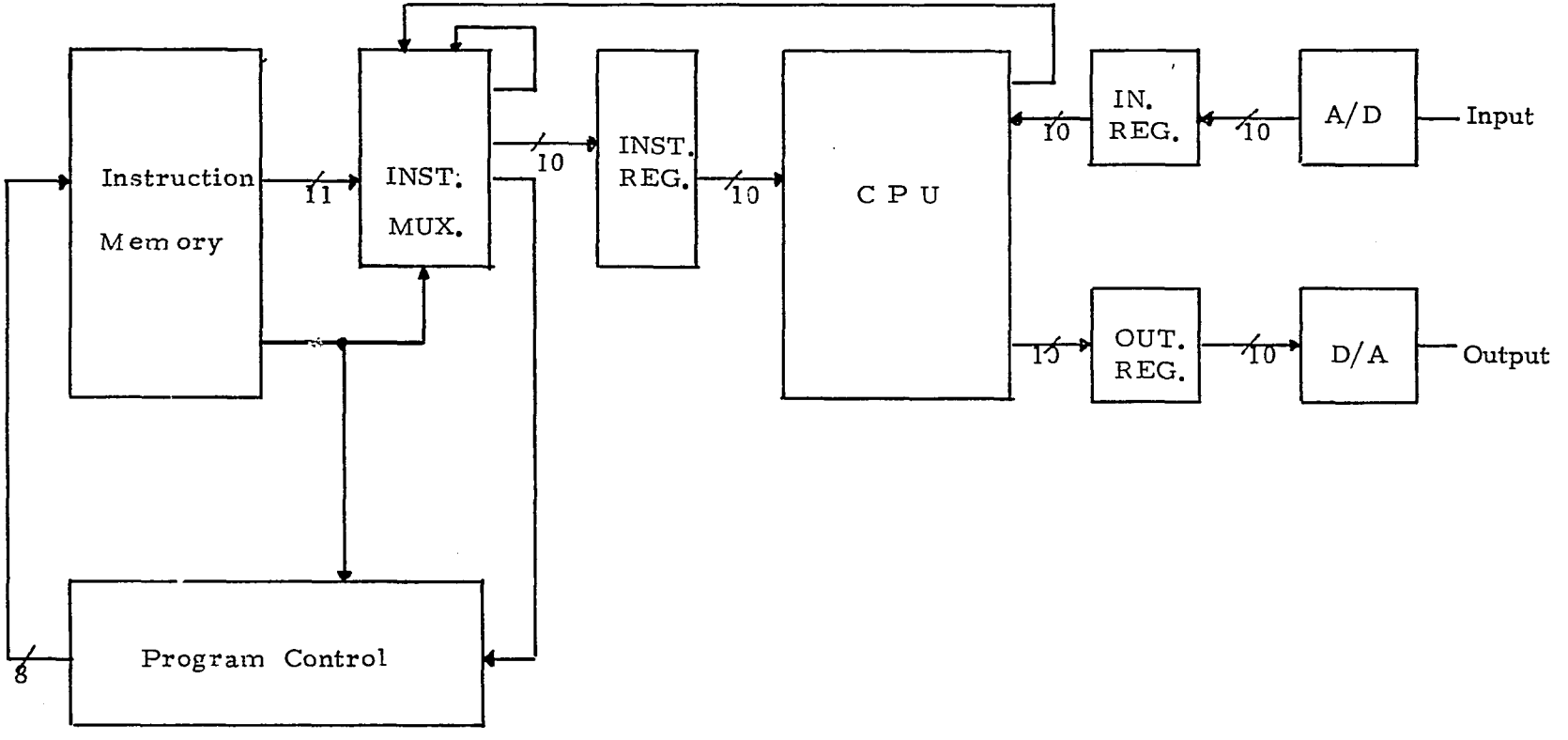


Fig. 2.2-5: Voice Processor Block Diagram.

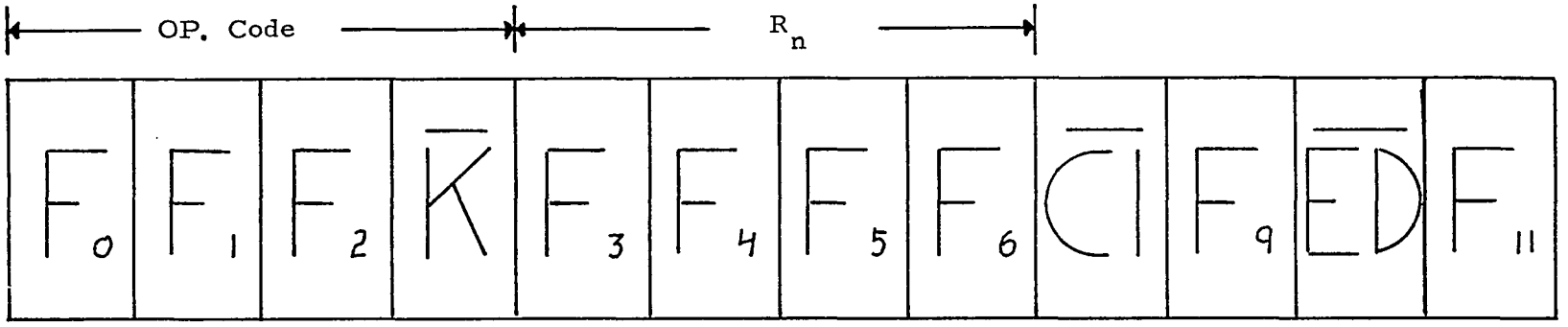


Fig. 2.2-6: Instruction Format.

MNEMONIC	$\overline{K}$ - BUS = 1	MNEMONIC	$\overline{K}$ - BUS = 0
ILR	$R_n + CI \rightarrow R_n, AC$	ALR	$AC + R_n + CI \rightarrow R_n, AC$
ACM	$M + CI \rightarrow AT$	AMA	$M + AC + CI \rightarrow AT$
SRA	$\overline{AT}_L \rightarrow RO$, $AT_H \rightarrow AT_L$ $LI \rightarrow AT_H$	-	
LMI	$R_n \rightarrow MAR$, $R_n + CI \rightarrow R$	DSM	$11 \rightarrow MAR$ $R_n - 1 + CI \rightarrow R_n$,
LMM	$M \rightarrow MAR$, $M + CI \rightarrow AT$	LDM	$11 \rightarrow MAR$ $M - 1 + CI \rightarrow AT$
CIA	$\overline{AT} \rightarrow CI \rightarrow AT$	DCA	$AT - 1 + CI \rightarrow AT$
CSR	$CI - 1 \rightarrow R_n$	SDR	$AC - 1 + CI \rightarrow R_n$
CSA	$CI - 1 \rightarrow AT$	SDA	$AC - 1 + CI \rightarrow AT$
-	(See CSA above)	LDI	$I - 1 + CI \rightarrow AT$
INR	$R_n + CI \rightarrow R_n$	ADR	$AC + R_n + CI \rightarrow R_n$
	(See ACM above)	-	(See AMA above)
INA	$AT + CI \rightarrow AT$	AIA	$I + AT + CI \rightarrow AT$
CLR	$CI \rightarrow CO$ $0 \rightarrow R_n$	ANR	$CI \vee (R_n \wedge AC) \rightarrow CO$, $R_n \wedge AC \rightarrow R_n$
CLA	$CI \rightarrow CO$ $0 \rightarrow AT$	ANM	$CI \vee (M \wedge AC) \rightarrow CO$, $M \wedge AC \rightarrow AT$
-	(See CLA above)	ANI	$CI \vee (AT \vee I) \rightarrow CO$, $AT \wedge I \rightarrow AT$
-	(See CLR above)	TZR	$CI \vee R_n \rightarrow CO$ $R_n \rightarrow R_n$
-	(See CLA above)	LTM	$CI \vee M \rightarrow CO$ $M \rightarrow AT$
-	(See CLA above)	TZA	$CI \vee AT \rightarrow CO$ $AT \rightarrow AT$
NOP	$CI \rightarrow CO$, $R_n \rightarrow R_n$	ORR	$CI \vee AC \rightarrow CO$ $R_n \vee AC \rightarrow R_n$
LMF	$CI \rightarrow CO$, $M \rightarrow AT$	ORM	$CI \vee AC \rightarrow CO$ $M \vee AC \rightarrow AT$
-	(See NOP above)	ORI	$CI \vee I \rightarrow CO$ $I \vee AT \rightarrow AT$
CMR	$CI \rightarrow CO$, $\overline{R_n} \rightarrow R_n$	XNR	$CI \vee (R_n \wedge AC) \rightarrow CO$ $R_n \overline{\oplus} AC \rightarrow R_n$
LCM	$CI \rightarrow CO$, $\overline{M} \rightarrow AT$	XNM	$CI \vee (M \wedge AC) \rightarrow CO$ $M \overline{\oplus} AC \rightarrow AT$
CMA	$CI \rightarrow CO$, $\overline{AT} \rightarrow AT$	XNI	$CI \vee (AT \wedge I) \rightarrow CO$ $I \overline{\oplus} AT \rightarrow AT$

Table 2.2-2: Microinstruction Set.

REGISTER GROUP	REGISTER	F_3	F_2	F_1	F_0
I	R_0	0	0	0	0
	R_1	0	0	0	1
	R_2	0	0	1	0
	R_3	0	0	1	1
	R_4	0	1	0	0
	R_5	0	1	0	1
	R_6	0	1	1	0
	R_7	0	1	1	1
	R_8	1	0	0	0
	R_9	1	0	0	1
	T	1	1	0	0
	AC	1	1	0	1
II	T	1	0	1	0
	AC	1	0	1	1
III	T	1	1	1	0
	AC	1	1	1	1

Table 2.2-3: Register Group Breakdown.

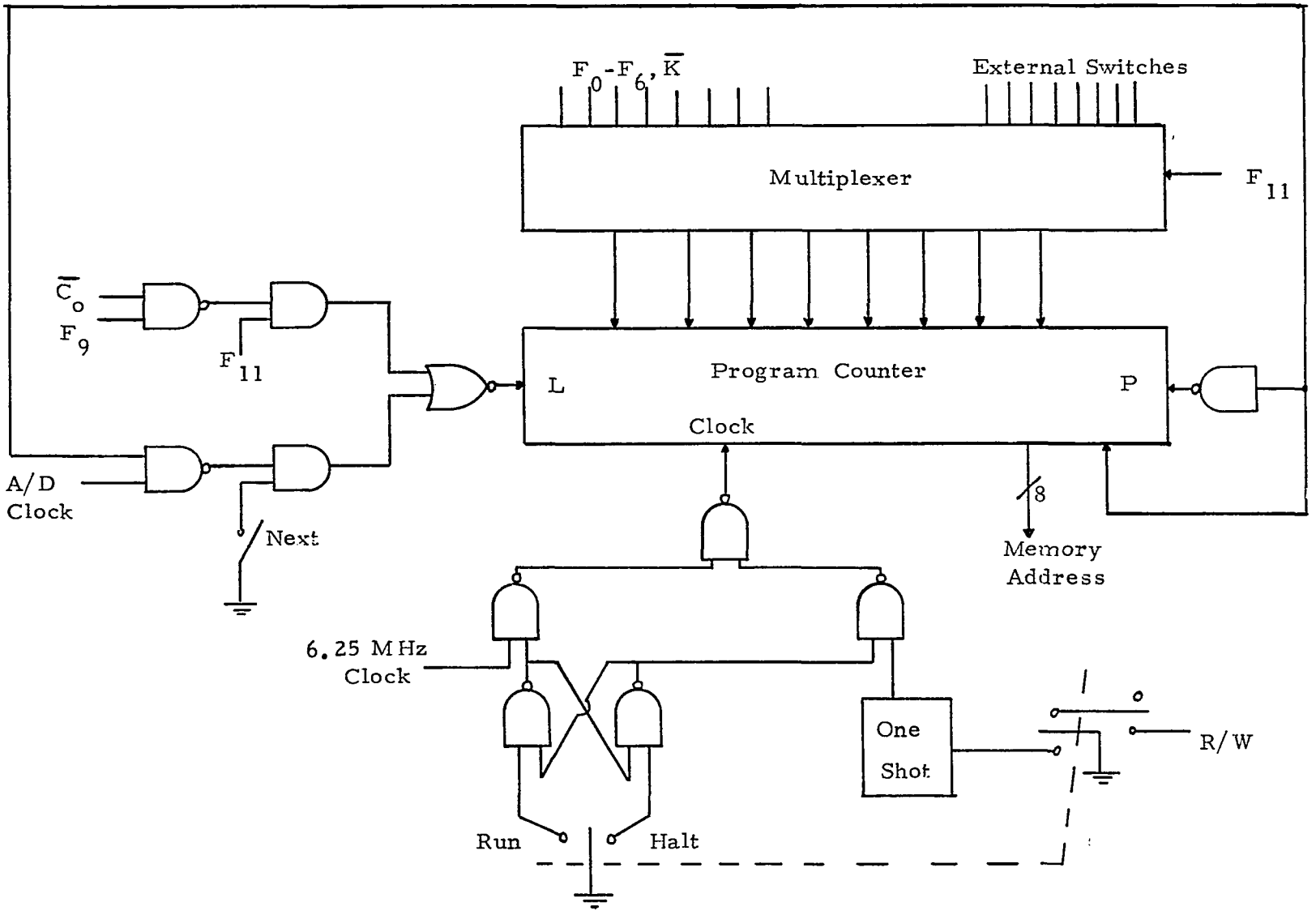
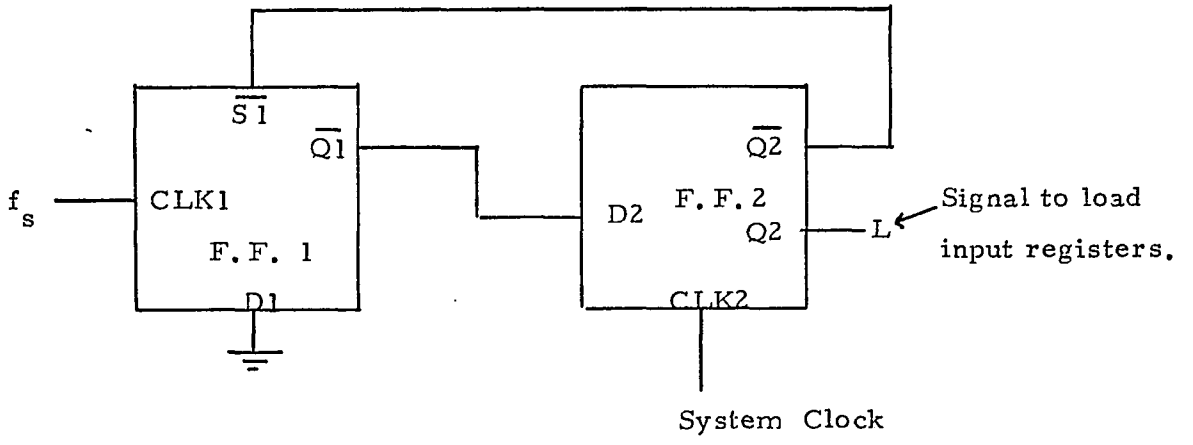
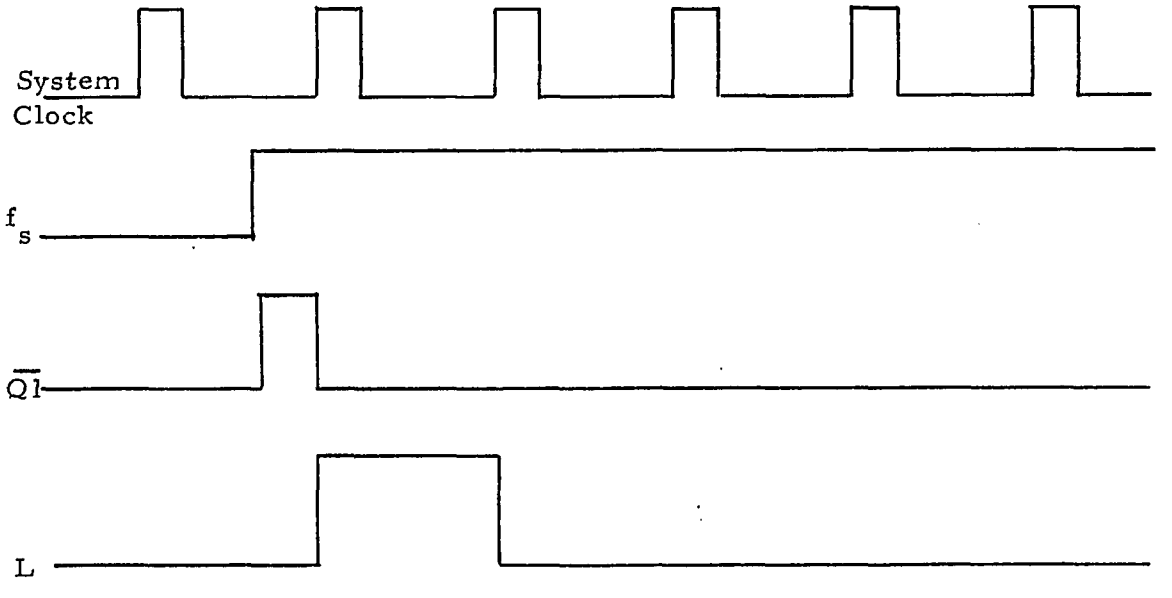


Fig. 2.2-7: Program Controller.



(a)



(b)

Fig. 2.2-8: Pulse Shaper (a) Circuit Diagram (b) Timing Diagram.

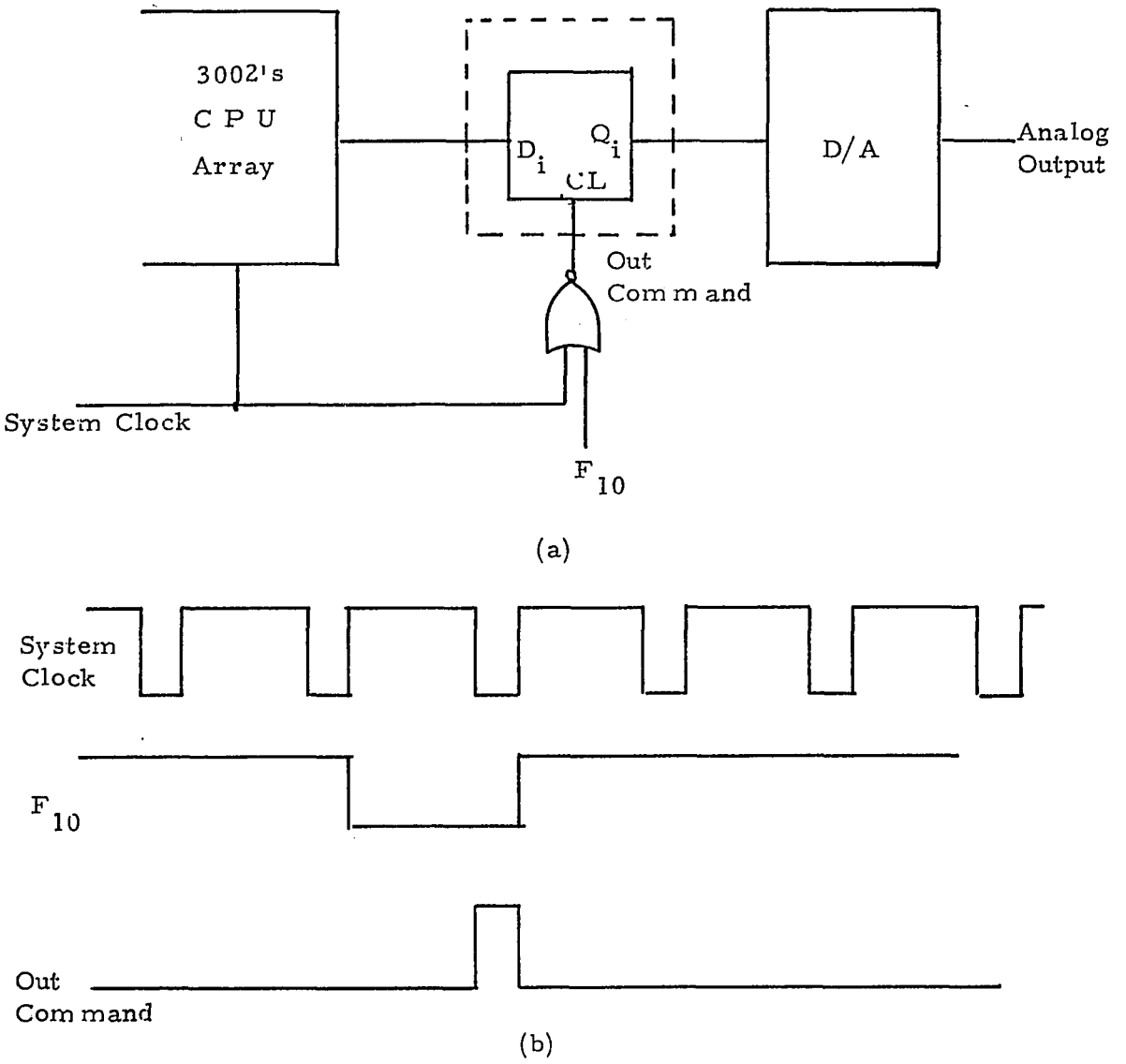


Fig. 2.2-9: (a) Output Com m and Circuitry, (b) Timing Diagram.

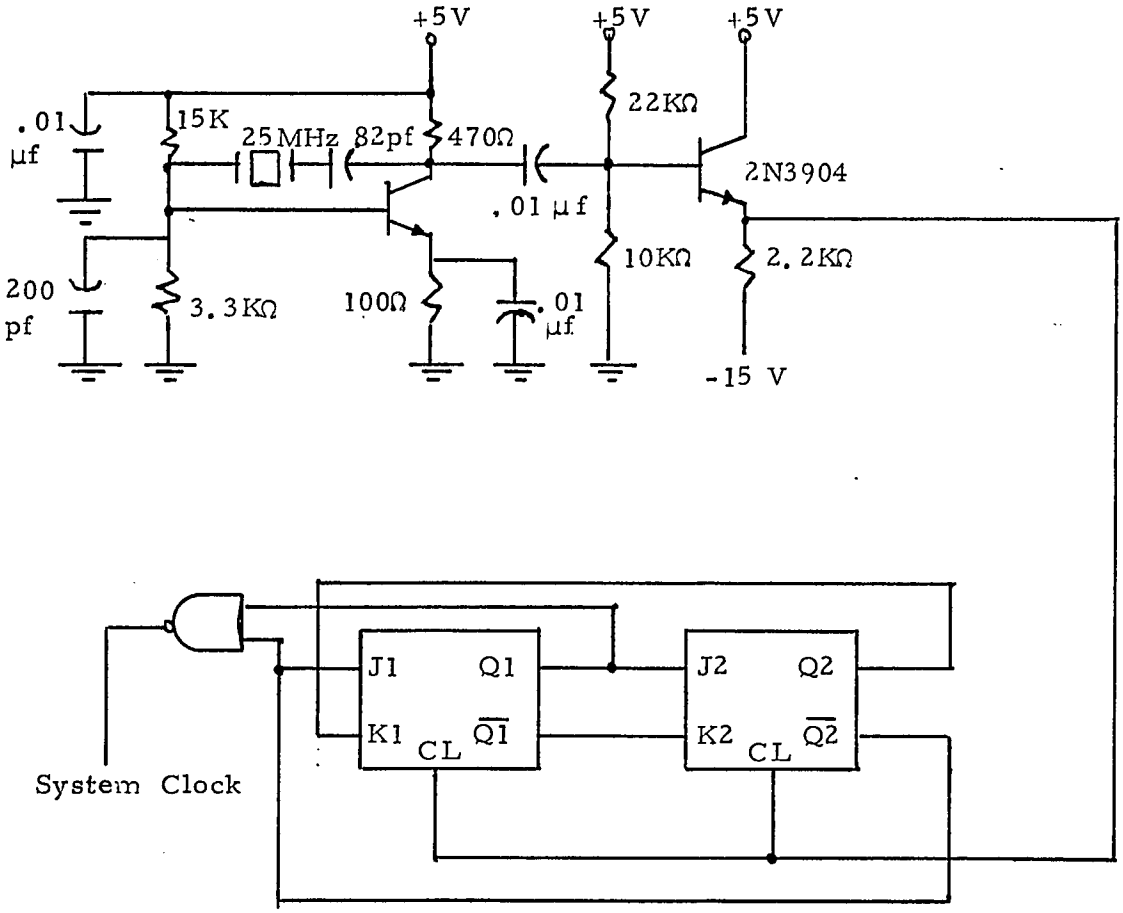


Fig. 2.2-10: Crystal Controlled Clock Circuit.

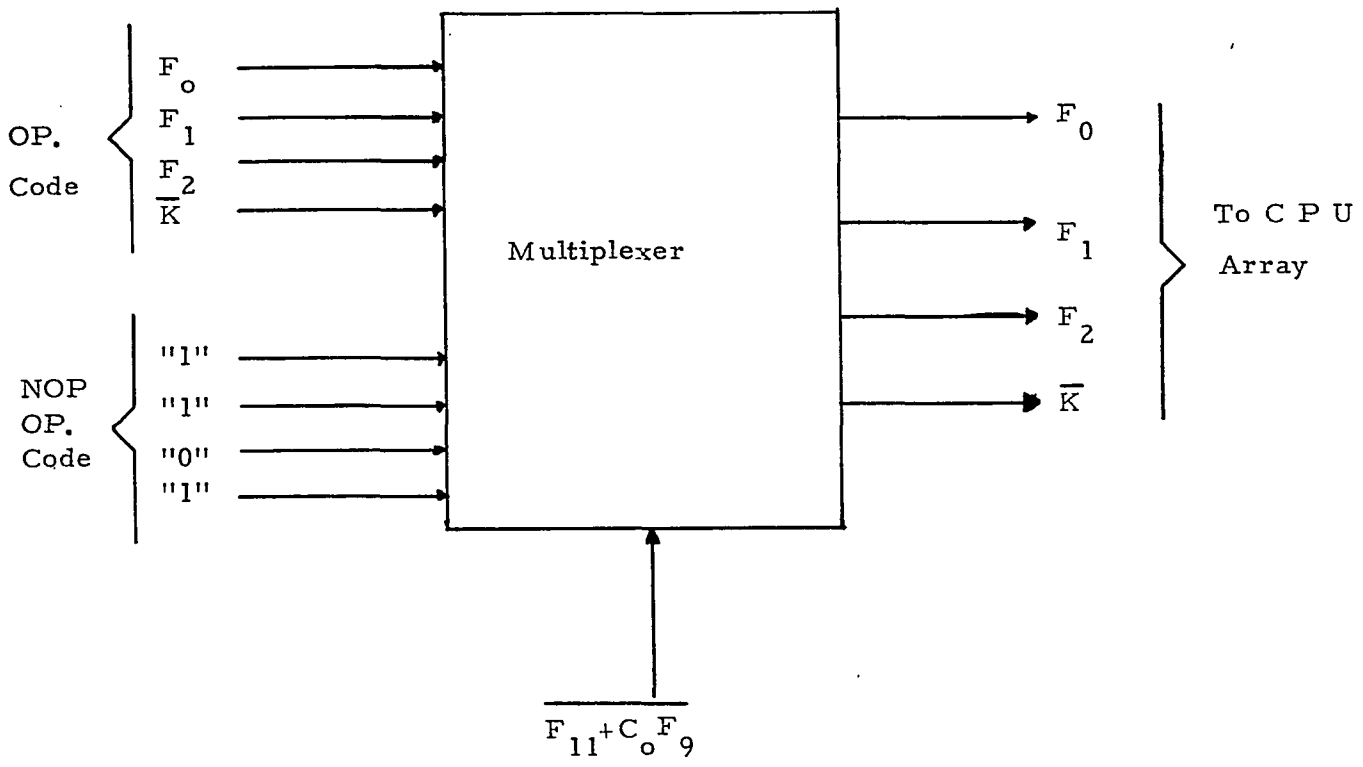


Fig. 2.2-11 : Implem entation of the Jum p Instruction.

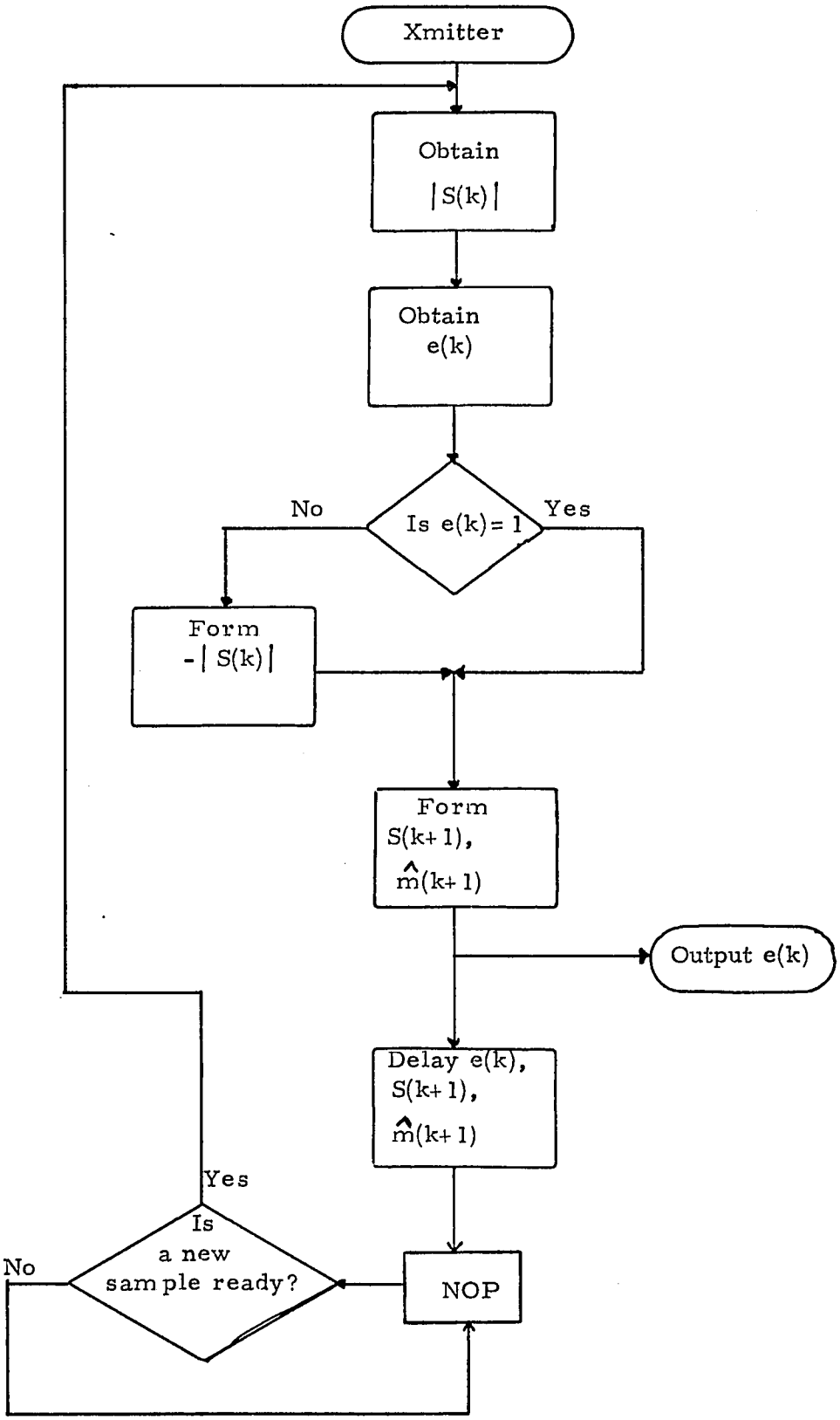


Fig. 2.3-1: Transmitter Flow Chart.

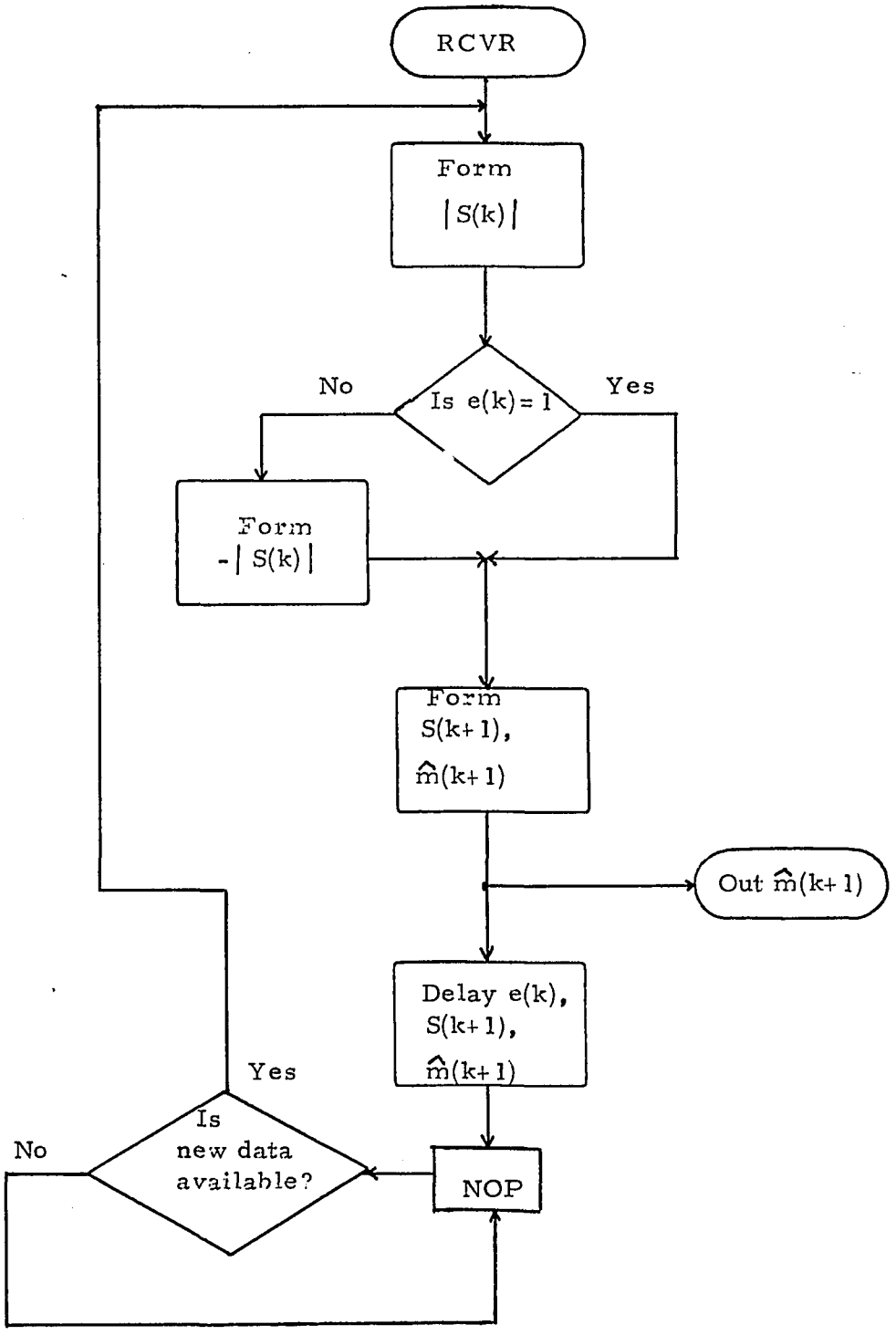


Fig. 2.3-2: Receiver Flow Chart.

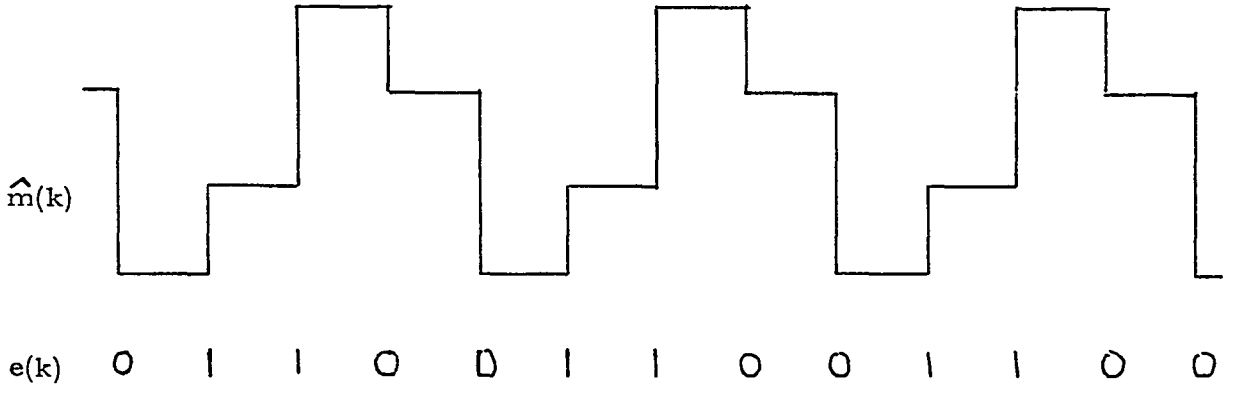


Fig. 2.3-3: Steady-State Pattern of ICDM.

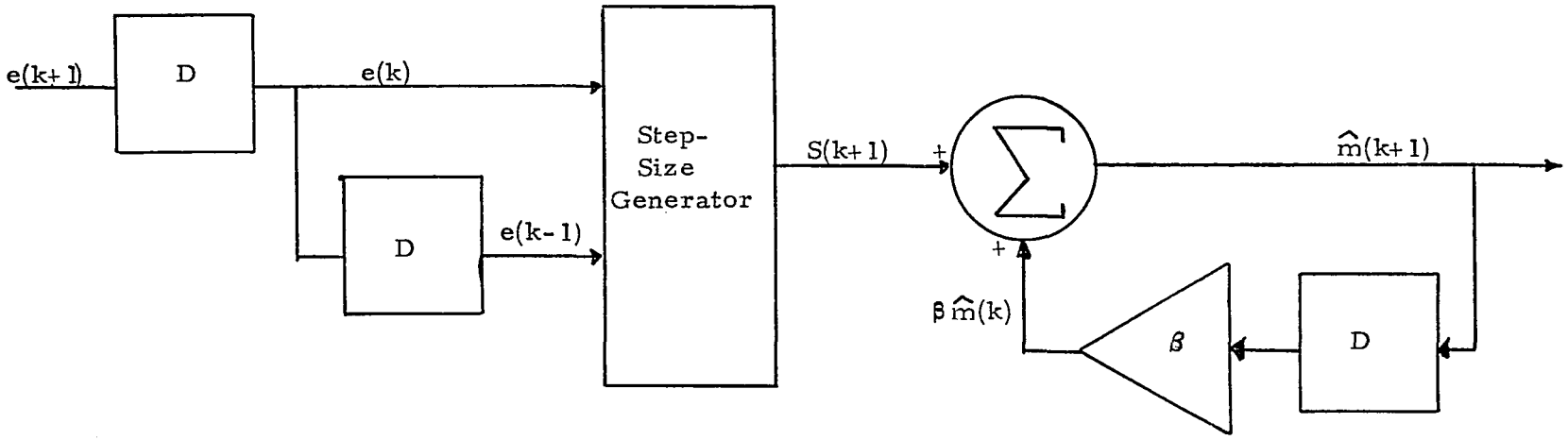


Fig. 2.3-4: Receiver Employing a Leaky Accumulator.

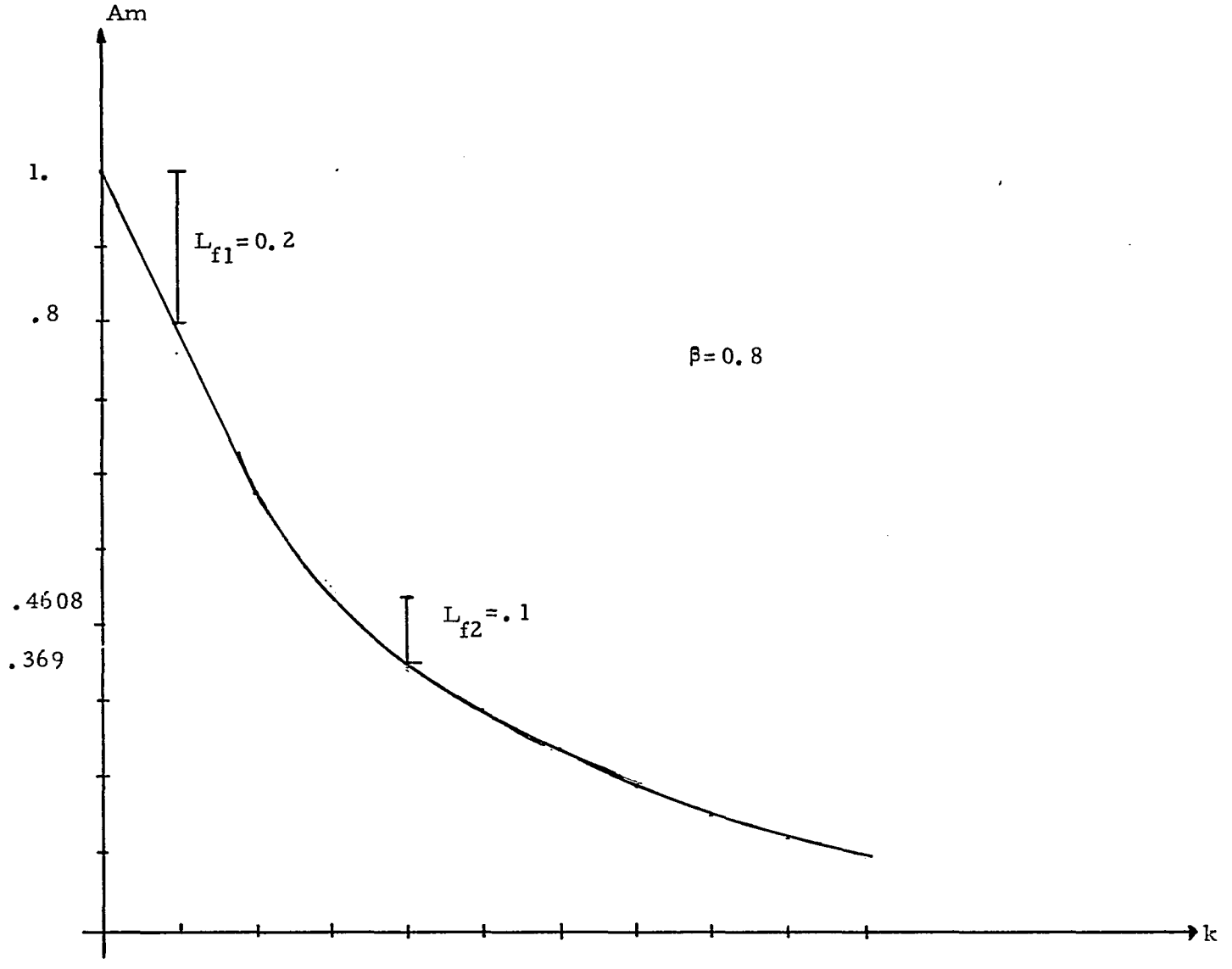


Fig. 2.3-5: Leakage as a Function of Estimate Level.

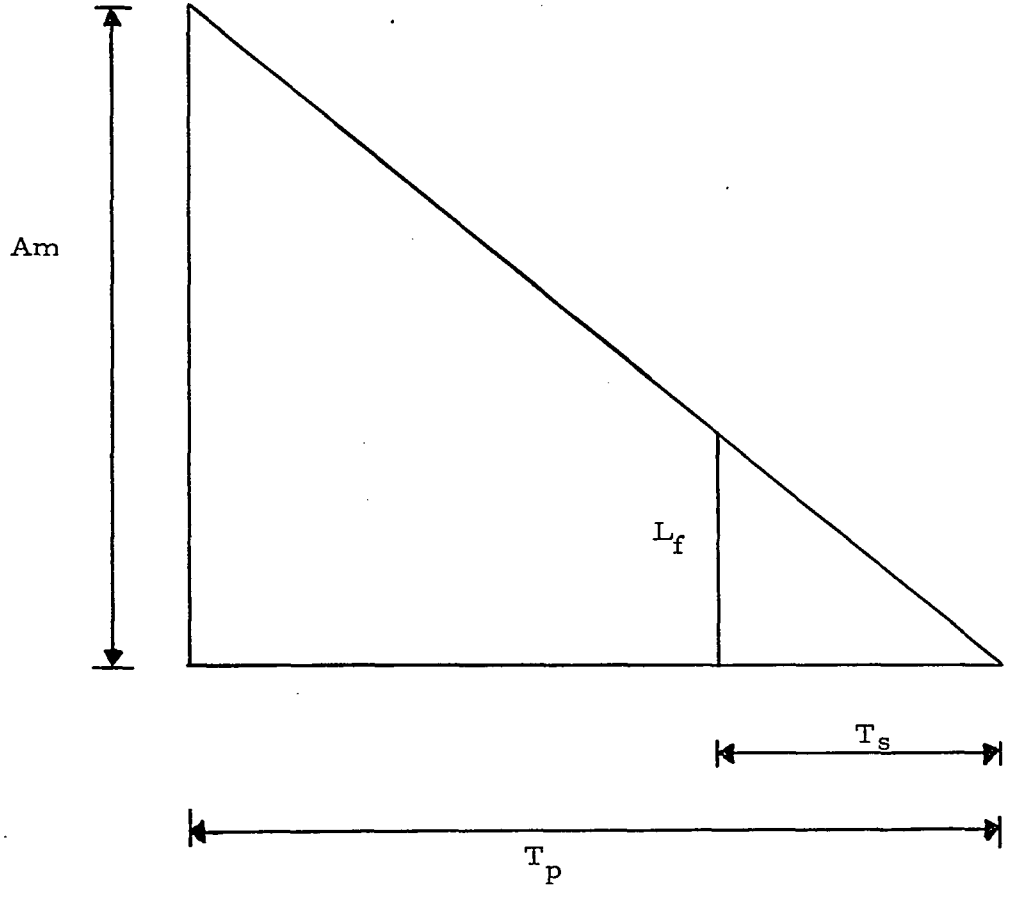


Fig. 2.3-6: Straight Line Approximation.

Fs,kHz	T_p= 100 msec L_f,mv	T_p= 10 msec L_f,mv
10	5	50
16	3.125	31.25
24	2.08	20.8
32	1.58	15.8
64	0.79	7.9

Table 2.3-1 leak factor as a function of sampling rates

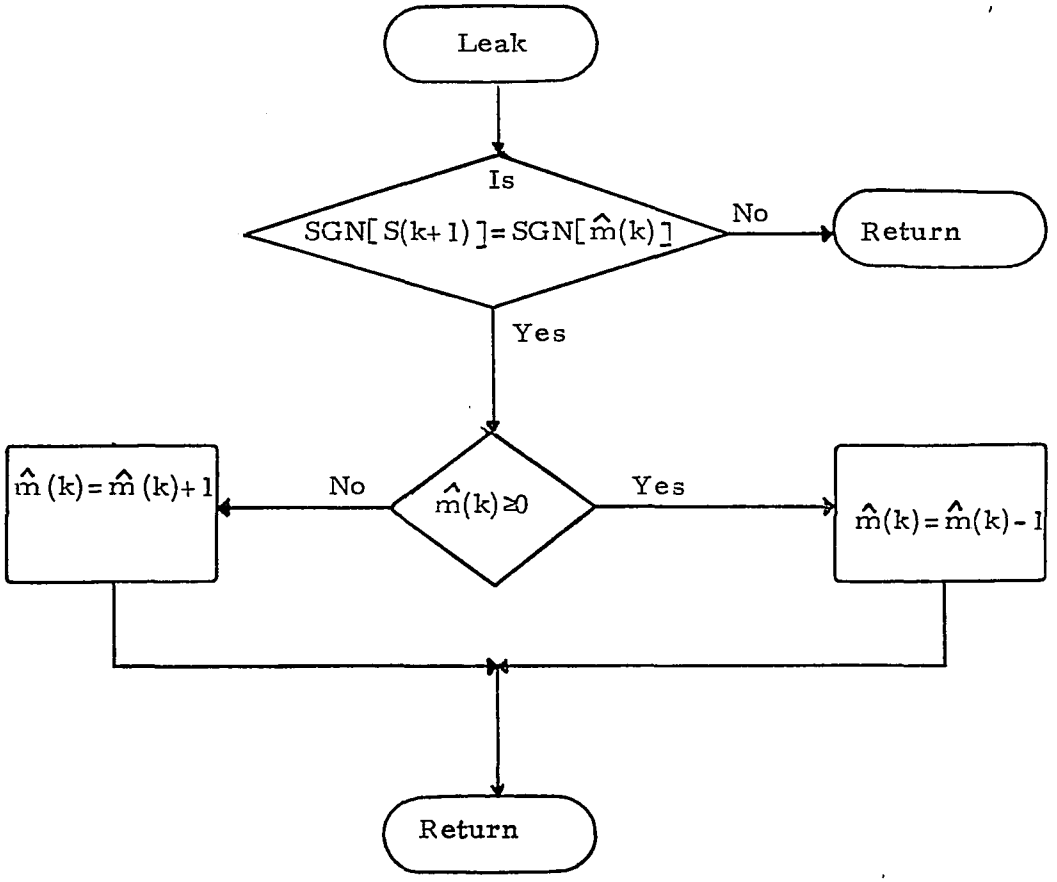


Fig. 2.3-7: Leaky Integrator Flow Chart.

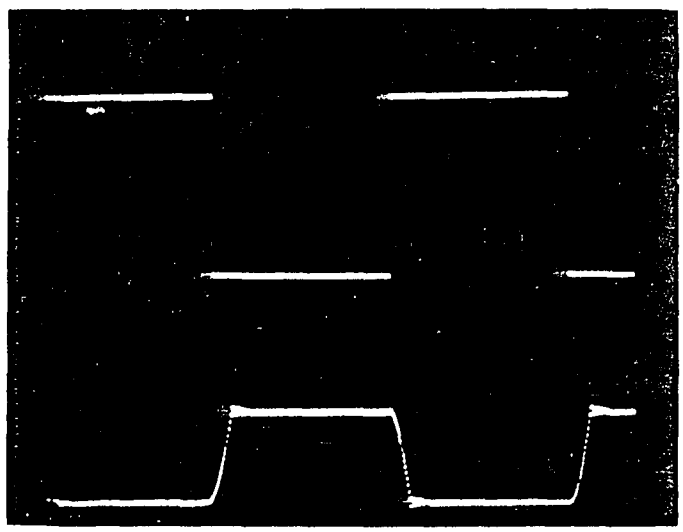


Fig. 2.3-8:Response Of SVADM Not Employing Leak Logic.

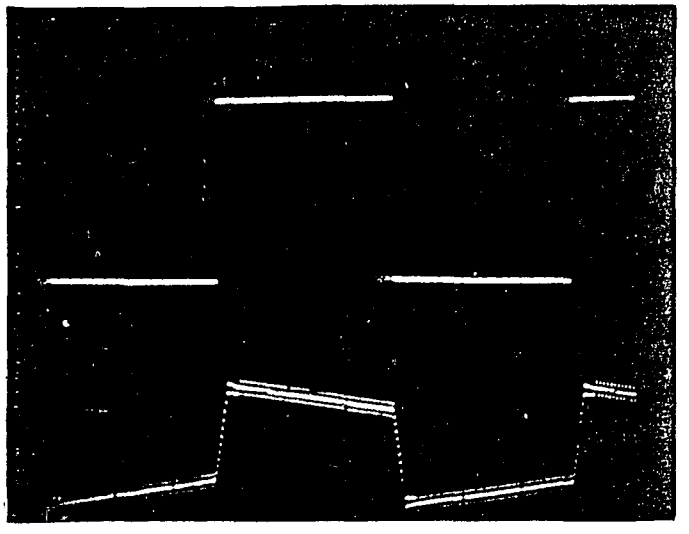


Fig. 2.3-9: Response Of SVADM Employing Leak Logic.

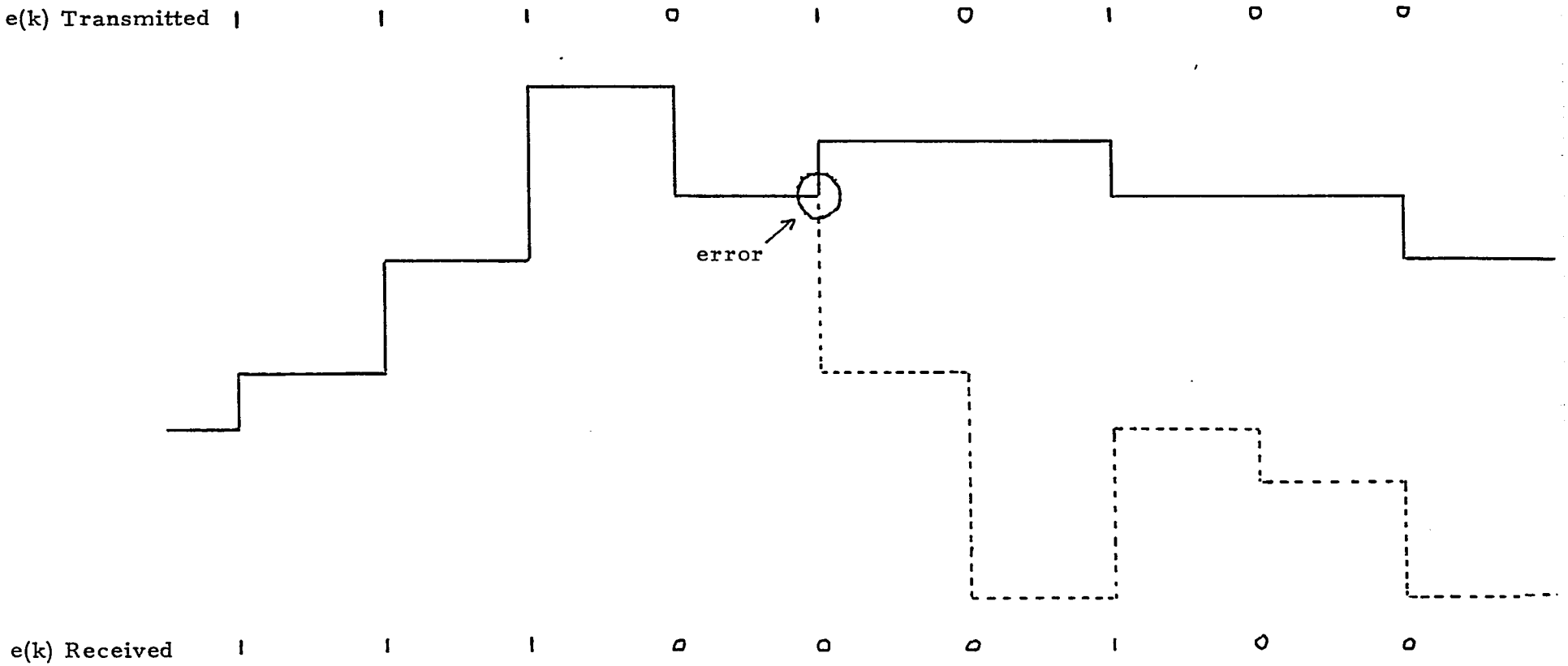
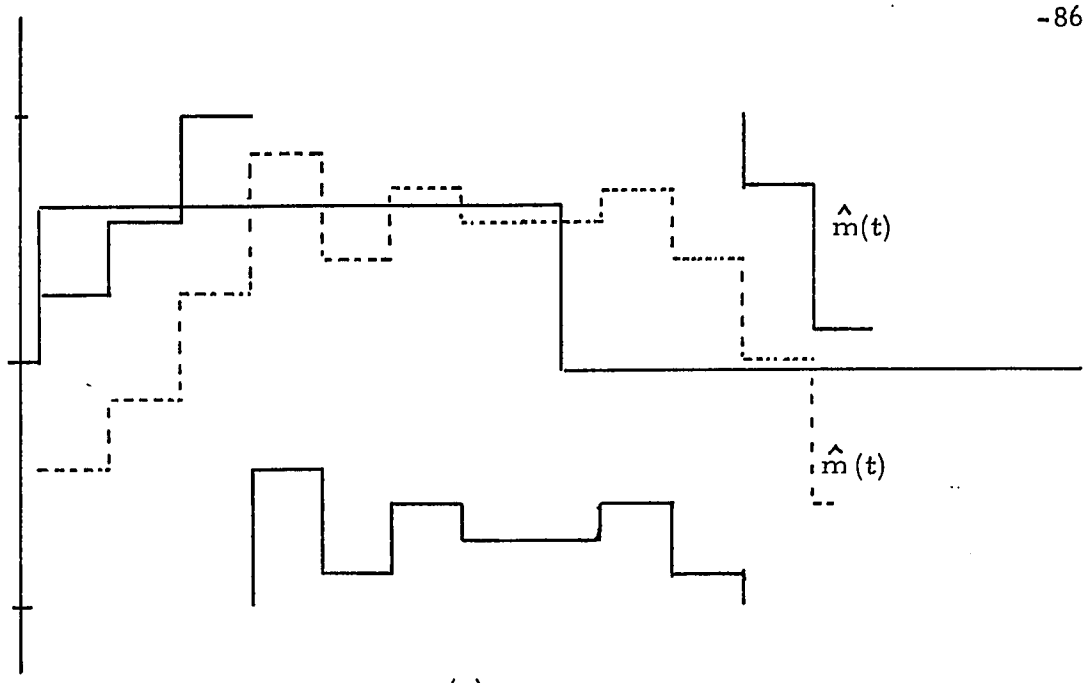
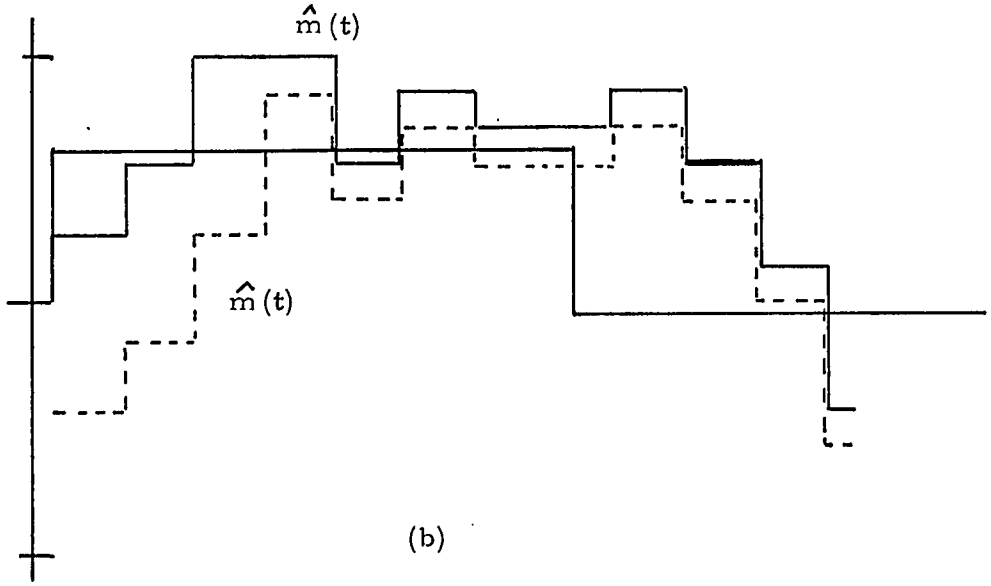


Fig. 2.3-10: Effect of Channel Errors on DM Performance.



(a)



(b)

Fig. 2.3-11:SVADM Response(a)Without Saturation Logic, (b)With Saturation Logic.

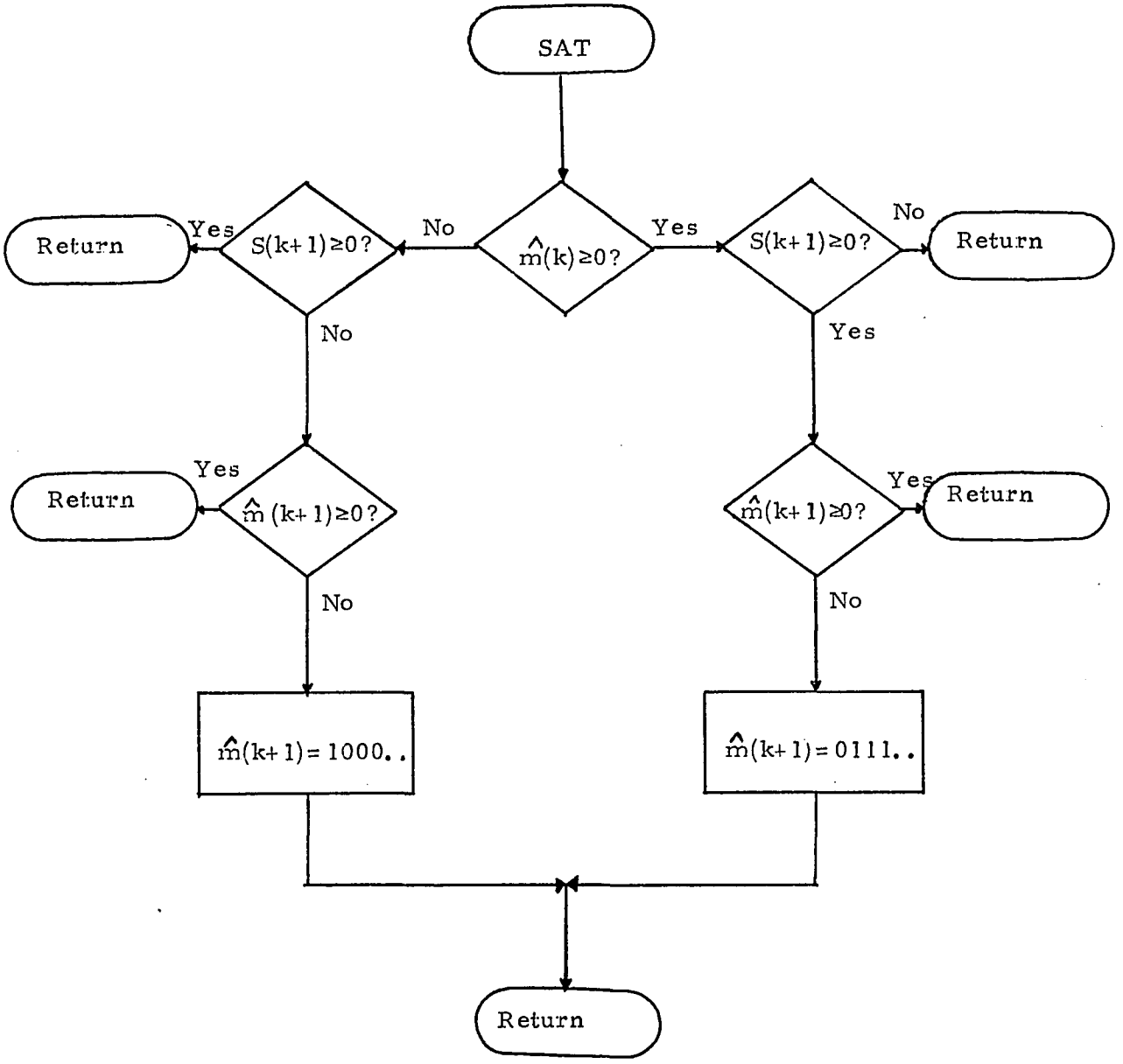
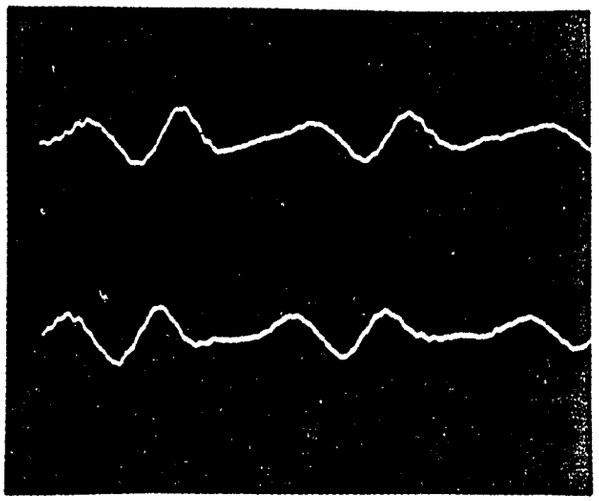


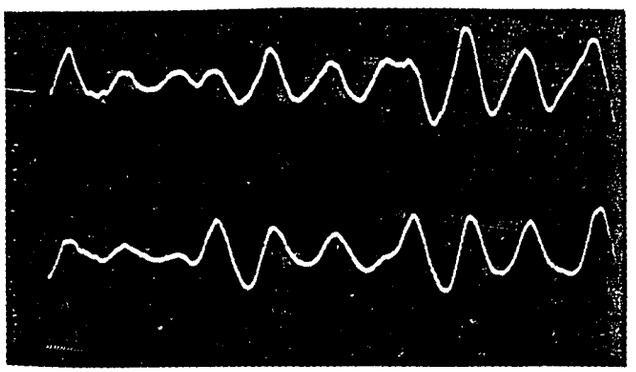
Fig. 2.3-12: Saturation Logic Flow Diagram.



Input

(a)

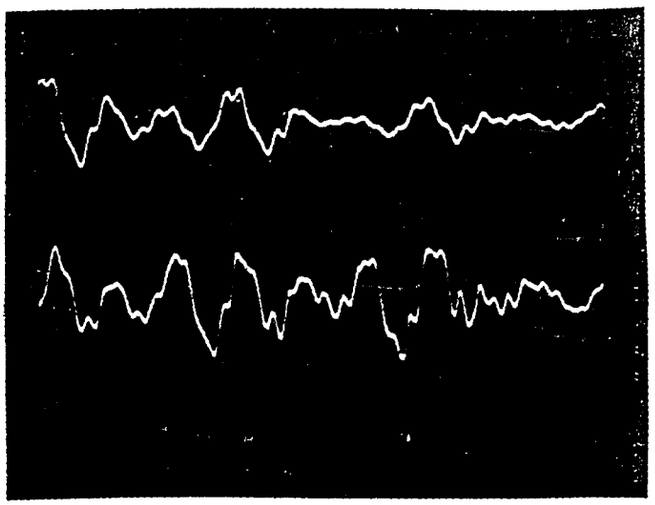
Estim ate



Input

(b)

Estim ate

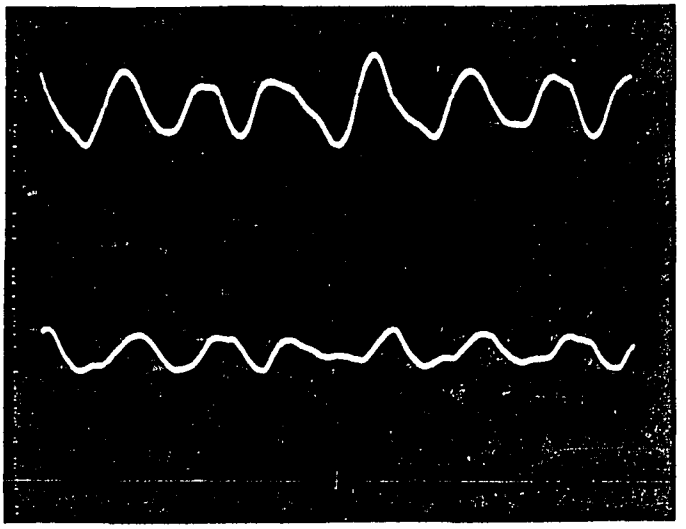


Input

(c)

Estimate

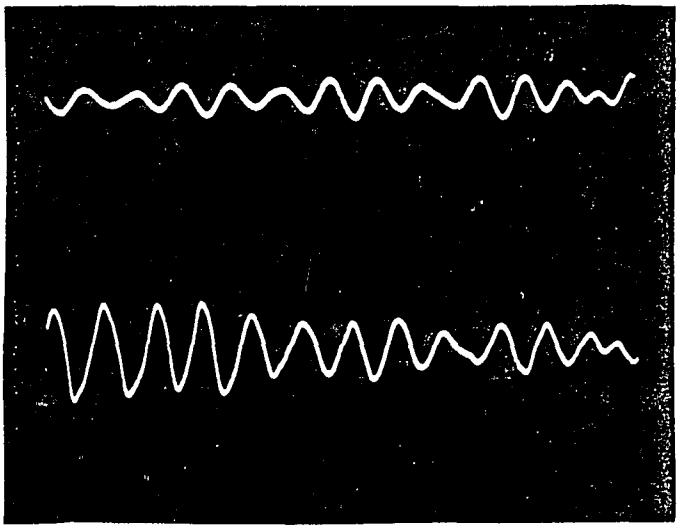
Fig. 2.3-13:Effect Of Errors On SVADM Not Employing Leak And Saturatic Logic;(a)No Errors, (b) 10^{-3} Error Rate, (c) 10^{-2} Error Rate.



Input

(a)

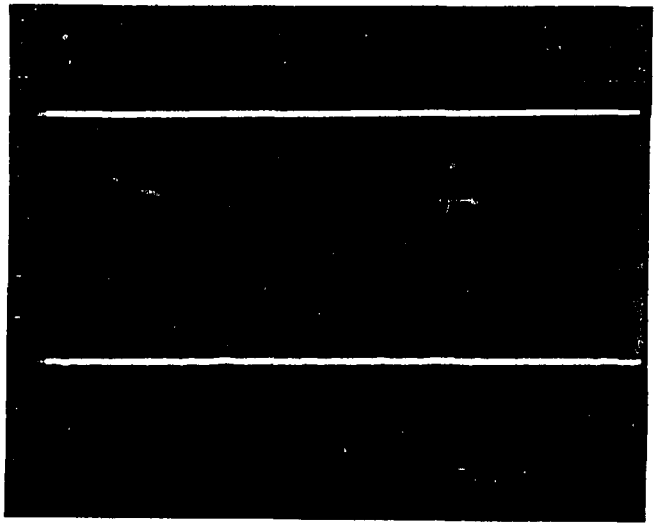
Estim ate



Input

(b)

Estimate

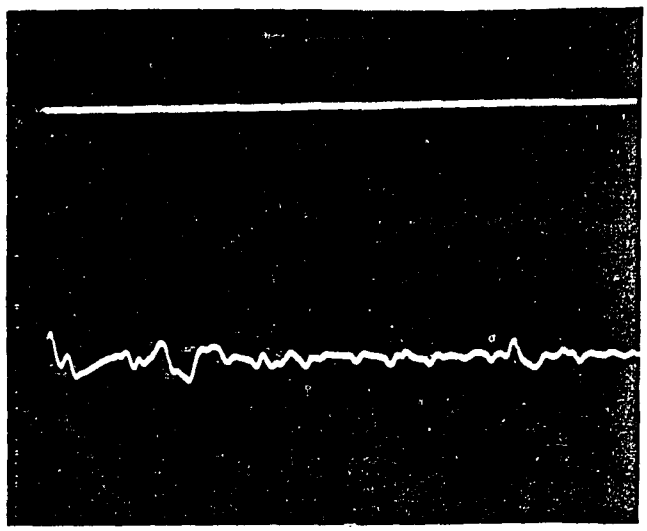


Input

(c)

Estimate

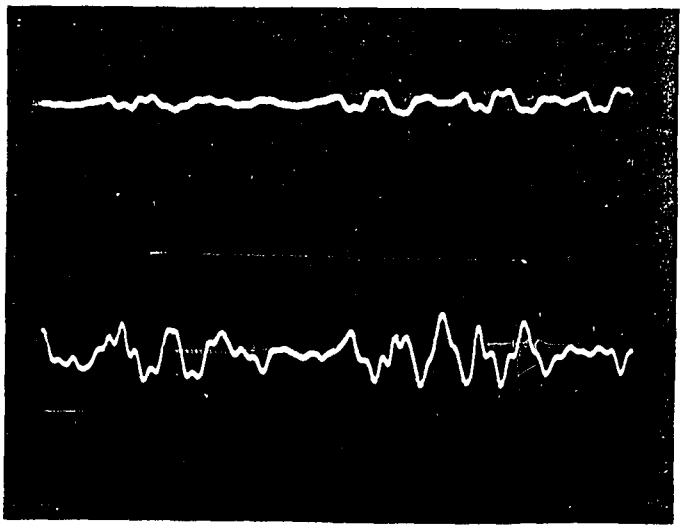
Fig. 2.3-14: Response of SVADM In Channel Errors Environment;(a) 10^{-3} Error Rate, (b) 10^{-2} Error Rate, (c) 10^{-2} Error Rate With No Input,



Input

(d)

Estimate



Input

(e)

Estimate

Fig. 2.3-14: Continued, (d) 10^{-1} Error Rate With No Input, (e) 10^{-1} Error Rate With Input.

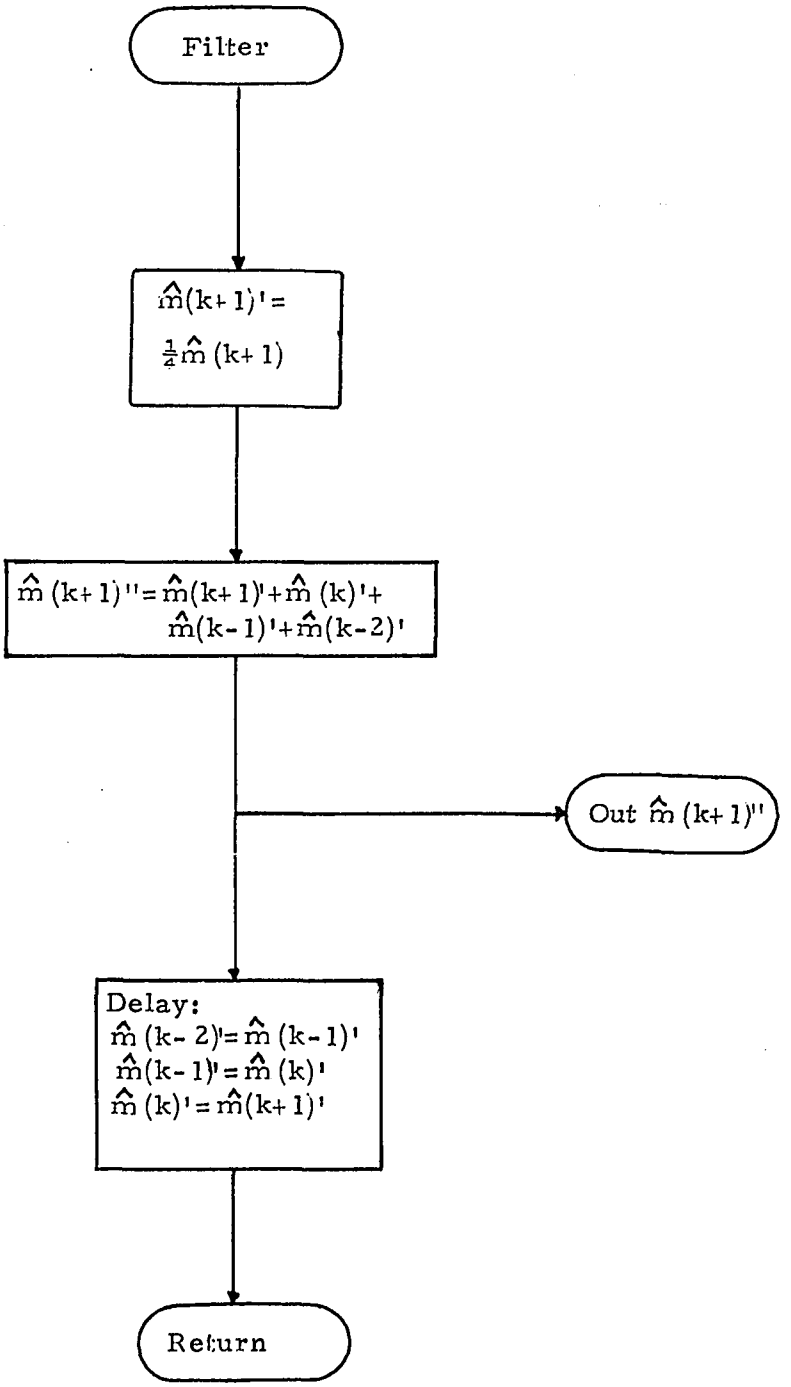


Fig. 2.3-15: Averaging Filter Flow Chart.

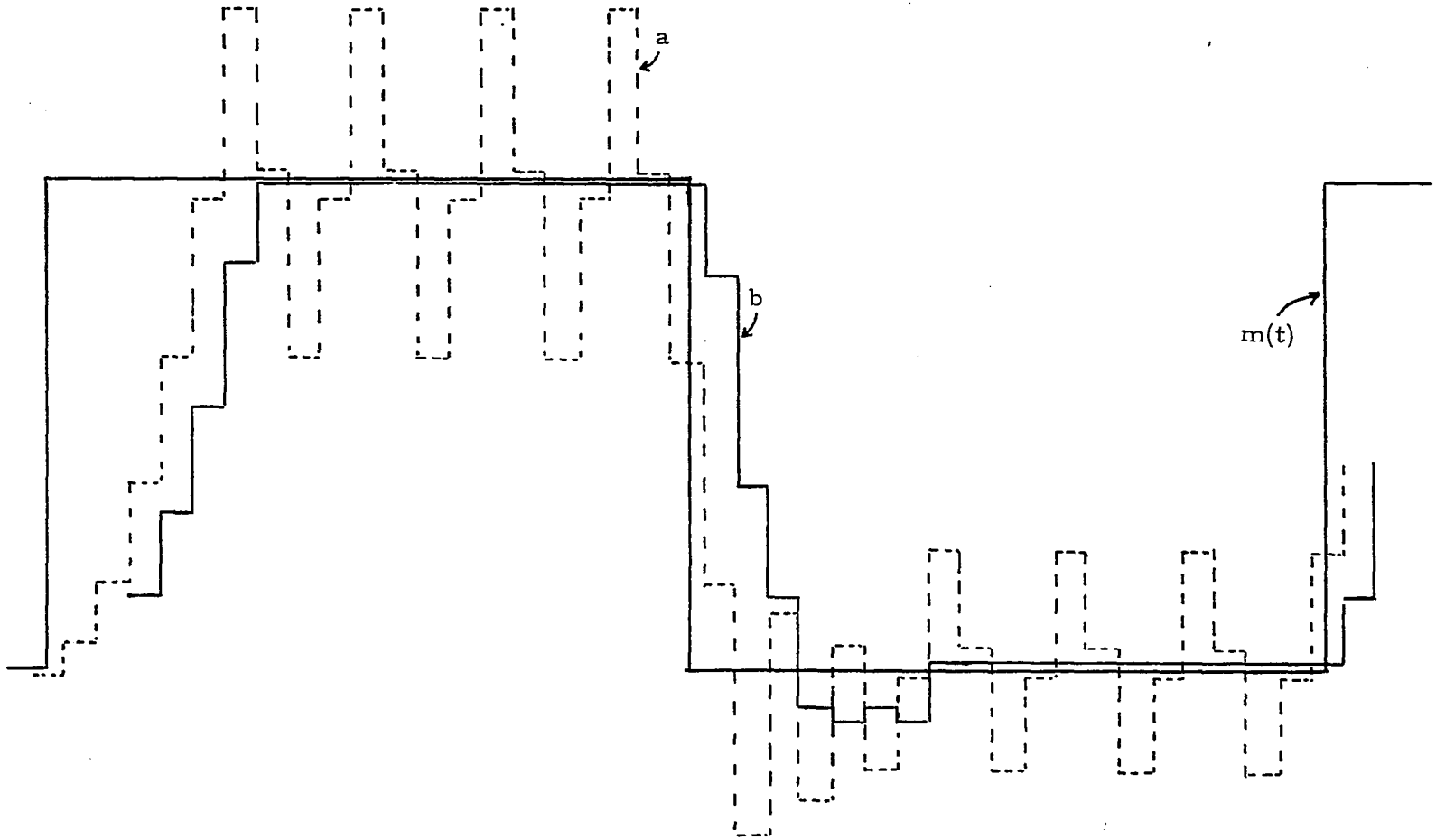


Fig. 2.3-16: Response of ADM , At 8 KHz Sampling Rate, (a) Without Filter, (b) With Filter.

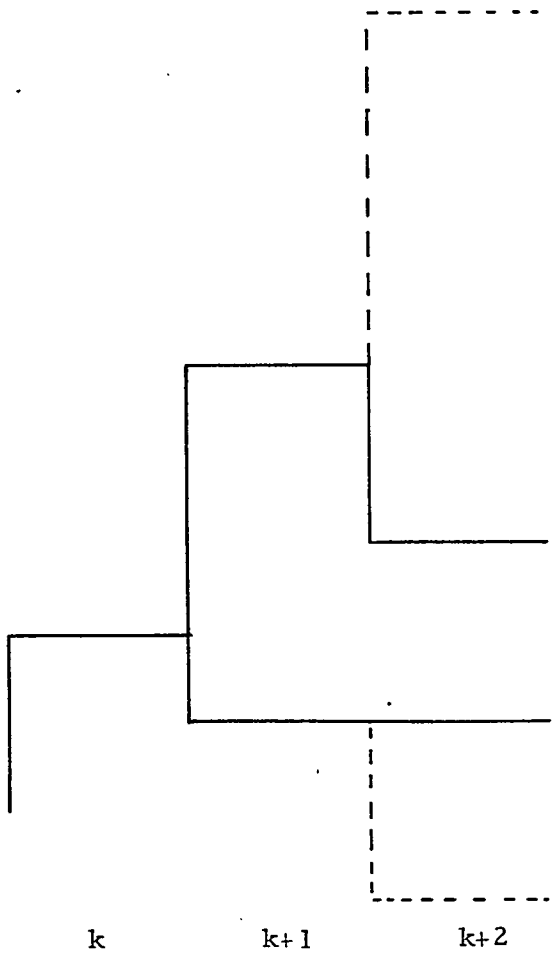


Fig. 2.3-17: 2-Bit Look ahead (all 4 possibilities).

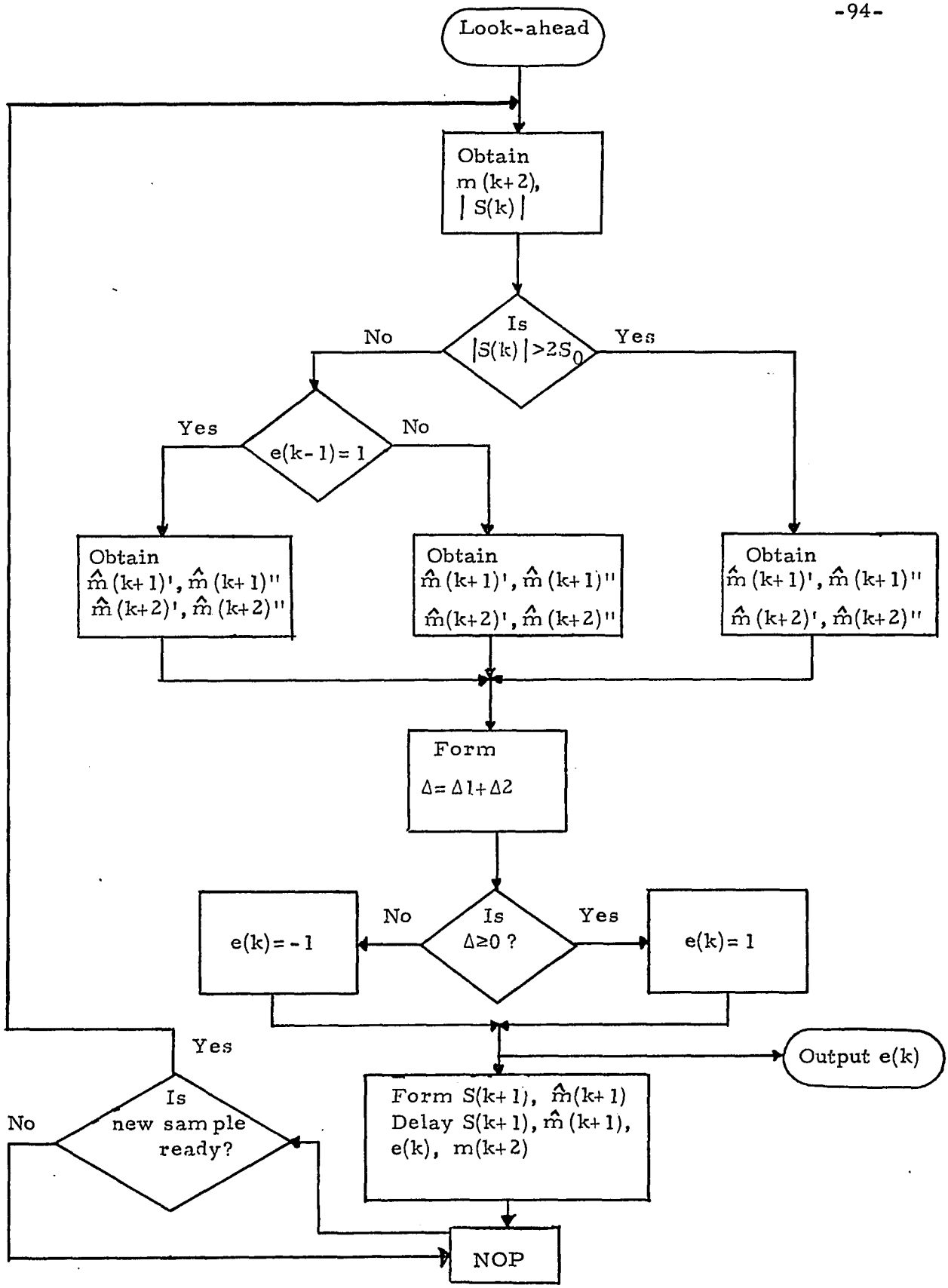


Fig. 2.3-18: Flow Chart of SVADM Transmitter Employing Look ahead.

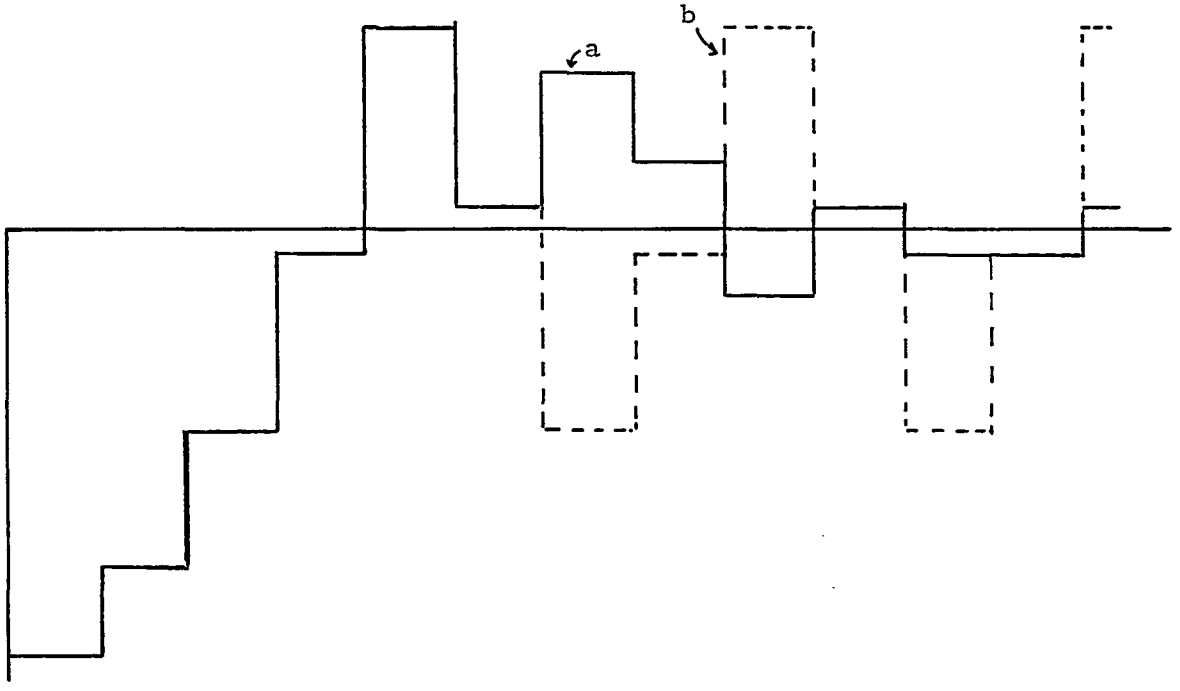


Fig. 2.3-19: Response of SVADM to a Step Input;(a)With Look-ahead, (b)Without Look-ahead.

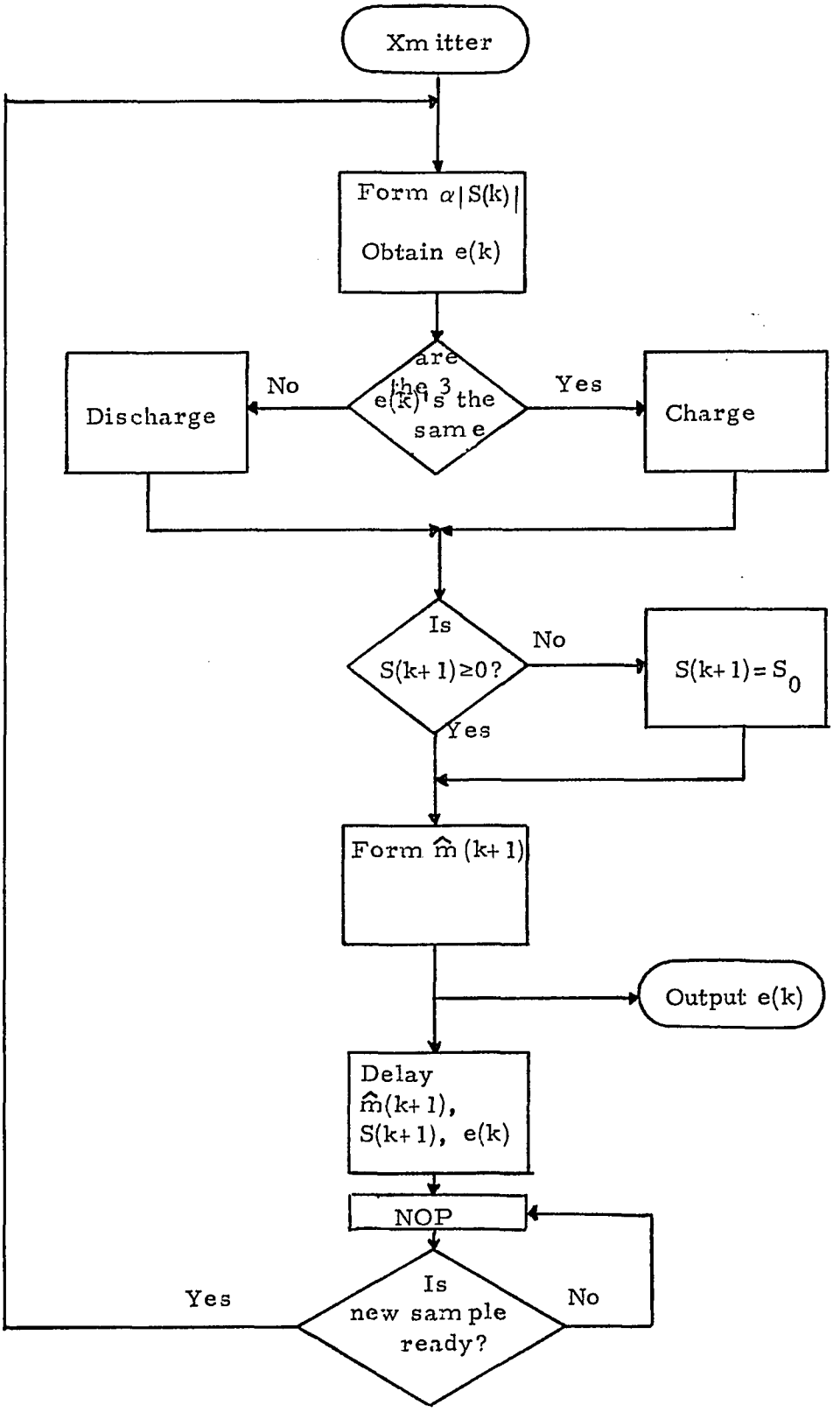


Fig. 2.4-1: Flow Chart of CVSD Transmitter.

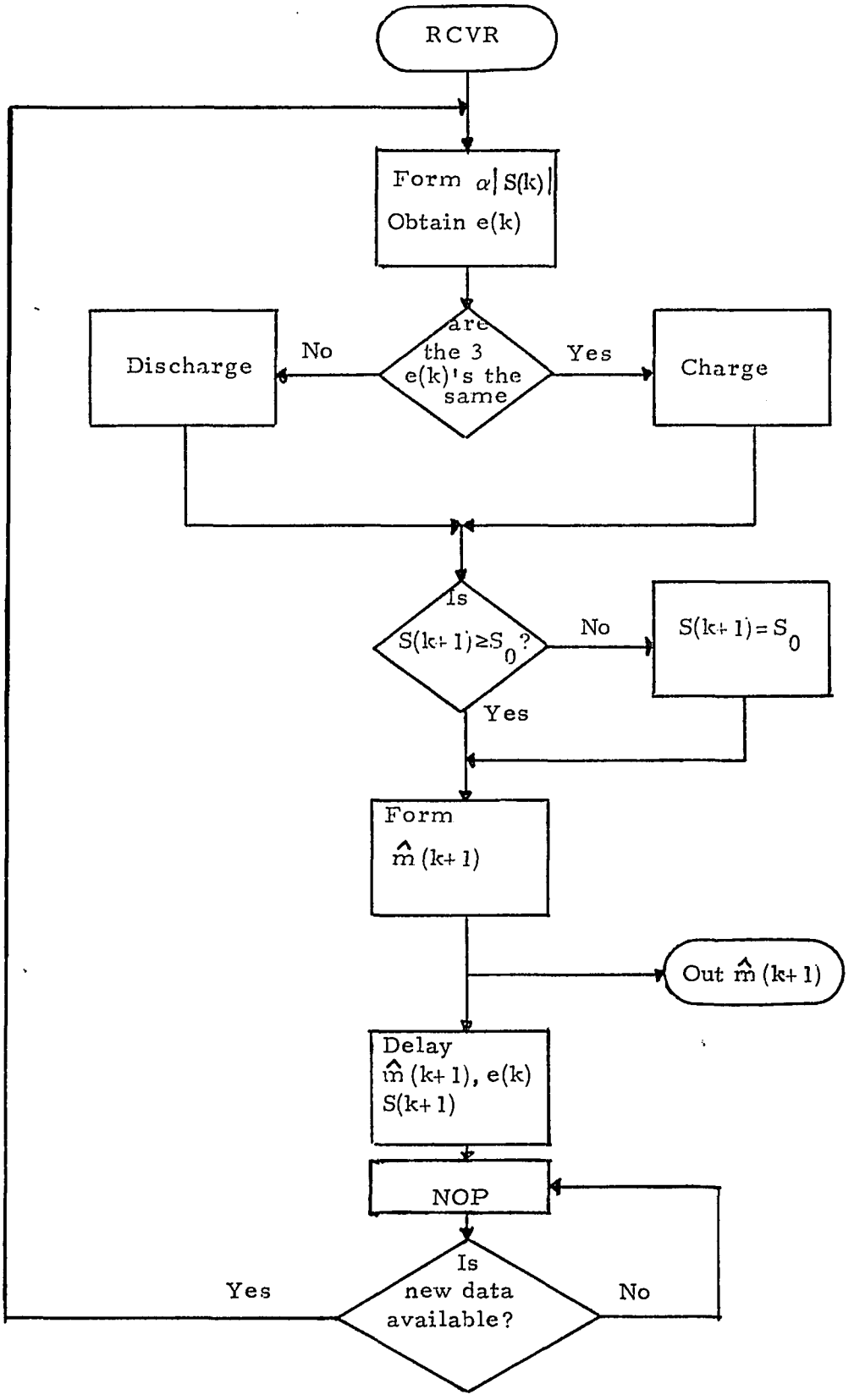
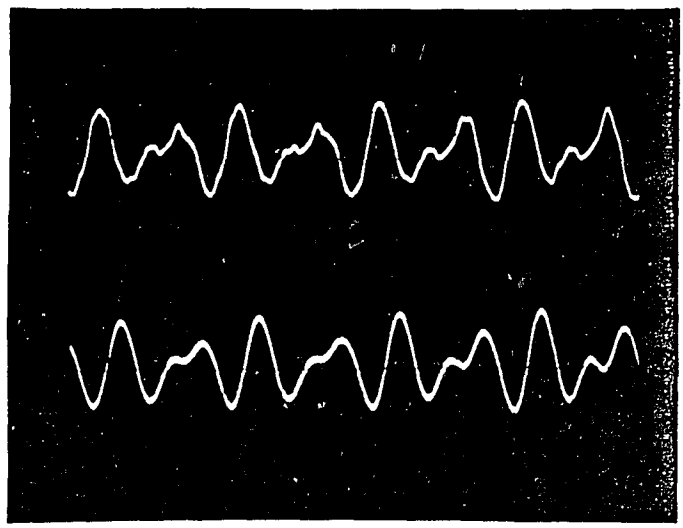


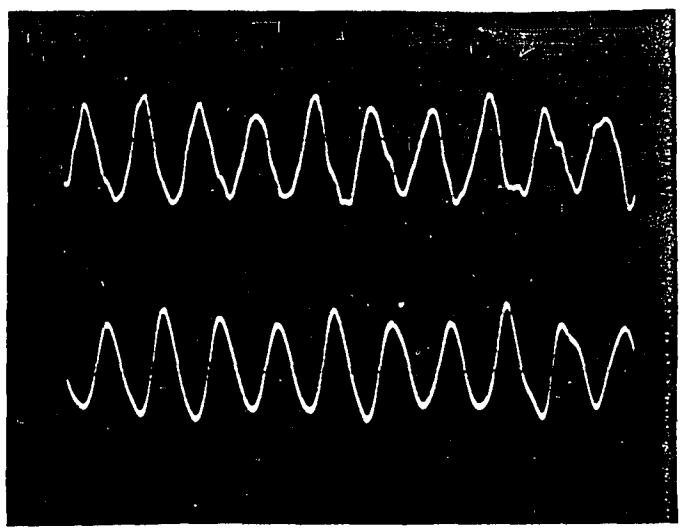
Fig. 2.4-2:Flow Chart of CVSD Receiver.



Estim ate

(a)

Input

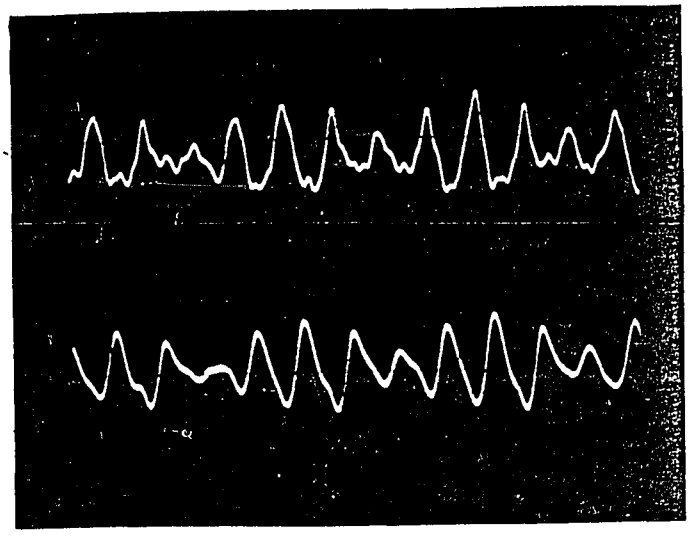


Estim ate

(b)

Input

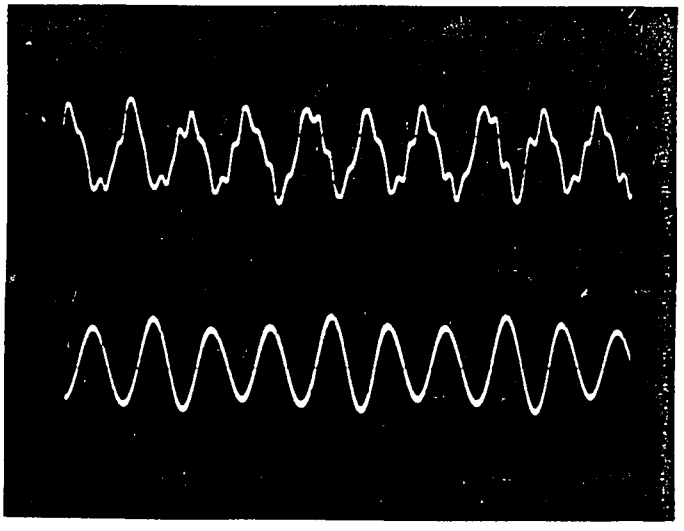
Fig. 2.4-3: Response of CVSD to Speech At;(a)32 KHz Sampling Rate, (b) 24 KHz Sam pling Rate.



Estimate

(c)

Input



Estimate

(d)

Input

Fig. 2.4-3: Continued, (c) 16 KHz Sampling Rate, (d) 9.6 KHz Sampling Rate.

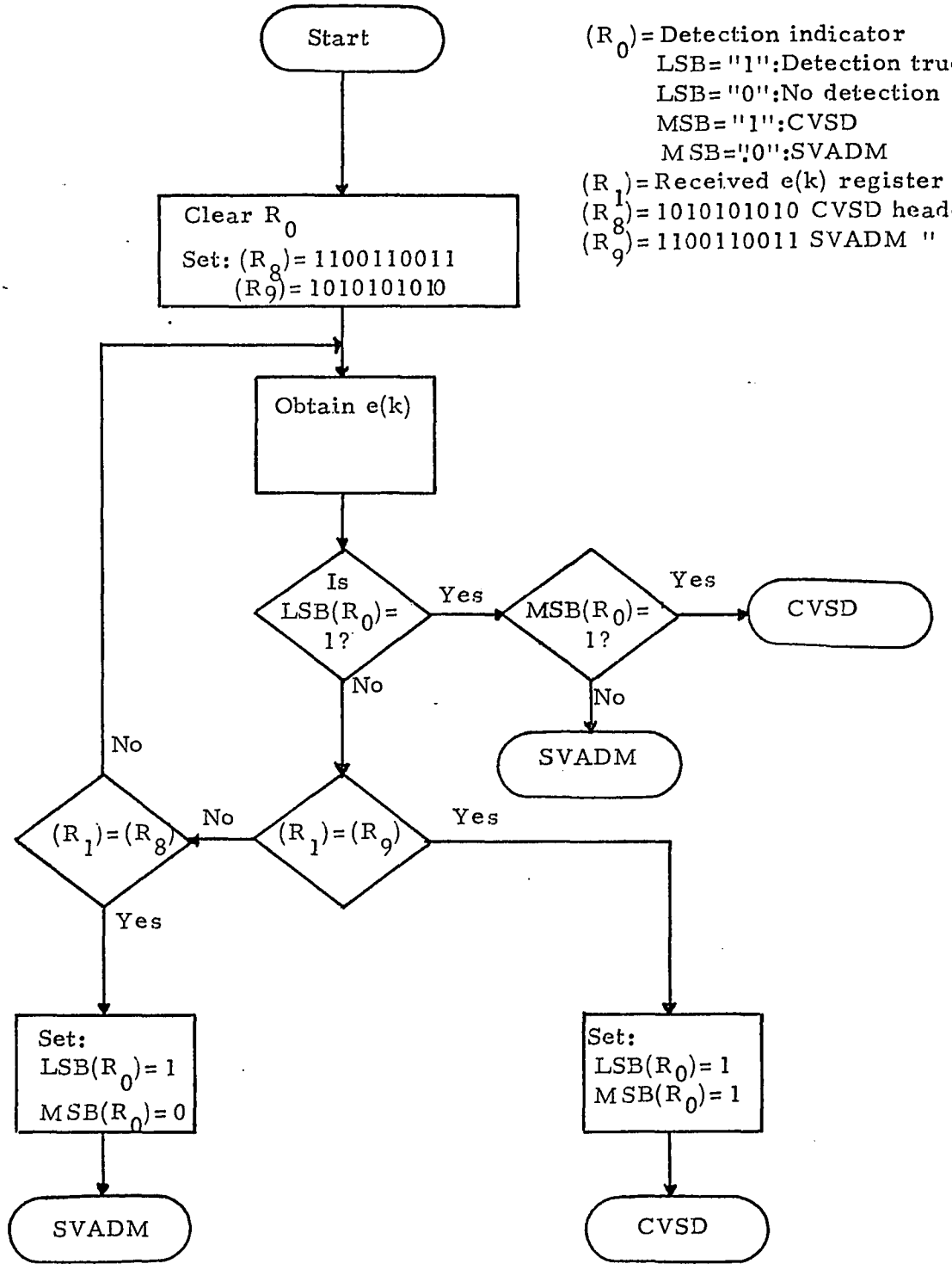


Fig. 2.5-1: Flow Chart of a Receiver Locking on a Header.

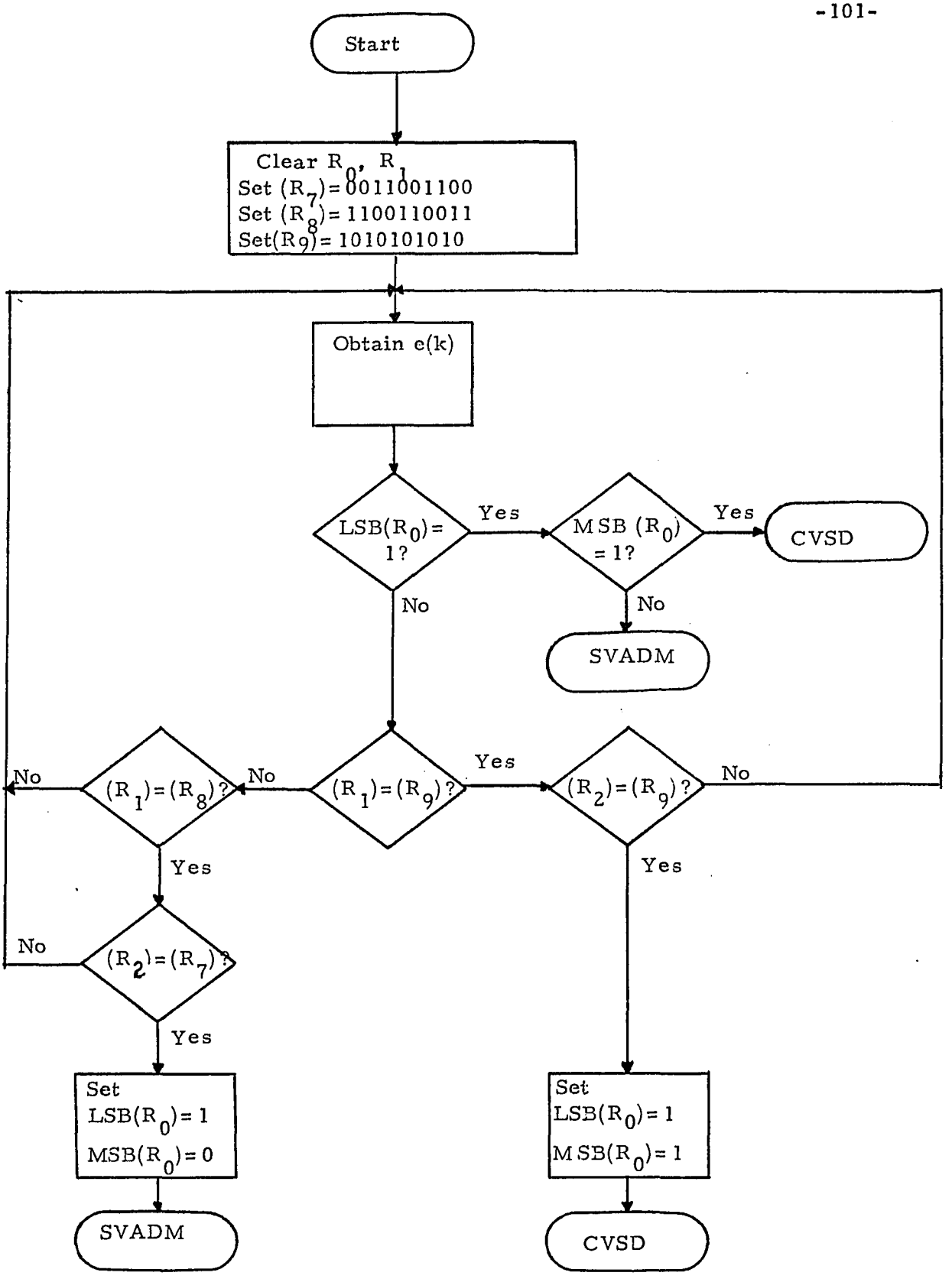


Fig. 2.5-2: Flow Chart of a Receiver locking on Steady State Pattern.

Chapter 3

3.1 Linear Delta Modulator

The operation of the LDM can be described mathematically as:

$$\hat{m}(k+1)=\hat{m}(k)+Se(k) \qquad (3.1\text{-}1a)$$

$$\Delta(k)=m(k)-\hat{m}(k) \qquad (3.1\text{-}1b)$$

$$e(k)=\text{sign}(\Delta(k)) \qquad (3.1\text{-}1c)$$

where k is an integral multiple of the sampling time τ_s, and m(k) is the value of m(t) at the sampling instant $t=k\tau_s$, and τ_s has been normalized to unity.

For the case of zero input, $\hat{m}(k)$ will alternate between +S/2 and -S/2 as shown in Fig. 3.1-1 . It is fairly safe to say that the probability that the estimated signal will be +S/2 at any instant k is equal to the probability that it be -S/2. Using probability notation it is equivalent to saying:

$$\Pr[\hat{m}(k)=S/2 \,|\, \text{zero input}]=\Pr[\hat{m}(k)=-S/2 \,|\, \text{zero input}]=1/2 \qquad (3.1\text{-}2)$$

Once an input is applied, the estimated signal $\hat{m}(k)$ will assume a staircase form where a level is held constant between sampling times and the difference between adjacent levels is the step size S. For a stochastic input a set of such staircase functions will be generated and will form a tree.

3.2 Probabilistic Model : Zero Order Markov case

3.2-1 Transitional probability structure

Let a stochastic input with a continuous amplitude probability density function be applied to the LDM at time zero, i.e. k=0. Since $Pr[\hat{m}(0)=S/2]=Pr[(\hat{m}(0)=-S/2]=1/2$ it does not matter whether $\hat{m}(0)=\frac{S}{2}$ or $\hat{m}(0)=-\frac{S}{2}$ is assumed. Let the initial condition at k=0 be $\hat{m}(0)=\frac{S}{2}$. At time k=1 $\hat{m}(1)$ can assume one of two values: $3\frac{S}{2}$ or $-\frac{S}{2}$. If at time k=0 $\Delta(0)\geq 0$, $\hat{m}(1)$ will be $3\frac{S}{2}$, if at time k=0 $\Delta(0)\leq 0$ $\hat{m}(1)$ will be $-\frac{S}{2}$. Expressing this probabilistically we have:

$$Pr[\hat{m}(1)=3\tfrac{S}{2}]=Pr[m(0)>\tfrac{S}{2},m(0)=\tfrac{S}{2}] \qquad (3.2\text{-}1a)$$

$$Pr[\hat{m}(1)=-\tfrac{S}{2}]=Pr[m(0)<\tfrac{S}{2},\hat{m}(0)=\tfrac{S}{2}] \qquad (3.2\text{-}1b)$$

Since the value of the estimated signal before the input is applied is indepencent of such input, Eq.(3.2-1) can be rewritten as:

$$Pr[\hat{m}(1)=3\tfrac{S}{2}]=Pr[\hat{m}(0)=\tfrac{S}{2}]Pr[m(0)>\tfrac{S}{2}] \qquad (3.2\text{-}2a)$$

$$Pr[\hat{m}(1)=-\tfrac{S}{2}]=Pr[\hat{m}(0)=\tfrac{S}{2}]Pr[m(0)\leq\tfrac{S}{2}] \qquad (3.2\text{-}2b)$$

Following the same line of reasoning as above and using Eq.(3.1-1), $\hat{m}(2)$ can assume one of three values $5\frac{S}{2}$, $\frac{S}{2}$ and $-3\frac{S}{2}$ and following Eqs.(3.1-2) and (3.2-2):

$$Pr[\hat{m}(2)=5\tfrac{S}{2}]=Pr[\hat{m}(1)=3\tfrac{S}{2},m(1)>3\tfrac{S}{2}] \qquad (3.2\text{-}3a)$$

$$Pr[\hat{m}(2)=\tfrac{S}{2}]=Pr[\hat{m}(1)=3\tfrac{S}{2},m(1)\leq 3\tfrac{S}{2}]+Pr[\hat{m}(1)=-\tfrac{S}{2},m(1)>-\tfrac{S}{2}] \qquad (3.2\text{-}3b)$$

$$Pr[\hat{m}(2)=-3\tfrac{S}{2}]=Pr[\hat{m}(1)=-\tfrac{S}{2},m(1)\leq-\tfrac{S}{2}] \qquad (3.2\text{-}3c)$$

Using Bayes' theorem Eq.(3.2-3) can be written as:

$$\Pr[\hat{m}(2)=5\tfrac{S}{2}]=\Pr[m(1)>3\tfrac{S}{2}\,|\,\hat{m}(1)=3\tfrac{S}{2}]\Pr[\hat{m}(1)=3\tfrac{S}{2}] \qquad (3.2\text{-}4a)$$

$$\Pr[\hat{m}(2)=\tfrac{S}{2}]=\Pr[m(1)\leq 3\tfrac{S}{2}\,|\,\hat{m}(1)=3\tfrac{S}{2}]\Pr[\hat{m}(1)=3\tfrac{S}{2}]+$$
$$\Pr[m(1)>-\tfrac{S}{2}\,|\,\hat{m}(1)=-\tfrac{S}{2}]\Pr[\hat{m}(1)=-\tfrac{S}{2}] \qquad (3.2\text{-}4b)$$

$$\Pr[\hat{m}(2)=-\tfrac{S}{2}]=\Pr[m(1)<-\tfrac{S}{2}\,|\,\hat{m}(1)=-\tfrac{S}{2}]\Pr[\hat{m}(1)=-\tfrac{S}{2}] \qquad (3.2\text{-}4c)$$

In Eq. (3.2-4a), for instance, $\Pr[m(1)>3\frac{S}{2}/\hat{m}(1)=3\frac{S}{2}]$ is known as the transitional probability. Note that:

$$\Pr[m(1)>3\tfrac{S}{2}\,|\,\hat{m}(1)=3\tfrac{S}{2}]=\Pr[\hat{m}(2)=5\tfrac{S}{2}\,|\,\hat{m}(1)=3\tfrac{S}{2}] \qquad (3.2\text{-}5)$$

This allows us to rewrite Eqs.(3.2-1) (3.2-2) and (3.2-4) in a recursive form as:

$$\Pr[\hat{m}(k)=\alpha\tfrac{S}{2}]=\sum_{\beta}\Pr[\hat{m}(k=\alpha\tfrac{S}{2}\,|\,\hat{m}(k-1)=\beta\tfrac{S}{2}]\Pr[\hat{m}(k-1)=\beta\tfrac{S}{2}]$$
$$=\sum_{\beta}P_{k,k-1}(\alpha,\beta)\Pr[\hat{m}(k-1)=\beta\tfrac{S}{2}] \qquad (3.2\text{-}6)$$

where $P_{k,k-1}(\alpha,\beta)$ is the transition probability from level $\beta\frac{S}{2}$ at time k-1 to level $\alpha\frac{S}{2}$ at time k.

3.2-2. Tree structure of the LDM

Let the value of the estimated signal at time k, $\hat{m}(k)$, be a random variable and let the value of that random variable be denoted as the state of the LDM at time k. We define a graph called the "Estimated Tree Graph", shown in Fig.3.2-1, in which the nodes are the states of the LDM at time k and the branches are the transitional probabilities.

At this point we note that since, as in all physical systems, the LDM response is limited to a finite number of levels and since the input signal in all existing systems is finite, the estimated tree will be bounded. However, generality will not be lost since the bound imposed is a function of the number of desired levels.

Let the input signal have an autocorrelation function which is an impulse. Suppose that the value of the random variable $\hat{m}(k-1)$ is $\beta\frac{S}{2}$, then:

$$\Pr[\hat{m}(k)=\alpha\tfrac{S}{2}\,|\,\hat{m}(k-1)=\beta\tfrac{S}{2}]=\begin{cases}\Pr[m(k-1)>\beta\frac{S}{2}] & \alpha=\beta+1 \qquad (3.2\text{-}6)\\ \Pr[m(k-1)\leq\beta\frac{S}{2}] & \alpha=\beta-1 \qquad (3.2\text{-}7)\end{cases}$$

Thus, using Eqs.(3.1-1), (3.2-5), (3.2-7), the assumption of a zero-order Markov input and a bounded response, it was found that:

$$\Pr[\hat{m}(k)=\alpha\tfrac{S}{2}/\hat{m}(k-1)=\beta\tfrac{S}{2}] = \begin{array}{ll} 0 & |\alpha-\beta|\neq 2\\ 0 & |\alpha|\geq 2N+3,\ |\beta|\geq 2N+1 \quad \alpha-\beta=2\\ 0 & |\alpha|\geq 2N+1,\ |\beta|\geq 2N+3 \quad \beta-\alpha=2\\ 1 & |\beta|=2N+1 \ \ |\alpha|=|\beta|-2 \qquad (3.2\text{-}8)\\ \Pr[m(k-1)\leq\beta\frac{S}{2}] & \beta-\alpha=2\\ \Pr]m(k-1)>\beta\frac{S}{2}] & \alpha-\beta=2 \end{array}$$

Where α is a value chosen from the set $\{2k+1-4i,\ i=0,1,2,\ldots,k\}$. $2k+1-4i$ assures us that α will have the correct value corresponding to the proper sampling time. N is a function of the maximum number of levels and if we denote the maximum number of levels by L, then:

$$L=2N+2 \tag{3.2-9}$$

A tree graph is now constructed using Eqs.(3.1-2), (3.2-2) and (3.2-8) in conjunction with the assumption that the state of the LDM at k=0 is $\frac{S}{2}$ and the knowledge of the input signal statistics in the following manner:

Since at time k=0 the state of the LDM is $\frac{S}{2}$, the possible states at time k are calculated from the following equation.

$$[\text{Possible states at time k}] = (2k+1-4i)S/2 \quad i=0,1,..,k \tag{3.2-10}$$

Therefore at time k=1 $\hat{m}(k=1)$ can have the value $3\frac{S}{2}$ or $-\frac{S}{2}$. Using Eq.(3.2-8) the transition probabilities $Pr[\hat{m}(1)=3\frac{S}{2}\ \hat{m}(0)=\frac{S}{2}]$ and $Pr[\hat{m}(1)=-\frac{S}{2}|\hat{m}(0)=\frac{S}{2}]$ are calculated. Knowing the transition probabilities, we can compute, using Eq.(3.2-6), the probability that the LDM will be in $3\frac{S}{2}$ or $-\frac{S}{2}$ levels at time k=1. Using Eq.(3.2-10) the possible states that the LDM can have at time k=2 are $5\frac{S}{2}$, $\frac{S}{2}$ and $-3\frac{S}{2}$. The probability of the LDM being in these states at time k=2 is again found by first computing the transition probabilities from LDM states at time k=1 to the LDM states at k=2 via Eq.(3.2-8) and then using Eq.(3.2-6).

The above procedure is repeated again and again for increasing values of time until the upper and lower bounds on the LDM response are achieved. From that point on the possible states of the LDM will repeat every two time samples. An example of such a trellis is shown in Fig.3.2-2, where the bound on the number os states is 8. The value of the levels at time k are assigned the value of the probability that the LDM is in these states at time k and the jump S from state a to state b is assigned the value of the possibility of the LDM being in state b at time k+1, given that it was in state a at time k. In other words, S is assigned the value of the transitional probability $P_{k+1,k}(b,a)$.

From Fig.3.2-2 it is evident that the states are repeated every two samples. The estimated signal will assume a value from the set {1,3,5,7} at times k=3,5,7,9,.. and will assume a value from the set {2,4,6,8} at times k=4,6,8,... . It is obvious that the LDM will reach a steady state pattern. A steady state pattern is defined as that pattern having transition probabilities and level probabilities which are independent of time, i.e.

$$P_{k+1,k}(b,a)=P_{k+3,k+2}(b,a)=P_{k+15,k+14}(b,a)=.. \qquad (3.2\text{-}11a)$$

$$P_r[\hat{m}(k)=\alpha\tfrac{S}{2}]=P_r[\hat{m}(k+2)=\alpha\tfrac{S}{2}]=P_r[\hat{m}(k+2)=\alpha\tfrac{S}{2}]=... \qquad (3.2\text{-}11b)$$

The time that it takes the LDM tree to reach a steady pattern is called the "transition time". This time will depend on the bounds on the LDM as well as on the input statistics.

3.3 The Autocorrelation and Power Spectral Density Functions

From the previous section we recall that the state of the LDM at time k, $\hat{m}(k)$, was defined as a random variable. However, since $\hat{m}(k)$ assumes different values at k-even than it does at k-odd, the process will not be stationary because the statistics of $\hat{m}(k)$ depend on the time k. Let us define a new random variable $\underline{X(T)}$, where $\underline{X(T)}$ assumes all possible values of the LDM state, and T is a function of two time instances, k and k+1. We label the most positive state as $x_1(T)$, the next most positive state as $x_2(T)$ and so on. In other words:

$$\underline{X(T)}=\{x_1(T),x_2(T),x_3(T),\ldots\}=\{\underline{X_o(T)},\underline{x_e(T)}\}$$

$\underline{X(T)}$ spans the plane consisting of odd and even values as shown in Fig. 3.3-1 . Then, if $x_i(T)\varepsilon\underline{X_o(T)}$ then $x_j(T+1)\varepsilon\underline{X_o(T+1)}=\underline{X_o(T)}$ where $x_i(T)$ is the state of the LDM at time T and $x_j(T+1)$ is the state of the LDM at time T+1. Similarly if $x_i(T)\varepsilon\underline{X_e(T)}$, then $x_j(T+1)\varepsilon\underline{X_e(T+1)}=\underline{X_e(T)}$.

We now specify a transition matrix $\underline{\underline{P(1)}}$ with elements $p_{ij}(1)$ defined as:

$$p_{ij}(1)=Pr\{X(T+1)=x_j(T+1)\,|\,X(T)=x_i(T)\}$$

Since the DM tree is in steady state, the transition matrix $\underline{\underline{P(1)}}$ is independent of T and thus the Markov chain can be considered homogeneous.

To find the autocorrelation of the random variable $\underline{X(T)}$ we proceed as follows; Since the process is stationary: $R_x(T,T+\tau)=R_x(\tau)=E\{\underline{X(T)}\,\underline{x(T+\tau)}\}$

Since $\underline{X(T)}$ can assume only discrete values:

$$R_x(\tau)=\sum_{ij}\sum x_i(T)x_j(T+\tau)\Pr\{\underline{X(T+\tau)}=x_j(T+\tau),\underline{X(T)}=x_i(T)\} \quad (3.3\text{-}4)$$

Using Bayes' theorem Eq. (3.3-3) can be written as:

$$R_x(\tau)=\sum_{ij}\sum x_i(T)\Pr\{\underline{X(T)}=x_i(T)\}p_{ij}(\tau)x_j(T+\tau) \quad (3.3\text{-}5)$$

where

$$p_{ij}(\tau)=\Pr\{\underline{X(T+\tau)}=x_j(T+\tau)\,\big|\,\underline{X(T)}=x_i(T)\} \quad (3.3\text{-}6)$$

$p_{ij}(\tau)$ is an element of the transition matrix $\underline{\underline{P(1)}}$ to the τ power. Equation (3.3-5) can then be written as:

$$R_x(\tau)=[x_1(T)\Pr[\underline{x(T)}=x_1(T)],\ldots,x_i(T)\Pr[\underline{x(t)}=x_i(T)],..] \quad (3.3\text{-}7)$$

$$\left[P_{ij}(1)\right]^{\tau}\begin{bmatrix}x_1(T+\tau)\\ \vdots\\ x_j(T+\tau)\\ \vdots\end{bmatrix}$$

Note $x_i(T)$ and $x_j(T+\tau)$ are the states of the LDM at times T and $T+\tau$ respectively. These states do not change with time. The term $\Pr[\underline{X(T)}=x_i(T)]$ is the initial condition. $\underline{\underline{P(1)}}$ depends only on the input signal statistics (Eq. (3.2-8)) and if the input signal is stationary, $\underline{\underline{P(\tau)}}$ will depend only on τ.

The power spectral density of the discrete signal can now be found as:

$$S_x(w)_d=\frac{1}{p_\tau}\sum R_x(\tau)e^{-jwp\tau} \quad (3.3\text{-}8)$$

where p is the period and d denotes discrete. However, if the continuous time function of the envelope is available one can find the power spectral density by using the Fourier Transform:

$$s_x(w) = \int_{-\infty}^{\infty} R_x(\tau) e^{-jw\tau} d\tau \qquad (3.8\text{-}9)$$

3.4 Autocorrelation and Power Spectral density function Calculations of a zero order Markov input

3.4-1 Example: perform the autocorrelation and power spectral density calculation of a zero order Markov input with a uniform pdf.

Consider an input m(t) bounded between +3V and -3V with a uniform pdf. of independent samples as shown in Fig. 3.4-1. Let the step size S be normalized to unity. The maximum number of levels is found to be 8 (Eq.(3.2-9)), as shown in Fig.3.2-2. The value of the LDM tree states at steady state is:

$$\underline{X}=\{3.5,2.5,1.5,0.5,-0.5,-1.5,-2.5,-3.5\} \qquad (3.4\text{-}1)$$

Using Eqs.(3.2-6), (3.2-8) and following the procedure described in the previous section the initial probabilities are found to be:

$$\begin{aligned} &Pr(X=3.5)=0.0037 \quad Pr(X=-3.5)=0.0037 \\ &Pr(X=2.5)=0.0441 \quad Pr(X=-2.5)=0.0441 \\ &Pr(X=1.5)=0.1615 \quad Pr(X=-1.5)=0.1615 \end{aligned} \qquad (3.4\text{-}2)$$

$Pr(X=0.5)=0.2907 \qquad Pr(X=-0.5)=0.2907$

We redraw Fig. 3.2-2 as Fig. 3.4-2 , with the modification of the added transition probabilities which were calculated using Eq.(3.2-8). From Fig. 3.4-2 it is evident that the steady state pattern starts at k=3.

Using Fig. 3.4-2 , the transition probability matrix, $\underline{\underline{P(1)}}$, is found to be:

$$\underline{\underline{P(1)}}=\begin{bmatrix} 1/12 & 0 & 11/12 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5/16 & 0 & 11/16 & 0 & 0 & 0 & 0 \\ 1/48 & 0 & 26/48 & 0 & 21/48 & 0 & 0 & 0 \\ 0 & 5/48 & 0 & 47/72 & 0 & 35/144 & 0 & 0 \\ 0 & 0 & 35/144 & 0 & 47/72 & 0 & 5/48 & 0 \\ 0 & 0 & 0 & 21/48 & 0 & 26/48 & 0 & 1/48 \\ 0 & 0 & 0 & 0 & 11/16 & 0 & 5/16 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11/12 & 0 & 1/12 \end{bmatrix} \qquad (3.4\text{-}3)$$

It is clear that the chain is homogeneous. To see if stationarity holds we proceed as follows: Multiplying the transition matrix to the 20-power and the 21-power the following matrix was obtained:

$$\underline{\underline{P(20)}}=\underline{\underline{P(21)}}=\begin{bmatrix} .0073 & 0 & .3231 & 0 & .5815 & 0 & .0881 & 0 \\ 0 & .0881 & 0 & .5815 & 0 & .3231 & 0 & .0073 \\ .0073 & 0 & .3231 & 0 & .5815 & 0 & .0881 & 0 \\ 0 & .0881 & 0 & .5815 & 0 & .3231 & 0 & .0073 \\ .0073 & 0 & .3231 & 0 & .5815 & 0 & .0881 & 0 \\ 0 & .0881 & 0 & .5815 & 0 & .3231 & 0 & .0073 \\ .0073 & 0 & .3231 & 0 & .5815 & 0 & .0081 & 0 \\ 0 & .0881 & 0 & .5815 & 0 & .3231 & 0 & .0073 \end{bmatrix} \qquad (3.4\text{-}4)$$

Since the sampling frequency is normalized to unity, for a 10^{-4} sec. sampling period as an example, w=1 in Fig. 3.4-4 corresponds to a frequency of 5000 Hz. As the sampling period decreases, w=1 corresponds to a larger frequency and if the bandwidth B is between 300Hz and 3000 Hz, then less noise power will be inband compared to signal power and the signal to noise ratio will increase.

3.4-2 Example: Perform the autocorrelation and power density calculations of a zero order Markov input with gaussian pdf.

Consider an input m(t) bounded between +3V and -3V, with a gaussian pdf. of independent samples and a variance of 1 as shown in Fig. 3.4-5 . Let the step size S be normalized to unity. The maximum number of levels is found to be 8 (Eq.(3.2-9)), as shown in Fig. 3.2-2. The value of the LDM tree states at steady state is same as in Eq.(3.4-1).

Using Eqs.(3.2-6),(3.2-8) and following the procedure described in section 3.3, the initial probabilities are found to be

$$\begin{aligned} P_r[X=3.5] &= 0.93X10^{-9} & P_r[X=-3.5] &= 0.93X10^{-9} \\ P_r[X=2.5] &= 0.93X10^{-4} & P_r[X=-2.5] &= 0.93X10^{-7} \\ P_r[X=1.5] &= 0.0685 & P_r[X=-1.5] &= 0.0685 \\ P_r[X=0.5] &= 0.4314 & P_r[X=-0.5] &= 0.4314 \end{aligned} \qquad (3.4\text{-}13)$$

In the previous section it was mentioned that the set of states of the LDM tree contains two disjoint sets:

$$\underline{X(T)} = [\{x_1 x_3 x_5 x_7\}, \{x_2 x_4 x_6 x_8\}] = \{\underline{X_o}(T), \underline{X_e}(T)\} \qquad (3.4\text{-}5)$$

The zeros in the odd columns correspond to the probability of a transition from an even state to an odd state. Since the two sets of states are disjoint, the transition probability is zero as expected. Same applies for zeros in the even columns.

Now, $\underline{\underline{P}}(20)$ can be transformed into a matrix $\underline{\underline{P_T}}(20)$ (see appendix H) where

$$\underline{\underline{P_T}}(20) = \left[\begin{array}{cccc|cccc} .0073 & .3231 & .5815 & .0881 & & & & \\ .0073 & .3231 & .5815 & .0081 & & & & \\ .0073 & .3231 & .5815 & .0081 & & \underline{\underline{O}} & & \\ .0073 & .3231 & .5815 & .0081 & & & & \\ \hline & & & & .0881 & .5815 & .3231 & .0073 \\ & & & & .0881 & .5815 & .3231 & .0073 \\ & \underline{\underline{O}} & & & .0881 & .5815 & .3231 & .0073 \\ & & & & .0881 & .5815 & .3231 & .0073 \end{array}\right] \qquad (3.4\text{-}6)$$

or in short hand notation:

$$\underline{\underline{P_T}}(20) = \left[\begin{array}{c|c} \underline{\underline{A_o}} & \underline{\underline{O}} \\ \hline \underline{\underline{O}} & \underline{\underline{A_e}} \end{array}\right] \qquad (3.4\text{-}7)$$

We observe that all the rows of submatrix $\underline{\underline{A_o}}$ are identical which means that the probability of the LDM being in odd state x_i is independent on starting state, providing the starting state is from the odd set of states.

However, this is precisely what the limit of Eq.(G-20) means. Therefore we conclude that since the limit exists and since the chain is homogeneous, the process is stationary in the asymptotic (wide) sense.

Using P(1), initial condition probabilities, Eq.(3.4-2), and Eq.(3.3-7) the first 11 terms of the autocorrelation function $R_X(\tau)$ are:

$R_X(0)=1.4740$ $R_X(4)=0.0596$ $R_X(8)=0.0024$

$R_X(1)=0.6607$ $R_X(5)=0.0267$ $R_X(9)=0.0011$

$R_X(2)=0.2963$ $R_X(6)=0.0123$ $R_X(10)=0.0005$

$R_X(3)=0.1329$ $R_X(7)=0.0054$ (3.4-8)

The autocorrelation function of the LDM estimate is plotted in Fig. 3.4-3 . Note that it decreases exponentially and we conclude, for the case of 8 states, that:

$$R_X(\tau)=Ae^{-\alpha|\tau|} \tag{3.4-9}$$

where $A=1.474$ and $\alpha=\ln(\frac{1.474}{0.6607})=0.802$. The power spectral density is readily shown to be (Eq. (3.3-9)):

$$S_X(w)=\frac{2\alpha A}{\alpha^2+w^2} \tag{3.4-10}$$

In Fig. 3.4-4 , the power spectral density is plotted as a function of the radian freauency w.
If the sampling period is T_s, then:

$$\tau=2T_s \tag{3.4-11}$$

and

$$w=\frac{2\Pi}{\tau} \tag{3.4-12}$$

Using the procedure outlined in Section 3.3 and Eq.(3.2-8), the transition probability matrix, $\underline{\underline{P(1)}}$, is found to be:

$$\underline{\underline{P(1)}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0.00001 & 0 & 0.99999 & 0 & 0 & 0 & 0 \\ 0 & 0.0014 & 0 & .9986 & 0 & 0 & 0 \\ 0 & 0 & .1587 & 0 & .8413 & 0 & 0 \\ 0 & 0 & 0 & .8413 & 0 & .1587 & 0 \\ 0 & 0 & 0 & 0 & .9986 & 0.0014 & 0 \\ 0 & 0 & 0 & 0 & .99999 & 0.0001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad (3.4\text{-}14)$$

To see if stationarity holds we proceed as follows: Multiplying the transition matrix to the 40-power and the 41-power the following matrix was obtained.

$$\underline{\underline{P(40)}} = \underline{\underline{P(41)}} = \begin{bmatrix} 0 & 0 & .1371 & 0 & .8627 & 0 & .0002 & 0 \\ 0 & .0002 & 0 & .8627 & 0 & .1371 & 0 & 0 \\ 0 & 0 & .1371 & 0 & .8627 & 0 & .0002 & 0 \\ 0 & .0002 & 0 & .8627 & 0 & .1371 & 0 & 0 \\ 0 & 0 & .1371 & 0 & .8627 & 0 & .0002 & 0 \\ 0 & .0002 & 0 & .8627 & 0 & .1371 & 0 & 0 \\ 0 & 0 & .1371 & 0 & .8627 & 0 & .0002 & 0 \\ 0 & .0002 & 0 & .8627 & 0 & .1371 & 0 & 0 \end{bmatrix} \qquad (3.4\text{-}15)$$

Again the zero's in the odd columns correspond to the probability of a transition from an even state to an odd state. Since the two sets of states are disjoint, the transition probability is zero as expected. The same result applies for zeros in the even columns.

Now, $\underline{\underline{P(40)}}$ can be transformed into a matrix $\underline{\underline{P_T(40)}}$ (see appendix II) where:

$$\underline{\underline{P_T}}(40) = \left[\begin{array}{cccc:cccc} 0 & .1371 & .8627 & .0002 & & & & \\ 0 & .1371 & .8627 & .0002 & & & & \\ 0 & .1371 & .8627 & .0002 & & & & \\ 0 & .1371 & .8627 & .0002 & & & & \\ \hdashline & & & & .0002 & .8627 & .1371 & 0 \\ & & & & .0002 & .8627 & .1371 & 0 \\ & & & & .0002 & .8627 & .1371 & 0 \\ & & & & .0002 & .8627 & .1371 & 0 \end{array}\right] \qquad (3.4\text{-}16)$$

or in short hand notation

$$\underline{\underline{P_T}}(40) = \left[\begin{array}{c:c} \underline{\underline{A_o}} & \underline{\underline{O}} \\ \hdashline \underline{\underline{O}} & \underline{\underline{A_e}} \end{array}\right] \qquad (3.4\text{-}17)$$

We observe that all the rows of submatrix $\underline{\underline{A_o}}$ are identical which means that the probability of the LDM being in odd state x_i is independent of the starting state, providing the starting state is from the odd set of states. However, this is precisely what the limit of Eq.(G-20) means. Therefore we conclude that since the limit exists and since the chain is homogeneous, the process is stationary in the asymptotic (wide) sense.

Using $\underline{\underline{P(1)}}$, initial condition probabilities, Eq.(3.4-13) and Eq.(3.3-7) the first 11≠terms of the autocorrelation function $R_X(\tau)$ are:

$$
\begin{aligned}
&R_x(o) = 0.5253 \quad R_x(4) = -.7X10^{-9} \quad R_x(8) = -.7X10^{-9} \\
&R_x(1) = .5X10^{-8} \quad R_x(5) = -.7X10^{-9} \quad R_x(9) = -.7X10^{-9} \\
&R_x(2) = -.7X10^{-9} \quad R_x(6) = -.7X10^{-9} \quad R_x(10) = -.7X10^{-9} \\
&R_x(3) = -.7X10^{-9} \quad R_x(7) = -.7X10^{-9}
\end{aligned} \tag{3.4-18}
$$

The autocorrelation function of the LDM estimate to zero order Markov gaussian input is plotted in Fig. 3.4-6 with five different values of the variance, σ 0.1, 0.5, 1, 1.5 and 2.

It is interesting to note that for all five cases the autocorrelation of the LDM estimate is still an impulse. Hence, for uncorrelated input samples the LDM estimate values are also uncorrelated, and response of LDM will not improve by observing the past history if the input is a zero order Markov.

3.5 Probabilistic Model:First order Markov case

3.5-1 Tree and transitional probability structure of the LDM

As with the zero order Markov case we let:

$$\Pr[\hat{m}(o)=s/2]=\Pr[\hat{m}(o)=-s/2]=1/2 \tag{3.5-1}$$

Since the value of the estimated signal before the input is applied is independent of such input, the probability of the two possible levels at time k=1 are found to be

$$Pr[\hat{m}(1)=3s/2]=Pr[\hat{m}(o)=s/2, m(o)>s/2]=Pr[\hat{m}(o)=s/2]Pr[m(o)>s/2] \quad (3.5\text{-}2a)$$

$$Pr[\hat{m}(1)=-s/2]=Pr[\hat{m}(o)=s/2, m(o)\leq s/2]=Pr[\hat{m}(o)=s/2]Pr[m(o)\leq s/2] \quad (3.5\text{-}2b)$$

Similarly, at time k=2, $\hat{m}(k)$ can assume one of three values (i.e. 5s/2, s/2 and -3s/2) with the following probabilities:

$$Pr[\hat{m}(2)=5s/2]=Pr[\hat{m}(1)=3s/2, m(1)>3s/2] \quad (3.5\text{-}3a)$$

$$Pr[\hat{m}(2)=s/2]=Pr[\hat{m}(1)=3s/2, m(1)\leq 3s/2]+Pr[\hat{m}(1)=-s/2, m(1)>-s/2] \quad (3.5\text{-}3b)$$

$$Pr[\hat{m}(2)=-3s/2]=Pr[\hat{m}(1)=-s/2, m(1)\leq -s/2] \quad (3.5\text{-}3c)$$

The "Estimated Tree Graph" is now redrawn in Fig. 3.5-1 , where the levels at time k are consecutively labled from 1 to k+1. The value of the ith level at time k is labled $x_i(k)$, and from Eq.(3.2-10)

$$x_i(k)=(2k+5-4i)s/2 \qquad i=1,2,\ldots k+1 \quad (3.5\text{-}4)$$

Based on Eqs.(3.5-1),(3.5-2),(3.5-3) and Fig. 3.5-1 , the general expression for the probability of a level is:

$$Pr[\hat{m}(k)=x_i(k)]=Pr[\hat{m}(k-1)=x_{i-1}(k-1), m(k-1)\leq x_{i-1}(k-1)] + Pr[\hat{m}(k-1)=x_i(k-1), m(k-1)>x_i(k-1)] \quad (3.5\text{-}5)$$

The probability of a level at time k, found in the above equation (i.e. Eq.(3.5-5)),can be assumed as being composed of two factors. One factor due to the level i-1 at time k-1. Let it be labled A(k,i). A second factor due to the level i at time k-1, and let it be labled B(k,i). In equation form

$$Pr[\hat{m}(k)=x_i(k)]=A(k,i)+B(k,i) \qquad (3.5\text{-}6)$$

For example:

$$A(2,2)=Pr[\hat{m}(1)=3s/2, m(1)\leq 3s/2) \qquad (3.5\text{-}7a)$$

$$B(2,1)=Pr[\hat{m}(1)=3s/2, m(1)>3s/2] \qquad (3.5\text{-}7b)$$

Examining the first part of Eq.(3.5-5) the transition probability from level $x_{i-1}(k-1)$ to level $x_i(k)$ can be found to be (see Appendix I):

$$Pr[\hat{m}(k)=x_i(k)\,|\,\hat{m}(k-1)=x_{i-1}(k-1)]=\frac{A(k,i)}{A(k-1,i-1)+B(k-1,i-1)} \qquad (3.5\text{-}8a)$$

and similarly:

$$Pr[\hat{m}(k)=x_i(k)\,|\,\hat{m}(k-1)=x_i(k-1)]=\frac{B(k,i)}{A(k-1,i)+B(k-1,i)} \qquad (3.5\text{-}8b)$$

A symmetrical bound is now imposed on the response of the LDM

$$-XN\leq x_i(k)\leq XN \qquad (3.5\text{-}9)$$

Using Eqs.(3.5-1),(3.5-3),(3.5-5),(3.5-6),(3.5-8) (3.5-9) and the set of initial conditions:

$$
\begin{aligned}
A(1,1) &= 0. \\
B(1,1) &= P_r[m(o)>s/2]P_r[\hat{m}(o)=s/2] \\
A(1,2) &= P_r[m(o)\leq s/2]P_r[\hat{m}(o)=s/2] \\
B(1,2) &= 0.
\end{aligned}
\qquad (3.5\text{-}10)
$$

An algorithm from which A(k,i) and B(k,i) can be found, was obtained. The algorithm, as is readily seen from Fig. 3.5-1 , is devided into four distinct regions; A, B, C and D.

The region lasting from initial time, k=1, to the time in which the upper bound is reached is defined as region -A-. In Fig. 3.5-1 the bound is XN=5S, hence region -A- lasts for $2\leq k\leq 5$. Region -B- is the region leading to the lower bound and for a symmetrical bound as in Fig. 3.5-1 , it lasts for one sampling period. Region -C- is the repetitive part of the tree containing the set of states with the upper bound state and region D is the repetitive part of the trellis containing the set of states with the lower bound state. The algorithm for each region can be verified to be:

Region A : $2 \leq K \leq XN$

$$A(k,i) = \begin{cases} 0 & i=1 \\ \Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) > x_{i-1}(k-2)] B(k-1,i-1) & i=2 \\ \Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) \leq x_{i-2}(k-2)] A(k-1,i-1) & i=k+1 \\ \left\{ \begin{array}{l} \Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) \leq x_{i-2}(k-2)] A(k-1,i-1) + \\ \Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) > x_{i-1}(k-2)] B(k-1,i-1) \end{array} \right\} & 3 \leq i \leq k \end{cases} \quad (3.5\text{-}11)$$

$$B(k,i) = \begin{cases} 0 & i=k+1 \\ \Pr[m(k-1) > x_i(k-1) \mid m(k-2) > x_i(k-2)] B(k-1,i) & i=1 \\ \Pr[m(k-1) > x_i(k-1) \mid m(k-2) \leq x_{i-1}(k-2)] A(k-1,i) & i=k \\ \left\{ \begin{array}{l} \Pr[m(k-1) > x_i(k-1) \mid m(k-2) \leq x_{i-1}(k-2)] A(k-1,i) + \\ \Pr[m(k-1) > x_i(k-1) \mid m(k-2) > x_i(k-2)] B(k-1,i) \end{array} \right\} & 2 \leq i \leq k-1 \end{cases} \quad (3.5\text{-}12)$$

Region B : $K = XN+1$

$$A(k,i) = \begin{cases} 0 & i=1 \\ B(k-1,i-1) & i=2 \\ \left\{ \begin{array}{l} \Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) \leq x_{i-2}(k-2)] A(k-1,i-1) + \\ \Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) > x_{i-1}(k-2)] B(k-1,i-1) \end{array} \right\} & 3 \leq i \leq k \\ \Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) \leq x_{i-2}(k-2)] A(k-1,i-1) & i=k+1 \end{cases} \quad (3.5\text{-}13)$$

$$B(k,i) = \begin{cases} 0 & i=1 \\ \left\{ \begin{array}{l} \Pr[m(k-1) > x_i(k-1) \mid m(k-2) \leq x_{i-1}(k-2)] A(k-1,i) + \\ \Pr[m(k-1) > x_i(k-1) \mid m(k-2) > x_i(k-2)] B(k-1,i) \end{array} \right\} & 2 \leq i \leq k-1 \\ \Pr[m(k-1) > x_i(k-1) \mid m(k-2) \leq x_{i-1}(k-2)] A(k-1,i) & i=k \\ 0 & i=k+1 \end{cases} \quad (3.5\text{-}14)$$

Region C : $k = [INT\{(k-XN)/2\}]\cdot 2 + XN$

$$A(k,i) =$$

$$\begin{cases} 0 & \\ \Pr[m(k-1) \le x_{i-1}(k-1)]A(k-1,i-1) + & \\ \Pr[m(k-1) \le x_{i-1}(k-1) \mid m(k-2) > x_{i-1}(k-2)]B(k-1,i-1) & \\ \begin{Bmatrix} \Pr[m(k-1) \le x_{i-1}(k-1) \mid m(k-2) \le x_{i-2}(k-2)]A(k-1,i-1) + \\ \Pr[m(k-1) \le x_{i-1}(k-1) \mid m(k-2) > x_{i-1}(k-2)]B(k-1,i-1) \end{Bmatrix} & \end{cases} \quad \begin{matrix} i=(k-XN+2)/2 \\ i=(k-XN+4)/2 \\ (3.5\text{-}15) \\ \frac{k-XN+6}{2} \le i \le \frac{k+XN+2}{2} \end{matrix}$$

$$B(k,i) =$$

$$\begin{cases} \Pr[m(k-1) > x_i(k-1)]A(k-1,i) + & \\ \Pr[m(k-1) > x_i(k-1) \mid m(k-2) > x_i(k-2)]B(k-1,i) & \\ \begin{Bmatrix} \Pr[m(k-1) > x_i(k-1) \mid m(k-2) \le x_{i-1}(k-2)]A(k-1,i) + \\ \Pr[m(k-1) > x_i(k-1) \mid m(k-2) > x_i(k-2)]B(k-1,i) \end{Bmatrix} & \\ A(k-1,i) & \\ 0 & \end{cases} \quad \begin{matrix} i=(k-XN+2)/2 \\ (3.5\text{-}16) \\ \frac{k-XN+4}{2} \le i \le \frac{k+XN}{2} \\ i=(k+XN+2)/2 \\ \text{otherwise} \end{matrix}$$

Region D: $k \neq [\text{INT}(\frac{k-XN}{2})] \cdot 2 + XN$

$A(k,i) =$

$$\begin{cases} B(k-1,i-1) & i = (k-XN+3)/2 \\ \left\{\begin{array}{l} \Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) \leq x_{i-2}(k-2)]A(k-1,i-1) \\ +\Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) > x_{i-1}(k-2)]B(k-1,i-1) \end{array}\right\} & \frac{k-XN+5}{2} \leq i \leq \frac{k+XN+1}{6} \\ \left\{\begin{array}{l} \Pr[m(k-1) \leq x_{i-1}(k-1) \mid m(k-2) \leq x_{i-2}(k-2)]A(k-1,i-1) \\ +\Pr[m(k-1) \leq x_{i-1}(k-1)]B(k-1,i-1) \end{array}\right\} & i = \frac{k+XN+3}{2} \\ 0 & \text{otherwise} \end{cases} \quad (3.5\text{-}17)$$

$B(k,i) =$

$$\begin{cases} \left\{\begin{array}{l} \Pr[m(k-1) > x_i(k-1) \mid m(k-2) \leq x_{i-1}(k-2)]A(k-1,i) + \\ \Pr[m(k-1) > x_i(k-1) \mid m(k-2) > x_i(k-2)]B(k-1,i) \end{array}\right\} & \frac{k-XN+3}{2} \leq i \leq \frac{k+XN-1}{2} \\ \left\{\begin{array}{l} \Pr[m(k-1) > x_i(k-1) \mid m(k-2) \leq x_{i-1}(k-2)]A(k-1,i) + \\ \Pr[m(k-1) > x_i(k-1)]B(k-1,i) \end{array}\right\} & i = \frac{k+XN+1}{2} \\ 0 & i = \frac{k+XN+3}{2} \\ 0 & \text{otherwise} \end{cases} \quad (3.5\text{-}18)$$

Knowing the transition probabilities and the probabilities of the various levels it is possible now to obtain the autocorrelation function the same way it was obtained in Section 3.3, namely:

$$R_x(\tau) =$$

$$x_1(T)\Pr[\underline{x(T)}=x_1(T)],\ldots,x_i(T)\Pr[\underline{x(T)}=x_i(T)],\ldots]$$

$$\cdot\left[P_{ij}[i]\right]^{\tau}\cdot\begin{bmatrix} x_1(T+\tau) \\ x_2(T+\tau) \\ \vdots \\ x_j(T+\tau) \end{bmatrix} \qquad (3.5\text{-}19)$$

where $x_i(T)$ and $x_j(T+\tau)$ are the states of the LDM at times T and $T+\tau$ respectively. The term $\Pr[\underline{x(T)}=x_i(T)]$ is the initial condition. $\underline{P}(1)$ depends only on the input signal statistics (Eqs. (3.5-11)-(3.5-18)) and with a stationary input signal, $\underline{P}(\tau)$ will depend only on τ.

The power spectral density of the discrete signal can now be found as:

$$S_x(\omega)_d = \frac{1}{p}\sum_{\tau} R_x(\tau)\ e^{j\omega p\tau} \qquad (3.5\text{-}20)$$

where p is the period and d denotes discrete. However if the continuous time function of the envelope is available one can find the power spectral density by using the Fourier transform:

$$S_x(\omega) = \int_{-\infty}^{\infty} R_x(\tau)\, e^{-j\omega\tau}\, d\tau \qquad (3.5\text{-}21)$$

3.5-2 Example: Perform the autocorrelation calculation of a first order Markov input with a Gaussian P df.

Consider an input m(t) with a zero mean joint Gaussian density function

$$P_{m_1,m_2}(\alpha,\beta) = \frac{1}{2\pi\sigma^2\sqrt{1-\rho^2}} \exp\left[-\frac{\alpha^2-2\rho\alpha\beta+\beta^2}{2\sigma^2(1-\rho^2)}\right], \quad 1<\rho<1 \qquad (3.5\text{-}22)$$

where σ is the variance of the random variables m_1 and m_2 and ρ is the covariance coefficient of m_1 and m_2:

$$\rho = \frac{E\{m_1\, m_2\}}{\sigma^2} \qquad (3.5\text{-}23)$$

Let the LDM response be bounded between +3V and -3V. Let the step size S be normalized to unity. The maximum number of levels is found to be 8. The value of the LDM tree state, at steady state is:

$$\underline{X} = \{3.5, 2.5, 1.5, 0.5, -0.5, -1.5, -2.5, -3.5\} \qquad (3.5\text{-}24)$$

Using Eqs. (3.9-6), (3.5-11)-(3.5-18), and following the procedure outlined in Section 3.3, the initial probabilities, with $\rho = 0.99$ and $\sigma = 1$ as parameters, are found to be:

$$\begin{aligned} &Pr[x=3.5] = 0.0026, \quad Pr[x=-3.5] = 0.0026 \\ &Pr[x=2.5] = 0.0300, \quad Pr[x=-2.5] = 0.0300 \\ &Pr[x=1.5] = 0.1465, \quad Pr[x=-1.5] = 0.1465 \\ &Pr[x=0.5] = 0.3167, \quad Pr[x=-0.5] = 0.3167 \end{aligned} \tag{3.5-25}$$

Using Eqs. (3.5-11)-(3.5-18), the transition probability matrix, $\underline{\underline{P(1)}}$, is found to be:

$$\underline{\underline{P(1)}} = \begin{bmatrix} 0.0855 & 0 & 0.9144 & 0 & 0 & 0 & 0 & 0 \\ 0 & .2569 & 0 & .7429 & 0 & 0 & 0 & 0 \\ 0.0160 & 0 & .4768 & 0 & .5070 & 0 & 0 & 0 \\ 0 & .0704 & 0 & .6948 & 0 & .2345 & 0 & 0 \\ 0 & 0 & .2345 & 0 & .6948 & 0 & .0704 & 0 \\ 0 & 0 & 0 & .5070 & 0 & .4767 & 0 & .0160 \\ 0 & 0 & 0 & 0 & .7430 & 0 & .2569 & 0 \\ 0 & 0 & 0 & 0 & 0 & .9144 & 0 & .0855 \end{bmatrix} \tag{3.5-26}$$

It is clear that the chain is homogeneous. To see if stationarity holds we proceed as follows: Multiplying the transition matrix to the 40-power and the 41-power the following matrix was obtained:

$$\underline{\underline{P}}(40)=\underline{\underline{P}}(41)=\begin{bmatrix} .0051 & 0 & .2933 & 0 & .6340 & 0 & .0601 & 0 \\ 0 & .0601 & 0 & .6340 & 0 & .2933 & 0 & .0051 \\ .0051 & 0 & .2933 & 0 & .6340 & 0 & .0601 & 0 \\ 0 & .0601 & 0 & .6340 & 0 & .2933 & 0 & .0051 \\ .0051 & 0 & .2933 & 0 & .6340 & 0 & .0601 & 0 \\ 0 & .0601 & 0 & .6340 & 0 & .2933 & 0 & .0051 \\ .0051 & 0 & .2933 & 0 & .6340 & 0 & .0601 & 0 \\ 0 & .0601 & 0 & .6340 & 0 & .2933 & 0 & .0051 \end{bmatrix} \quad (3.5\text{-}27)$$

In Section 3.3 it was mentioned that the set of states of the LDM tree contains two disjoint sets:

$$\underline{X(T)} = [\{x_1x_3x_5x_7\},\{x_2x_4x_6x_8\}] = [\underline{X_0(T)},\underline{X_e(T)}] \quad (3.5\text{-}28)$$

The zeros in the odd columns correspond to the probability of a transition from an even state to an odd state. Since the two sets of states are disjoint, the transition probability is zero as expected. The same result applies for zeros in the even columns.

Now, $\underline{\underline{P}}(40)$ can be transformed into a matrix $\underline{\underline{P_T}}(40)$ (see Appendix H) where

$$\underline{\underline{P}}_T(40)=\begin{bmatrix} .0051 & .2933 & .6340 & .0601 & \vdots & & \\ .0051 & .2933 & .6340 & .0601 & \vdots & & \\ .0051 & .2933 & .6340 & .0601 & \vdots & & \\ .0051 & .2933 & .6340 & .0601 & \vdots & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \underline{\underline{0}} & \\ & & & & \vdots & .0601 \; .6340 \; .2933 \; .0051 & \\ & \underline{\underline{0}} & & & \vdots & .0601 \; .6340 \; .2933 \; .0051 & \\ & & & & \vdots & .0601 \; .6340 \; .2933 \; .0051 & \\ & & & & \vdots & .0601 \; .6340 \; .2933 \; .0051 & \end{bmatrix} \qquad (3.5\text{-}29)$$

or in shorthand notation:

$$\underline{\underline{P}}_T(40) = \begin{bmatrix} \underline{\underline{A}}_0 & \vdots & \underline{\underline{0}} \\ \cdots & \vdots & \cdots \\ \underline{\underline{0}} & \vdots & \underline{\underline{A}}_e \end{bmatrix} \qquad (3.5\text{-}30)$$

We observe that all the rows of submatrix $\underline{\underline{A}}_0$ are identical which means that the probability of the LDM being in odd state x_i is independent on starting state, providing the starting state is from the odd set of states. However, this is precisely what the limit of Eq. (G-20) means. Therefore we conclude that since the limit exists and since the chain is homogeneous, the process is stationary in the asymptotic (wide) sense.

Using $\underline{P(1)}$, initial condition probabilities, Eqs. (3.5-19) and (3.5-25), a correlation coefficient ρ of 0.99 and a variance σ of value 1., the first 12 terms of the autocorrelation function $R_x(\tau)$ are:

$$\begin{array}{lll} R_x(0) = 1.25600, & R_x(4) = 0.00069, & R_x(8) = -0.00007 \\ R_x(1) = 0.01502, & R_x(5) = 0.00021, & R_x(9) = -0.00008 \\ R_x(2) = 0.00554, & R_x(6) = 0.00002, & R_x(10) = -0.00008 \\ R_x(3) = 0.00201, & R_x(7) = -0.00004, & R_x(11) = -0.00008 \end{array} \qquad (3.5\text{-}31)$$

The autocorrelation function of the LDM estimate to a first order Markov input with a zero mean gaussian pdf,

is plotted in Figs. 3.5-2a to 3.5-2c, where each figure shows the autocorrelation function plotted for a fixed value of ρ and three different values of σ.

Results show that when the variance is comparable in magnitude to the bounds imposed on the LDM response (σ=2, N=3), the autocorrelation of the estimate signal follows an exponential decay. On the other hand when the variance is small (σ=0.5, N=3), the autocorrelation of the estimate signal is an impulse.

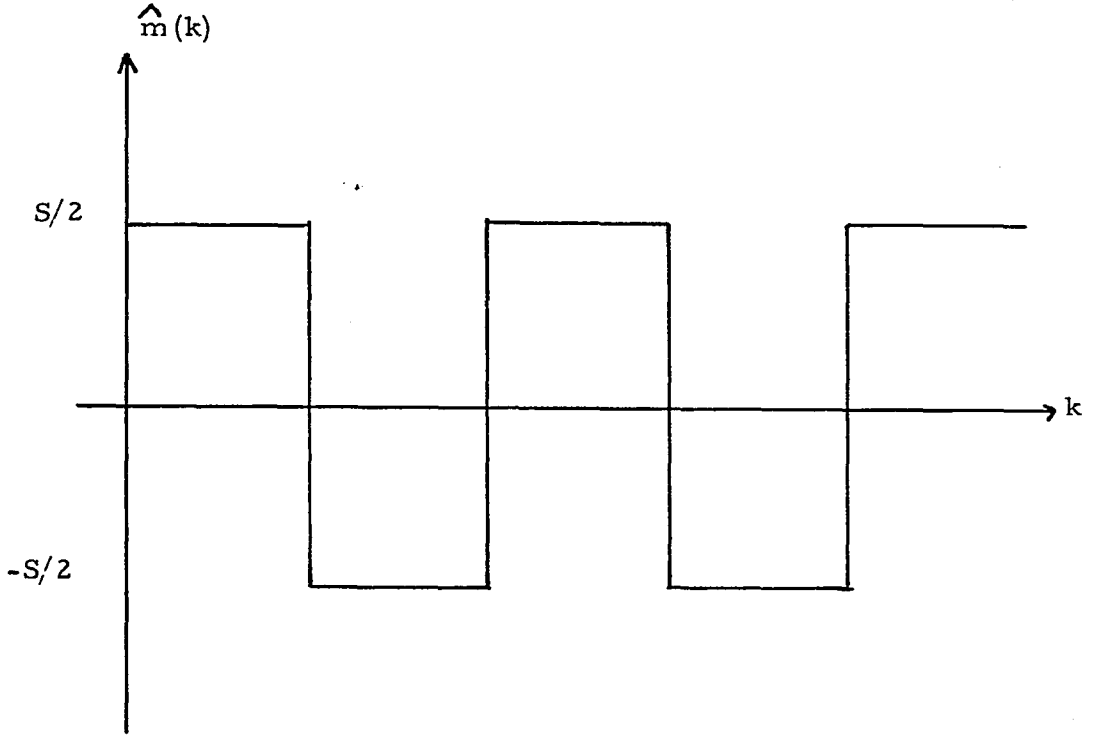


Fig. 3.1-1: Zero Input Response of LDM.

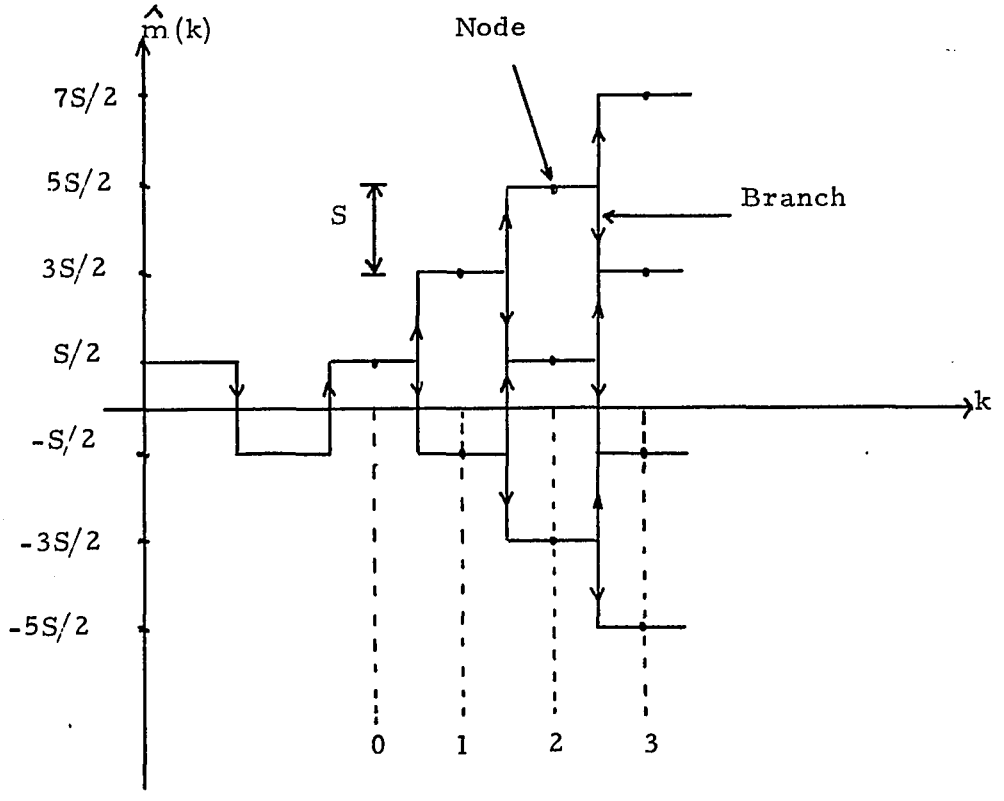


Fig. 3.2-1: The Estimate Tree Graph.

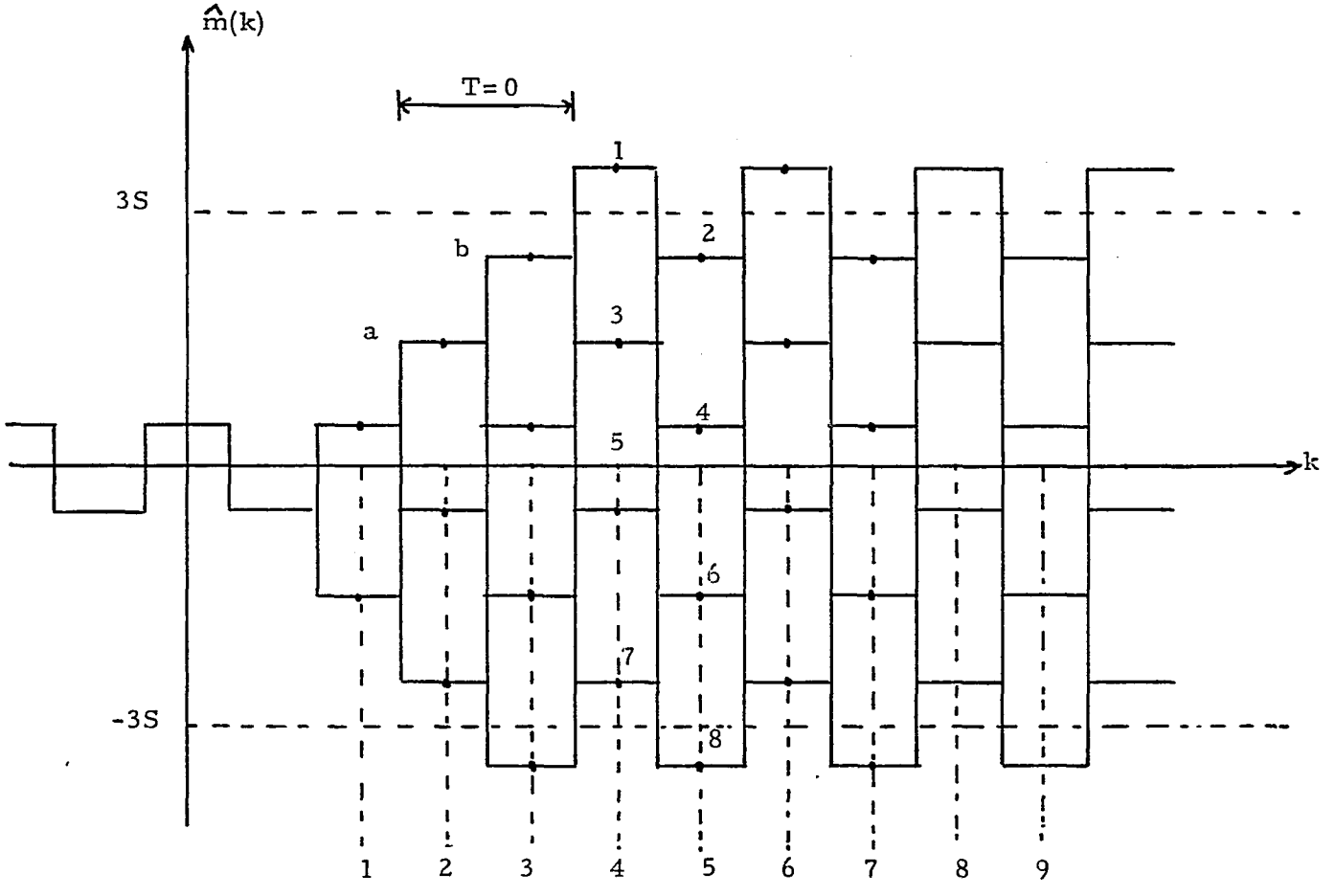


Fig. 3.2-2: Stochastic Tree for the L=8 Case.

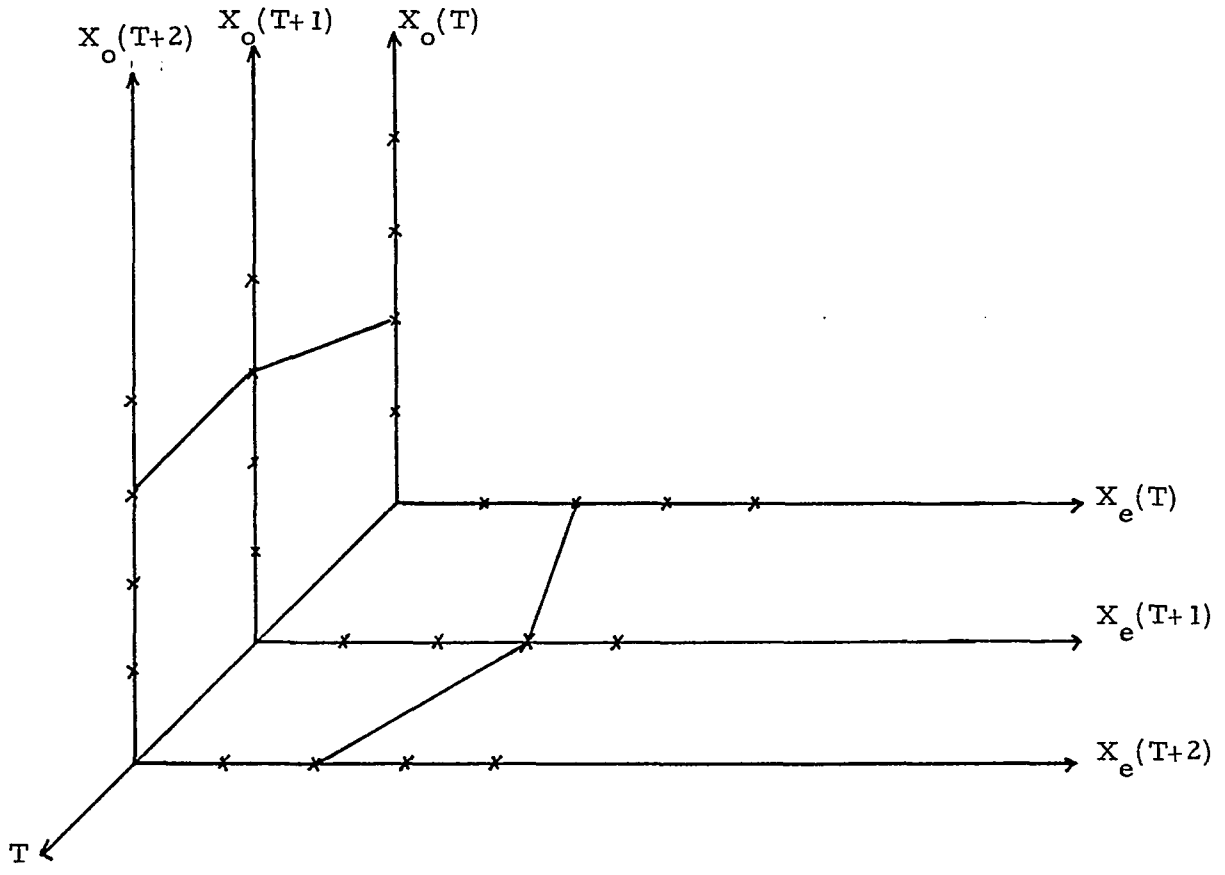


Fig. 3.3-1: Plane Representation of LDM Possible Levels for the L=8 Case.

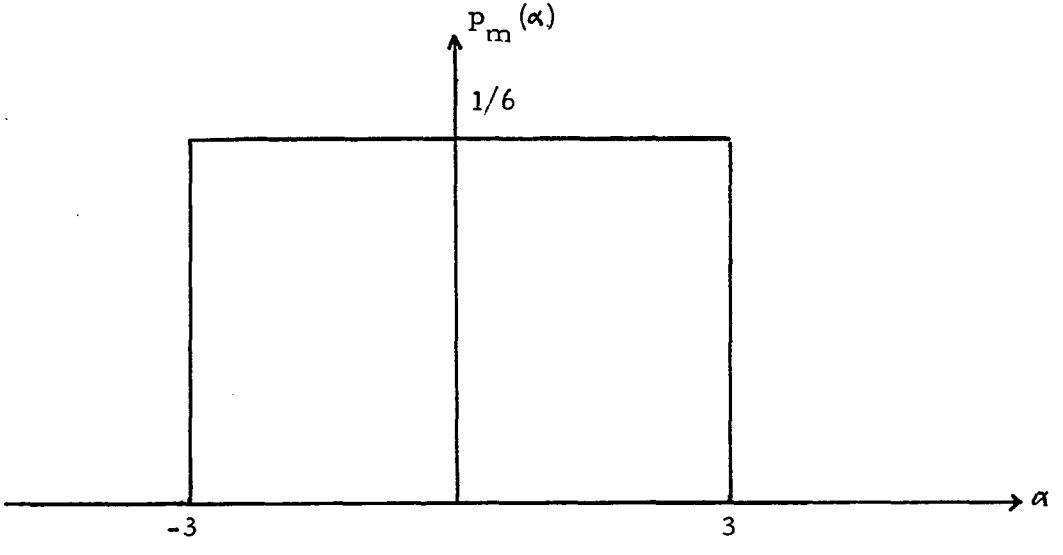


Fig. 3.4-1: Uniform Probability Density Function.

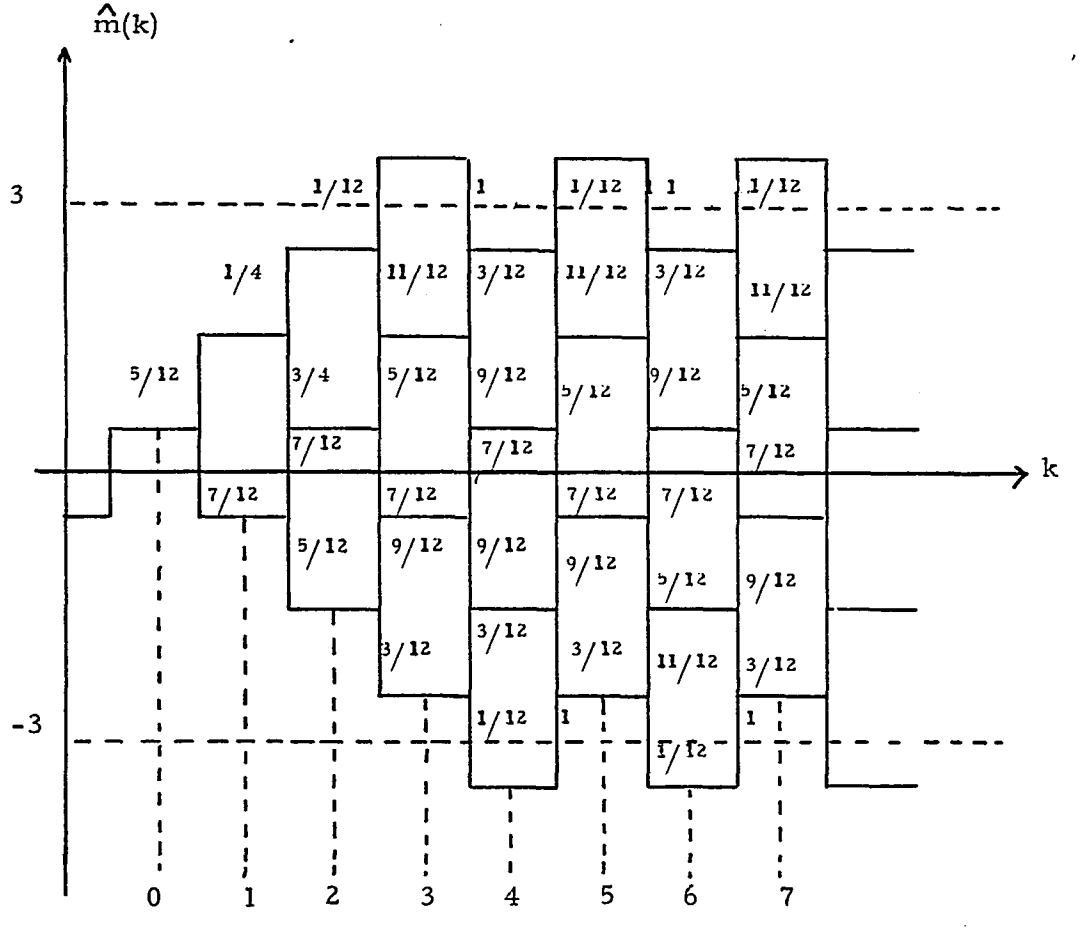


Fig. 3.4-2: Stochastic Tree Showing Transition Probabilities.

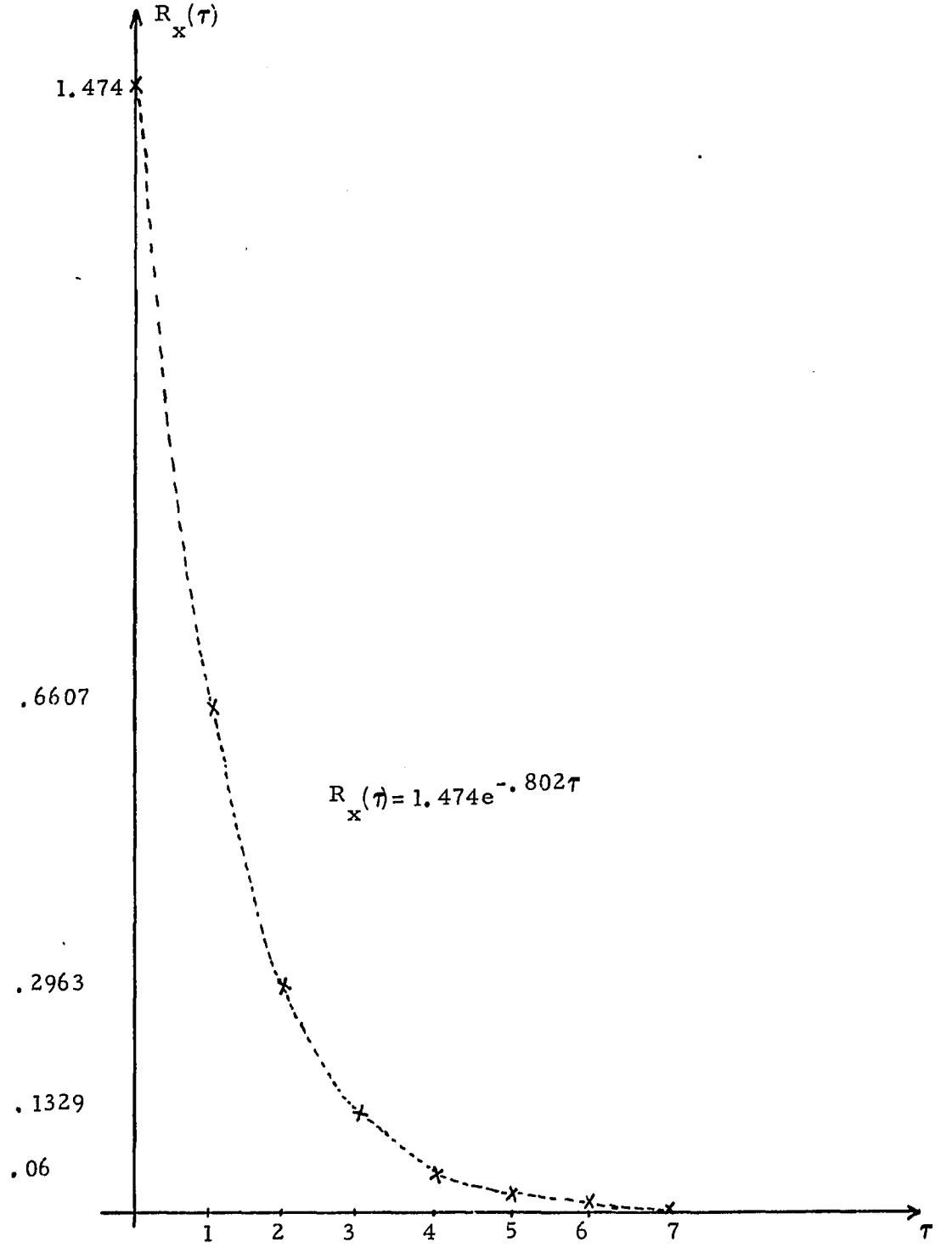


Fig. 3.4-3: 8-State Autocorrelation Function.

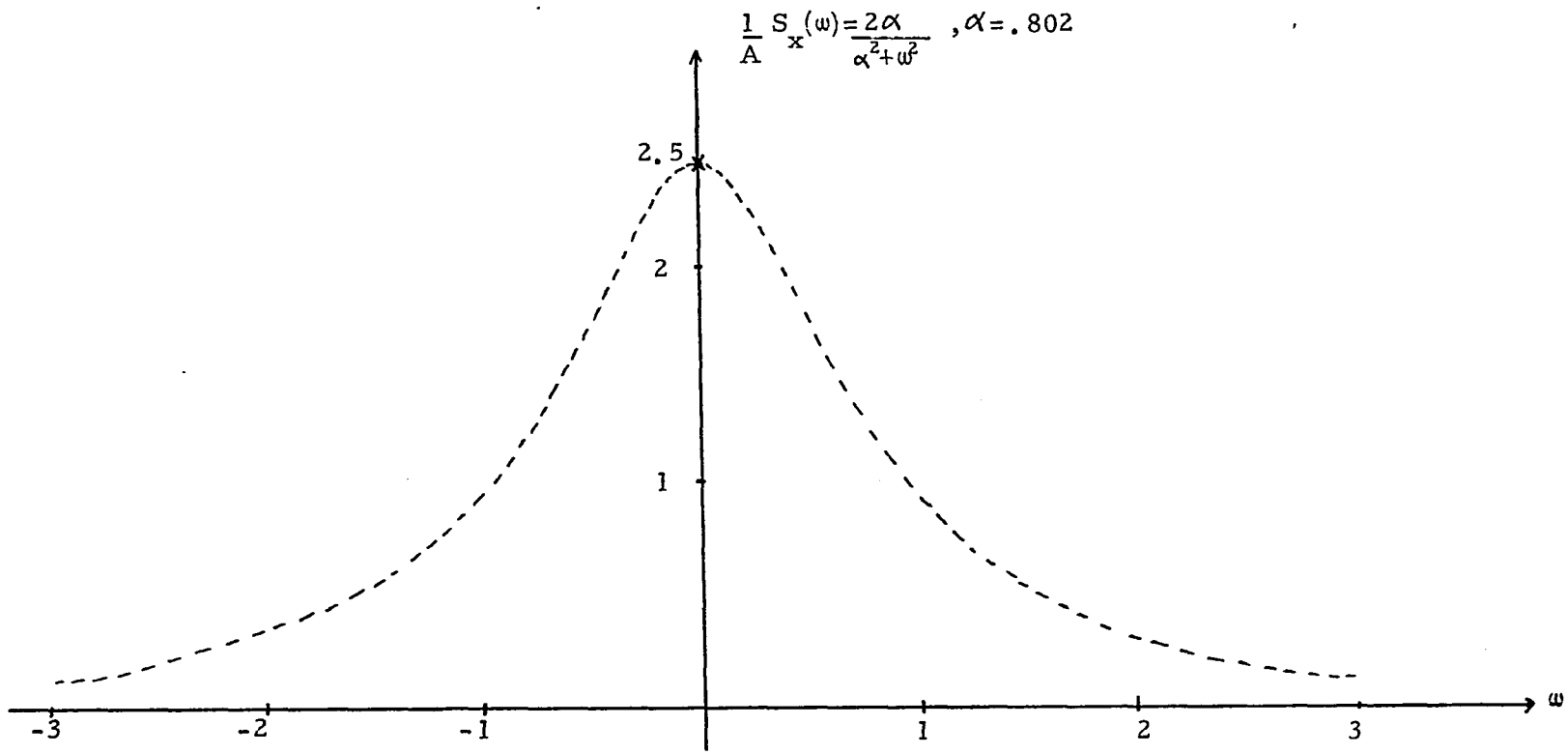


Fig. 3.4-4: The Power Spectral Density for the L=8 Case.

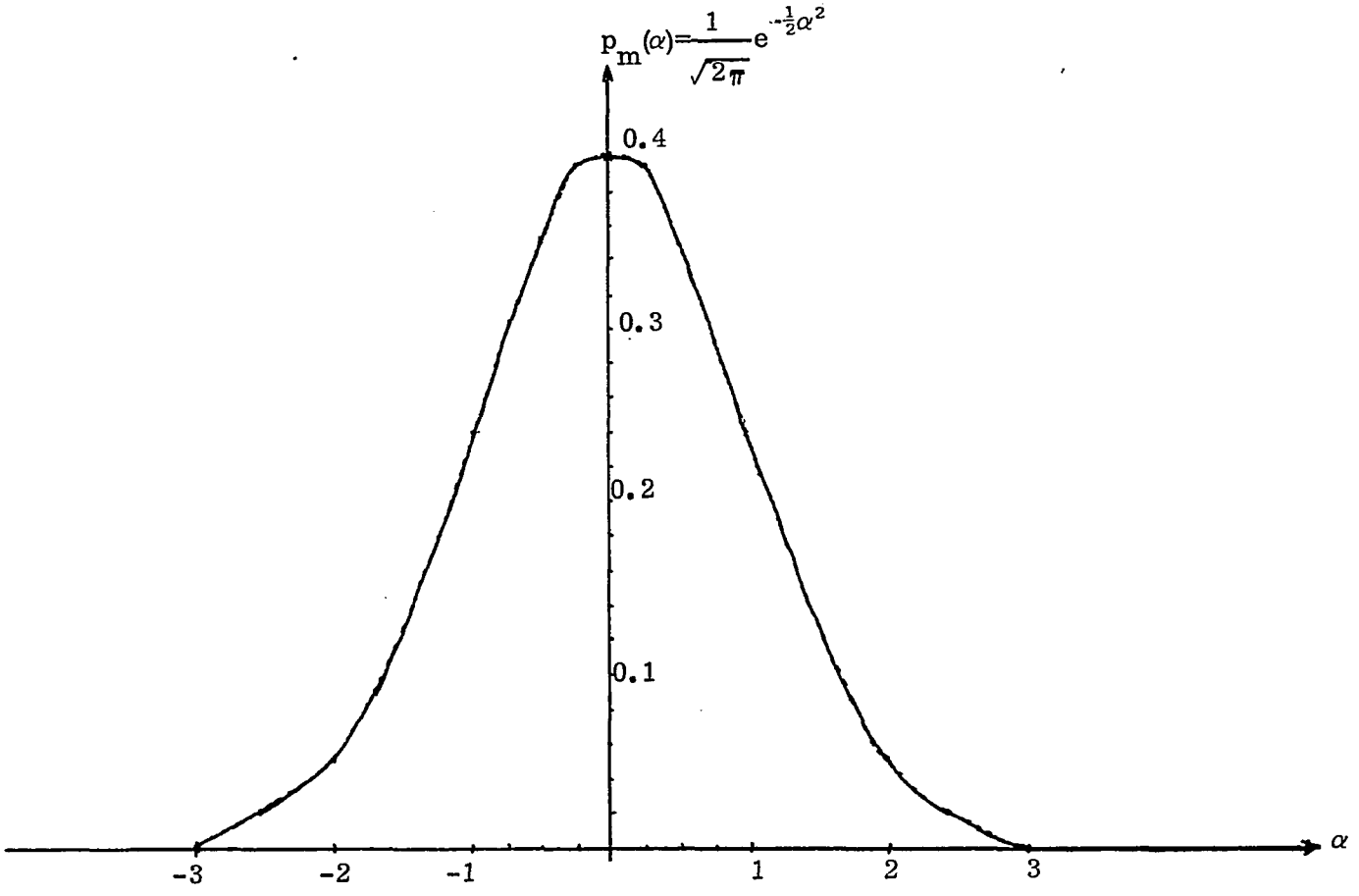


Fig. 3.4-5: Gaussian Probability Density Function(zero mean and σ=1).

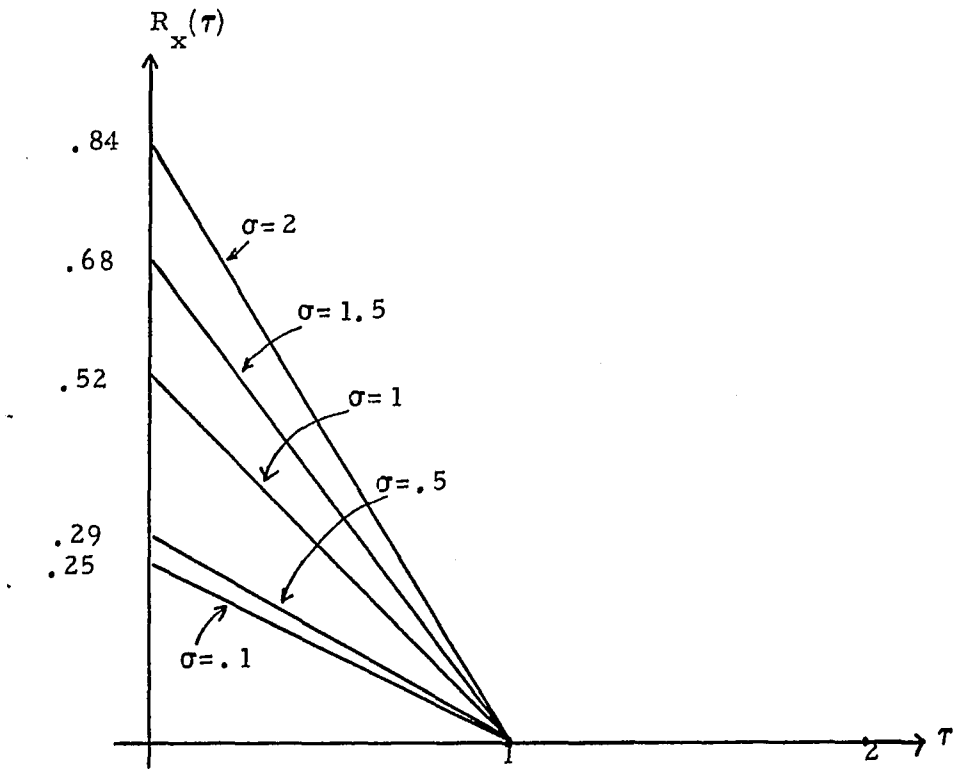


Fig. 3.4-6: Autocorrelation Function for Five Values of σ.

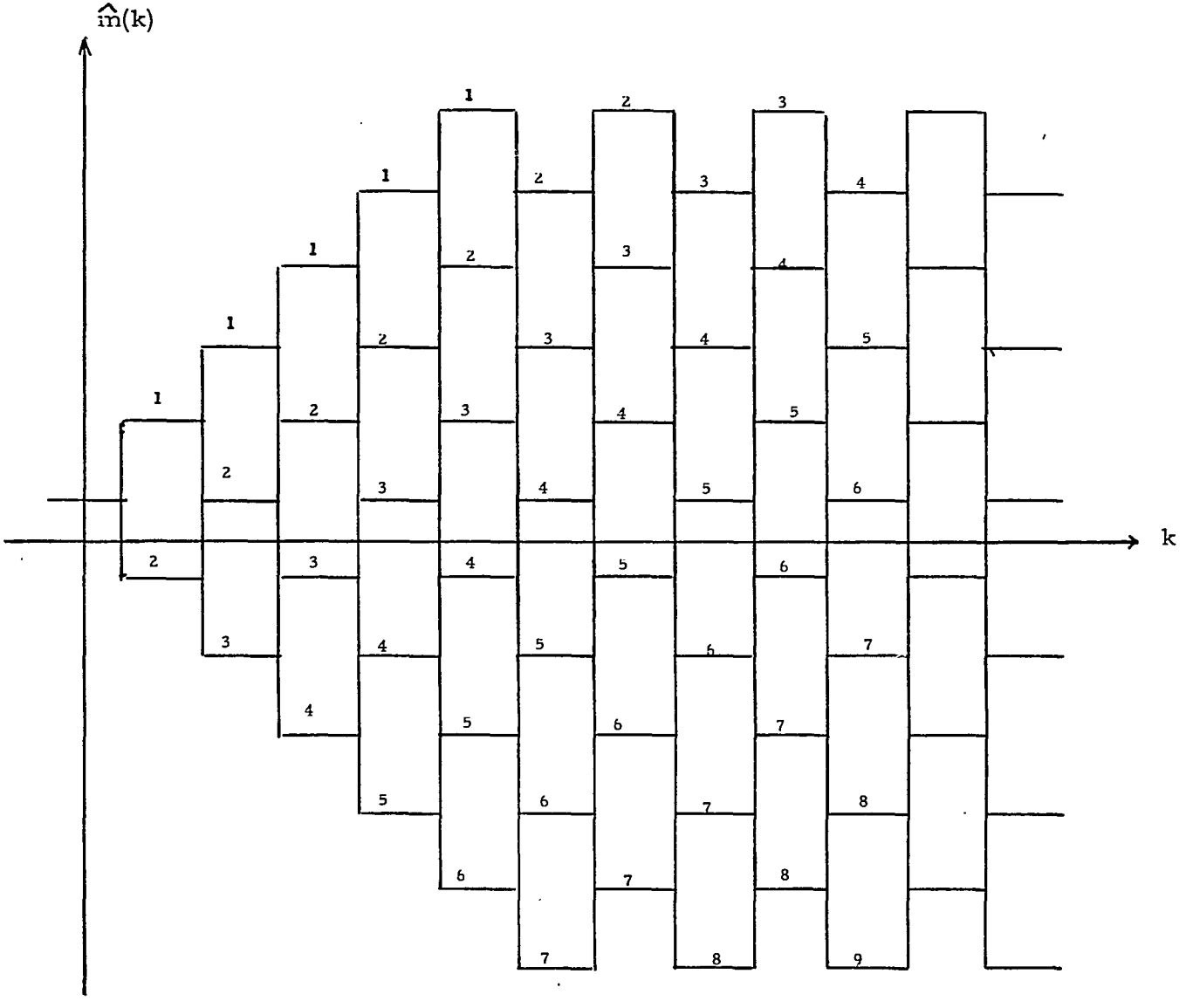


Fig. 3.5-1: Estimated Tree With Levels Assigned In An Ascending Order and Bounded Response.

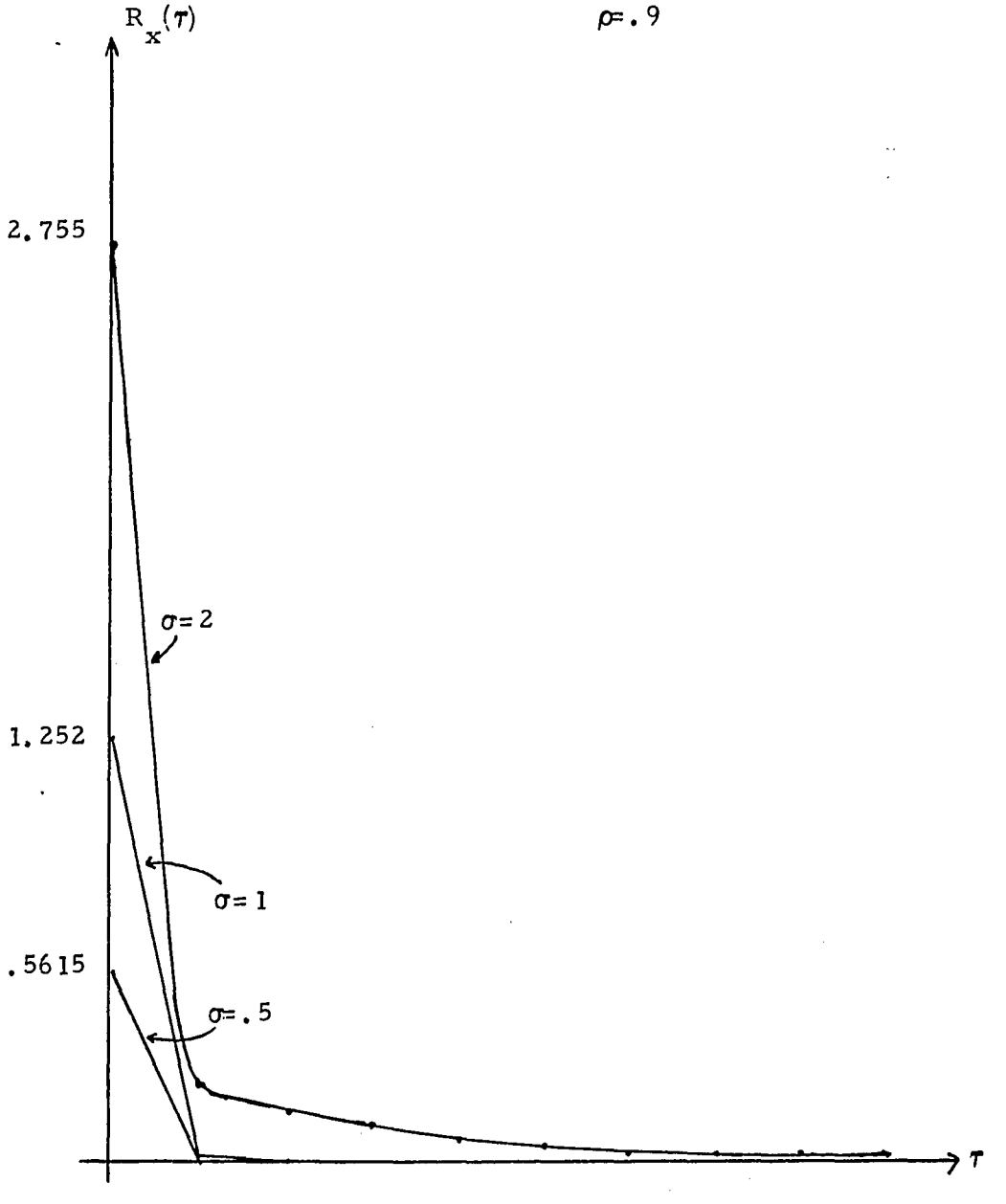


Fig. 3.5-2a: The Autocorrelation Function of the LDM Estimate to a First Order Markov Input With $\rho = .9$ and Three Values of σ.

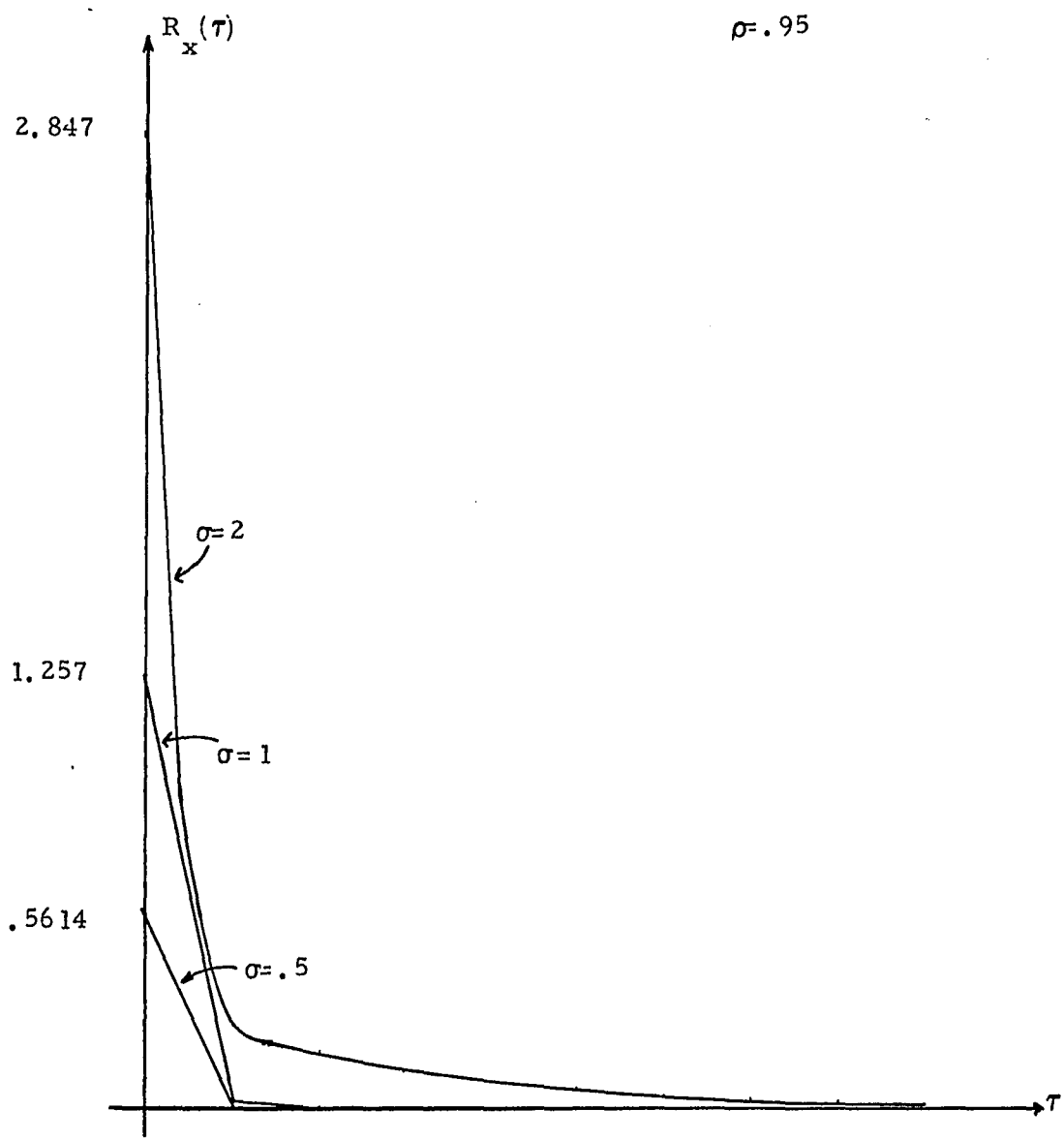


Fig. 3.5-2b: The Autocorrelation Function of the LDM Estimate to a First Order Markov Input With $\rho = .95$ and Three Values of σ.

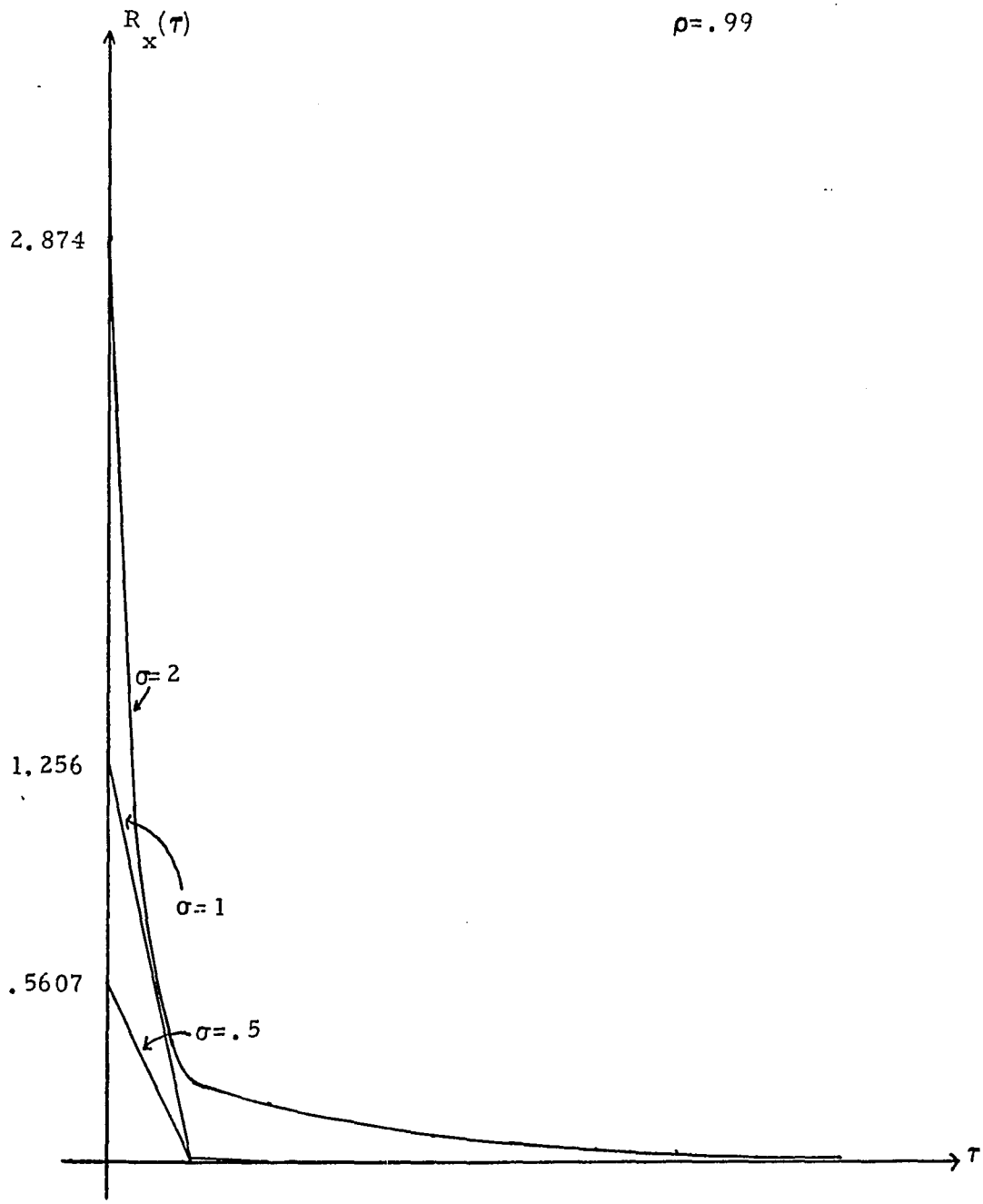


Fig. 3.5-2c:The Autocorrelation Function of the LDM Estimate to a First Order Markov Input With $\rho = .99$ and Three Values of σ.

Chapter 4

4.1 Conclusion

In this dissertation we have designed a programmable real time ADM voice processor and formulated the auto-correlation function of the LDM system.

Design requirements established in Chapter 2 include: a) Programmability, that is ability to execute instructions stored in memory. b) Fast instruction execution time. Since a complete processing cycle is less than 20 μsec an upper bound on instruction execution time was set to 160 nsec to enable the execution of 125 instructions/sample at 50 kHz sampling rate. c) Full duplex operation, that is the capability of simultaneously dealing with analog signals as well as with digital information. Since a transmitter and a receiver must be implemented, in most cases, on each side of a communication channel. Only the bit-slice class of cpu's meet requirement b. Of the three different bit-slices available, the Intel-3002 2 bit slice, lends itself, almost naturally, to a full duplex mode of operation with an instruction set which is versatile, yet simple, enough to allow such operation. Since the instruction cycle time is comparable to the other two bit-slices, the Intel 3002 2 bit slice was chosen. The voice processor as seen in Fig.2.2-5 has four major components: the CPU consisting

of five 2-bit slices allowing for a 10-bit dynamic resolution; the instruction memory containing the instruction set (12 bit words); the PC controlling instruction execution order; and the I/O devices consisting of 10-bits A/D and D/A converters.

The SVADM is implemented as an example of an ICDM. Flow chart implementation of encoder-decoder is shown in Figs. 2.3-2 and 2.3-2; with explanation of system operation. A solution to transmitter-receiver synchronization problem in the form of a leaky accumulator is shown and implemented in Fig. 2.3-1. Effects of channel errors on receiver operation are shown in Fig.2.3-10. To minimize such effects, saturation logic was incorporated in the receiver design. Response of a SVADM receiver employing a leaky accumulator and saturation logic was demonstrated in Fig.2.3-14. The smoothing action of an averaging filter seemed to improve voice quality at low sampling rates. However, using look ahead with SVADM algorithm did not seem to improve receiver performance. The CVSD was implemented as an example of a SCDM. Flow chart implementation of encoder-decoder was shown in Figs. 2.7-1 and 2.7-2, with explanation of system operation. Response of the CVSD encoder algorithm to voice input was demonstrated in Fig.2.4-3 for different sampling rates.

A switching receiver implementation, allowing communication between the CVSD and the SVADM systems, was designed and shown in Fig.3.5-2. Since 50% of speech is silence, the e(k) pattern, which is unique to each system was used to detect and identify which of the two algorithms currently operating.

In Chapter 3 we modeled the LDM system probabilistically. A Markov chain approach is used in obtaining the autocorrelation function of the LDM algorithm, without assuming that granular noise and slope overload noise are independent. An algorithm, to obtain the transition probabilities matrix and the steady state level probabilities, was developed for the zero-order and first-order Markov cases. Transition probabilities were shown to be a function of the input statistics alone. Knowledge of the transition probabilities matrix and the steady state level probabilities enabled the calculation of the autocorrelation function of the estimate signal. For the uniform, zero-order Markov case the autocorrelation function was found to be an exponential decay indicating that for such an input the LDM acts as a low pass filter. For a white Gaussian input the function was found to be an impulse indicating that for such an input the LDM acts as an all pass filter. For the first order Markov-Gauss input the

autocorrelation function was found to be an impulse when the variance was small compared to the bounds on the response and an exponential decay when the variance was comparable in magnitude to the imposed bounds. Results were demonstrated in Figs.3.5-2a-3.5-2c.

4.2 Suggestions for future research.

The voice processor is capable of storing DM algorithms and selecting a specified algorithm. In a communication link in which the other party employs one of the stored algorithms, the other party initially transmits a code indicating the desired algorithm, the voice processor receives the code and employs the selected algorithm. The question arises as to whether we can "learn", by receiving some prescribed "test-pattern", the algorithm employed. If we examine the spectrum of the bit stream transmitted by a DM encoder, we find that the spectrum has a component proportional to the original signal. Hence, if we low-pass filter the received bit stream, the input signal can be recovered although distorted. Such a procedure is equivalent to assuming that the delta modulator is linear and the step-size is always constant. If the stepsizes are now adjusted so as to minimize the distortion, we will have obtained the step-size algorithm. Such a technique can be employed to learn algorithms. Furthermore, most users employ one of two basic algorithms, ICDM's algorithm or SCDM's algorithm. If a user is known a priori to employ one

of the two algorithms it is an easy matter to determine the parameters of his system. It is left for further research to devise "learning" techniques whereby the voice processor can constract a receiver capable of decoding a received bit stream.

Yet another area of interest is the study of adaptive predictive coders (APC). With the introduction of fast hardware multipliers (a 16 by 16 multiplication in 120 nsec) and fast memories (typical access time of ECL memories is 15 nsec), the voice processor can be modified to include above options and thus programmable correlation receivers employing APC algorithms can be designed and tested in real time.

Appendix A

Instruction Set

Mnemonic	Machine Code	Description
Arithmetic Operations		
ILR X	0001 xxxx $\overline{CI}$010	CI+(x)→ Ac,X
ADD X	0000 xxxx $\overline{CI}$010	CI+(x)+(Ac)→ Ac,X
CSR X	0101 xxxx $\overline{CI}$010	CI-1→ X
SDR X	0100 xxxx $\overline{CI}$010	(Ac)-1+CI→ X
INR X	0111 xxxx $\overline{CI}$010	(X)+CI→ X
ADR X	0110 xxxx $\overline{CI}$010	(X)+(Ac)+CI→ X
CLR X	1001 xxxx 1010	0→ X
NOP X	1101 xxxx 1010	(X)→ X
CMR X	1111 xxxx 1010	$(\overline{X})$→ X
RRA Y	0001 yyyy 1010	y(o)→y(n-1),→y(n-i)→y(n-i-1)
CIA Y	0011 yyyy $\overline{CI}$010	$(\overline{y})$+CI→ y
DCA Y	0010 yyyy $\overline{CI}$010	(y)-1+CI→ y
INA Y	0111 yyyy $\overline{CI}$010	(y)+CI→ y
NOP Y	1101 yyyy $\overline{CI}$010	(y)→y
CMA Y	1111 yyyy 1010	$(\overline{y})$→y
Logic operations		
ANR X	1000 xxxx 1010	(x)∧(Ac)→x
ORR X	1100 xxxx 1010	(x)∨(Ac)→x
XNR X	1110 xxxx 1010	$\overline{(x)(+)(Ac)}$→x
ANI y	1000 yyyy 1010	(y)∧$(\overline{I})$→y
ORI y	1100 yyyy 1010	(y)∨(I)→y
XNI y	1110 yyyy 1010	$\overline{(y)(+)(\overline{I})}$→y

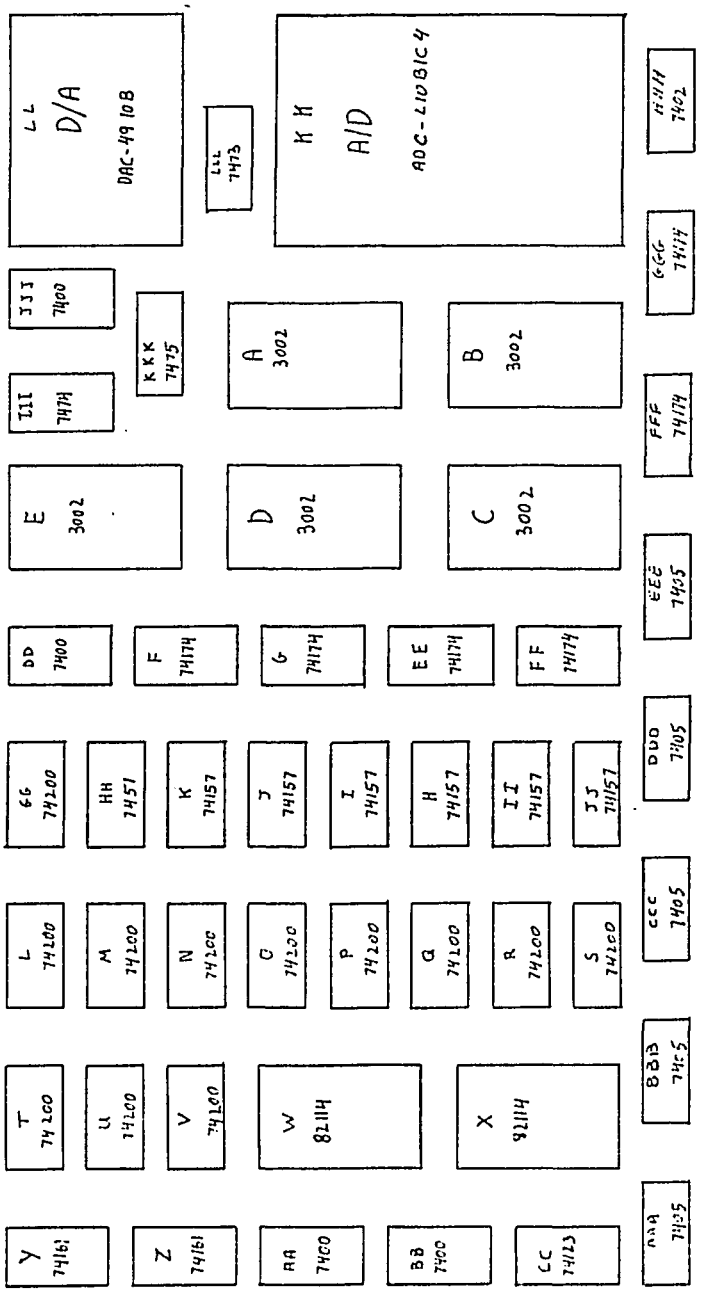
Input Operations					
LDI	Y	0100	yyyy	CI010	$(\overline{I})$-1+CI→Y
AIA	Y	0110	yyyy	CI010	$(\overline{I})$+CI+Y→Y
AMA	Z	0000	zzzz	CI010	$(\overline{M})$+(Z)+CI→Z
LDI	Z	0001	zzzz	CI010	$(\overline{DI})_{MSB}\rightarrow Z_{MSB}$, $CI\rightarrow Z_{MSB}$ $z_i=0$ $i\neq 1,n-1$
LDE	Z	1101	zzzz	1010	$(\overline{DI})\rightarrow Z_{MSB}$, $z_i=0$ $i\neq n-1$
LOD	Z	1111	zzzz	1010	$(DI)\rightarrow Z_{MSB}$, $z_{i=1}$ $i\neq n-1$
Output Operations					
Out	Ac	1101	0000	1000	(Ac)→ D/A
Out	X_d	0011	xxxx	1010	$(X)_{MSB}$→ Digital output
Branching Operations					
JMP	LOC	$A_7A_6A_5A_0$	$A_4A_3A_2A_1$	1011	unconditional jump to location
SKP	AC				$A_7A_6A_5A_4A_3A_2A_1A_0$. Next instruction is to be skipped if (Ac)≥0, providing the instruction to be skipped is not a jump instruction. If it is, then the jump condition is on negative accumulator content [i.e. (Ac)<0].

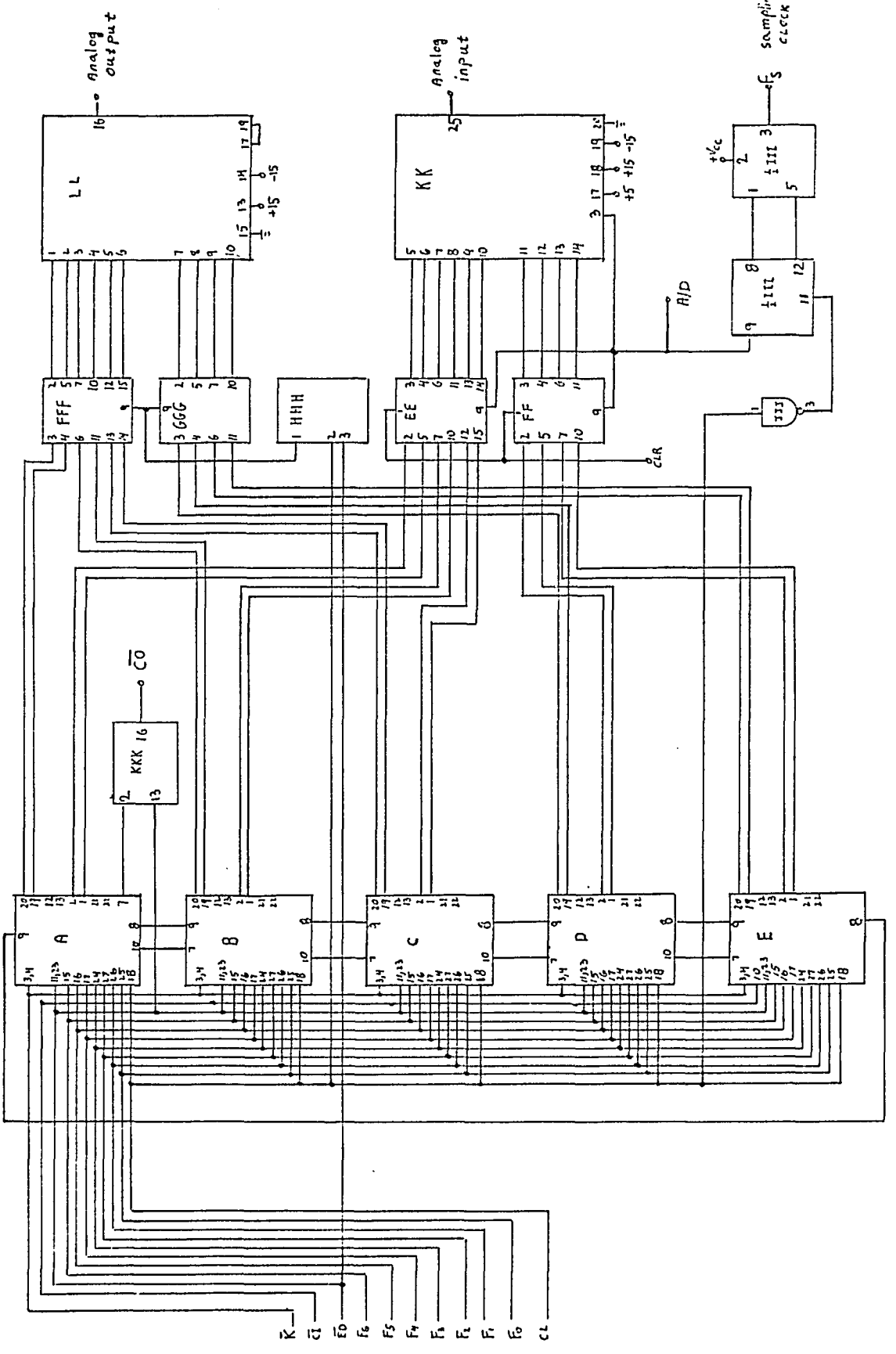
X Y and Z are chosen from table A1. I is a 10 bit sampled analog input. Ac is the accumulator

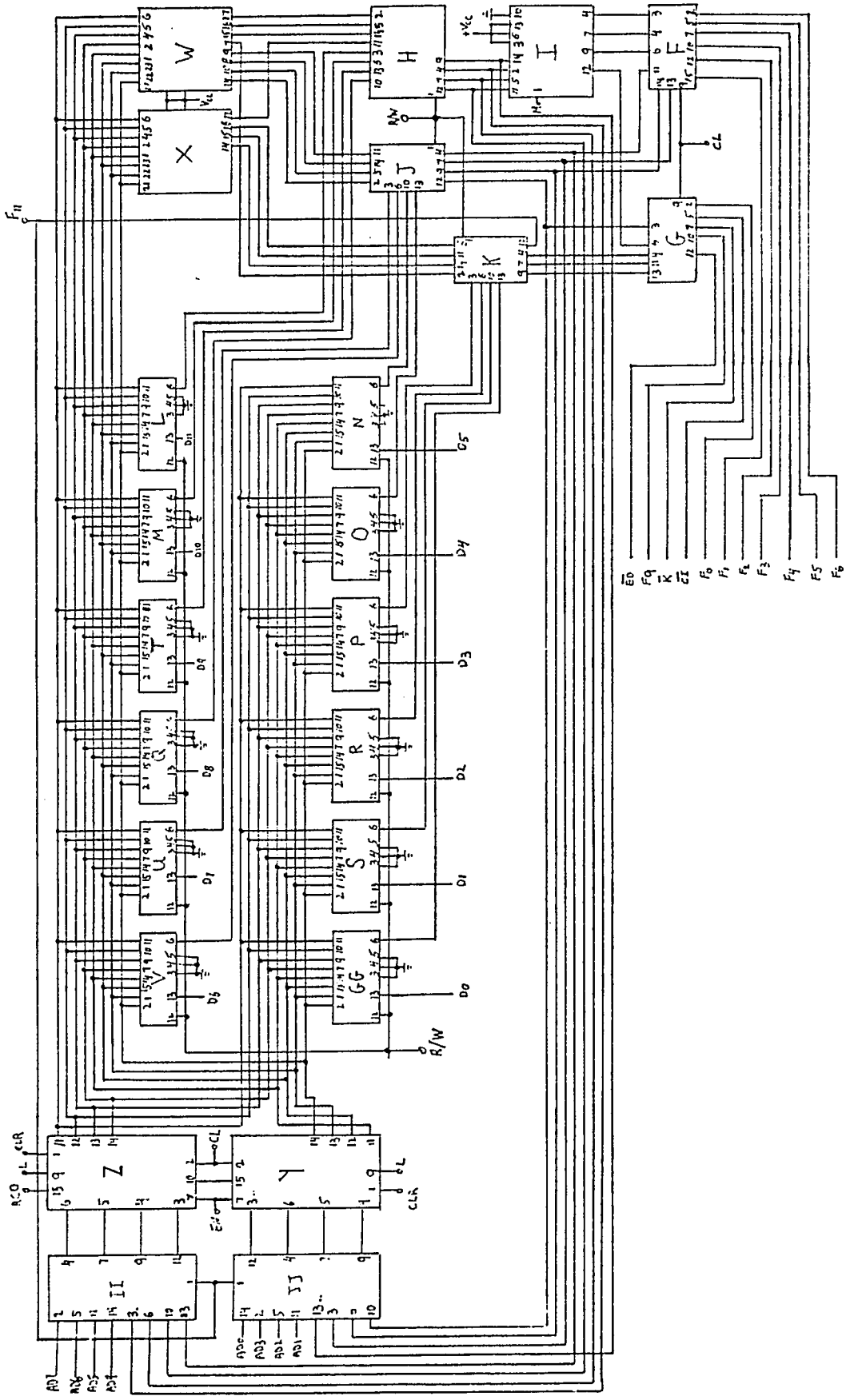
X group	y group	Z group
X xxxx	y yyyy	Z zzzz
R_0 0000	T 1110	T 1010
R_1 0001	Ac 1111	Ac 1011
R_2 0010		
R_3 0011		
R_4 0100		
R_5 0101		
R_6 0110		
R_7 0111		
R_8 1000		
R_9 1001		
T 1100		
Ac 1101		

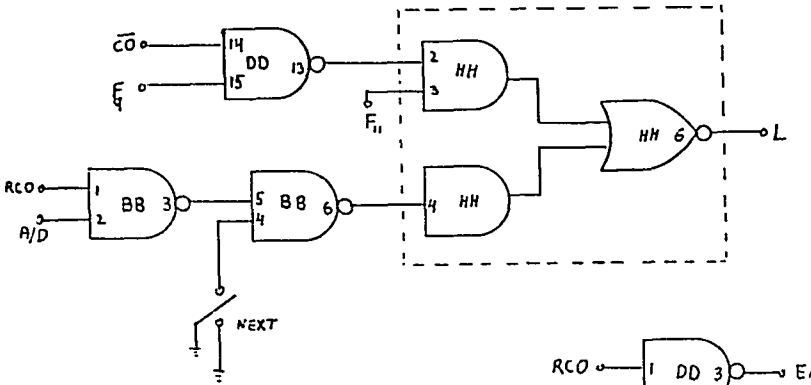
Table A1: Registers codes

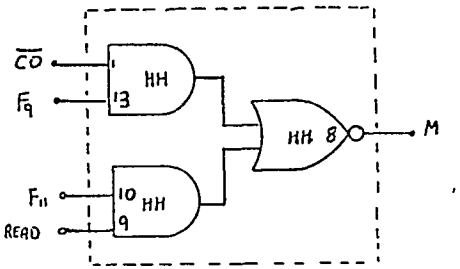
APPENDIX B


Y 74161
Z 74161
AA 7400
BB 7400
CC 74123
T 74200
U 74200
V 74200
W 82114
X 91114
L 74200
M 74200
N 74200
O 74200
P 74200
Q 74200
R 74200
S 74200
GG 74200
HH 7451
K 74157
J 74157
I 74157
H 74157
II 74157
JJ 74157
DD 7400
F 74174
G 74174
EE 74174
FF 74174
E 3002
D 3002
C 3002
A 3002
B 3002
III 74174
JJJ 7400
KKK 7475
LL D/A DAC-49 10B
LLL 7473
KK A/D ADC-10 BIC 4
MMM 7402
NNN 7405
BBB 7405
CCC 7405
DDD 7405
EEE 7405
FFF 74174
GGG 74174

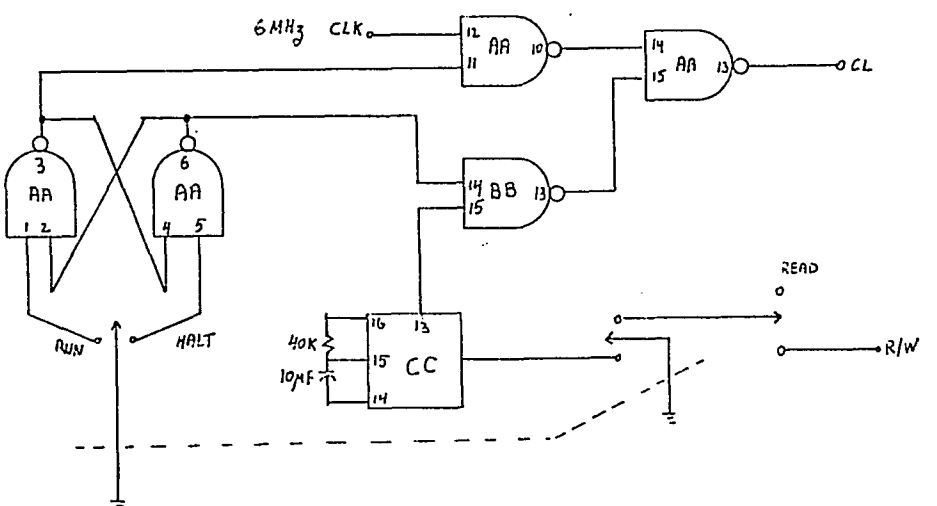

Analog output
Analog input
sampling clock
LL
KK
FFF
GGG
HHH
EE
FF
KKK 16
A
B
C
D
E
AID
CLR
CO
K
CI
EO
F6
F5
F4
F3
F2
F1
F0
CL

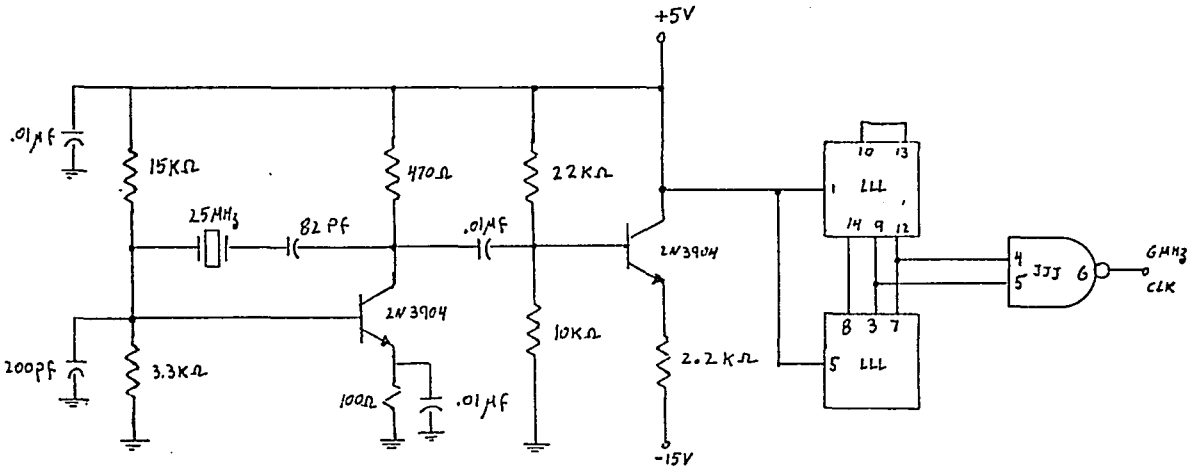

W
X
H
I
J
K
G
F
L
M
N
O
T
P
Q
R
U
S
V
GG
Z
Y
II
III
CLR
CL
R/W
+Vcc
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
A00
A01
A02
A03
A04


CO
F9
DD
F11
HH
HH 6
L
RCO
A/D
BB 3
BB 6
HH
NEXT


CO
F9
HH
F11
READ
HH
HH 8
M

RCO
DD 3
EN


6MHz
CLK
AA
AA
CL
AA
AA
BB
READ
R/W
40K
10MF
CC
RUN
HALT



D0 - D11 : Instruction Word
AD1 - AD7 : Address Switches (SPDT)
R/W : Read - Write switch (SPDT)
CLR : Master clear (Momentary)
RUN/HALT : RUN - don't RUN switch (DPDT)

Appendix C

SVADM Encoder - Decoder Programs

C-1 SVADM Encoder

$R_3 = \hat{m}(k)$ $R_4 = s(k)$ $R_6 = e(k-1)$

Address	Mnemonic	Machine Code	Description
xmitter	ILR R_4	0001 0100 1010	
	SDR T	0100 1100 0010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	$\|s(k)\| \rightarrow T$
	ILR R_3	0001 0011 1010	$\hat{m}(k) \rightarrow Ac$
	AIA Ac	0110 1111 0010	$\hat{m}(k) - m(k) \rightarrow Ac$
	SDR R_5	0100 0101 0010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	CIA T	0011 1110 0010	$\|s(k)\|e(k) \rightarrow T$
	ILR R_6	0001 0110 1010	$So e(k-1) \rightarrow Ac$
	ADD T	0000 1100 1010	$s(k+1) \rightarrow Ac, T$
	SDR R_4	0100 0100 0010	$s(k+1) \rightarrow R_4$
	ILR R_3	0001 0011 1010	$\hat{m}(k) \rightarrow Ac$
	ADD T	0000 1100 1010	$\hat{m}(k+1) \rightarrow Ac, T$

Address	Mnemonic	Machine Code	Description
	SDR R_3	0100 0011 0010	m(k+1) R_3
	CMR Ac	1111 1101 1010	
	out D/A	1101 0000 1000	output m(k+1). Optional, replace by a NOP when not needed
	ILR R_5	0001 0101 1010	
	out A_D	0011 1101 1010	output digital bit.
	CSR T	0101 1100 1010	-1→ T
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	e(k)→ T
	ILR T	0001 1100 1010	
	SDR R_6	0100 0110 0010	delay e(k) as e(k-1)→R_6
	Jmp WAIT	1111 1111 1011	wait for new input sample

C-2 SVADM Decoder

$R_1 = e(k-1)$, $R_7 = \hat{m}(k)$, $R_8 = s(k)$

Address	Mnemonic	Machine Code	Description
RCVR	ILR R8	0001 1000 1010	s(k)→ Ac
	SDR T	0100 1100 0010	s(k)→ T, Ac
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	s(k)→T
	LDC Ac	1101 1011 1010	$(\overline{DI})\rightarrow(Ac)_{MSB}$
	SDR R5	0100 0101 0010	Temporarily store e(k) in R_5
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	CIA T	0011 1110 0010	|s(k)| e(k) T
	ILR T	0001 1100 1010	
	SDR R8	0100 1000 0010	|s(k)|→e(k)→ R_8
	CSR T	0101 1100 1010	-1 →T
	ILR R1	0001 0001 1010	e(k-1)→ Ac
	SDR R8	0000 1000 1010	s(k+1)→ R_8,Ac
	Jmp leak		
Back	Jmp Sat		
cont	ILR R7	0001 0111 1010	$\hat{m}(k+1)\rightarrow$ Ac
	CMA Ac	1111 1111 1010	
	Out D/A	1101 0000 1000	output RCVR estimate

Address	Mnemonic	Machine Code	Description
	ILR R_5	0001 0101 1010	
	CSR T	0101 1100 1010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	
	SDR R_1	0100 0001 0010	delay e(k) as e(k-1)→ R_1
	Jmp Wait	1111 1111 1011	

C-3 Leaky Integrator

In this program the estimate $\hat{m}(k)$ will leak off towards 0 volts in steps of 10 mV if the signs of $s(k+1)$ and $\hat{m}(k)$ are the same.

Address	Mnemonic	Machine Code	Description
Leak	ILR R8	0001 1000 1010	$s(k+1) \rightarrow Ac$
	SDR T	0100 1100 0010	$s(k+1) \rightarrow Ac, T$
	ILR R7	0001 0111 1010	$\hat{m}(k) \rightarrow Ac$
	XNR T	1110 1100 1010	$\overline{\hat{m}(k) \oplus s(k+1)} \rightarrow T$
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	Jmp Back		signs not same
	ILR R7	0001 0111 1010	signs the same, $\hat{m}(k) \rightarrow Ac$
	SNA	0000 1101 1110	
	Jmp Pos		$\hat{m}(k) > 0$
	INR R7	0111 0111 0010	$\hat{m}(k) < 0$, $\hat{m}(k) = m(k)+1$
	Jmp Back		
Pos	CSR Ac	0101 1101 1010	$(-1) \rightarrow Ac$
	ADD R7	0000 0111 1010	$\hat{m}(k) = \hat{m}(k)-1$
	Jmp Back		

C-4 Saturation Logic Program

Address	Mnemonic	Machine Code	Description
Sat	ILR R7	0001 0111 1010	$\hat{m}(k) \rightarrow Ac$
	SNA	0000 1101 1110	
	Jmp GEQ		$\hat{m}(k) \geq o$
	ILR R8	0001 1000 1010	$\hat{m}(k) < o$, $s(k+1) \rightarrow Ac$
	SNA	0000 1101 1110	
	Jmp cont		$s(k+1) > o$, $\hat{m}(k) < o$
	ILR R8	0001 1000 1010	$\hat{m}(k) < o$, $s(k+1) < o$
	ADD R7	0000 0111 1010	$\hat{m}(k+1) \rightarrow R7$, Ac
	SNA	0000 1101 1110	
	Jmp sat neg		$\hat{m}(k+1) \geq o$
	Jmp cont		$\hat{m}(k+1) < o$
Satneg	CLR Ac	1001 1101 1010	
	INR Ac	0111 1101 0010	
	RRA Ac	0001 1111 1010	
	SDR R7	0100 0111 0010	$(512)_{10} \rightarrow R7$, $\hat{m}(k+1)$ most neg.#
	Jmp cont		
GEQ	ILR R8	0001 1000 1010	$\hat{m}(k) \geq o$, $s(k+1) \rightarrow Ac$
	SNA	0000 1101 1110	
	Jmp Prob		$s(k+1) \geq o$, $\hat{m}(k) \geq o$
	Jmp cont		$s(k+1) < o$, $\hat{m}(k) \geq o$

Address	Mnemonic	Machine Code	Description
Prob	1LR R_8	0001 1000 1010	
	ADD R_7	0000 0111 1010	$\hat{m}(k+1) \rightarrow R_7$, Ac
	SNA	0000 1101 1110	
	Jmp cont		$\hat{m}(k+1) \geq o$
	CLR Ac	1001 1101 1010	$\hat{m}(k+1) < o, s(k+1) \geq o, \hat{m}(k) > o$
	1NR Ac	0111 1101 0010	
	RRA Ac	0001 1111 1010	
	CMA Ac	1111 1111 1010	
	SDR R_7	0100 0111 0010	
	JMP cont		

Appendix D

SVADM encoder with 2-bit delayed encoding

Delayed encoding algorithm:

$|s(k)| > 2S_o$

$$\hat{m}(k+1)' = \hat{m}(k) + |s(k)| + S_o e(k-1) \qquad \hat{m}(k+2)' = \hat{m}(k) + S_o \qquad \text{D.1-1}$$

$$\hat{m}(k+1)'' = \hat{m}(k) - |s(k)| + S_o e(k-1) \qquad \hat{m}(k+2)'' = \hat{m}(k) - S_o \qquad \text{D.1-2}$$

$S_o \leq |s(k)| \leq 2S_o$

$$\hat{m}(k+1)' = \hat{m}(k) + S_o \qquad \hat{m}(k+2)' = \hat{m}(k) \qquad e(k-1) = -1 \qquad \text{D.1-2}$$

$$\hat{m}(k+1)'' = \hat{m}(k) - |s(k)| - S_o \qquad \hat{m}(k+2)'' = \hat{m}(k) - S_o$$

$$\hat{m}(k+1)' = \hat{m}(k) + |s(k)| + S_o \qquad \hat{m}(k+2)' = \hat{m}(k) + S_o \qquad e(k-1) = 1 \qquad \text{D.1-3}$$

$$\hat{m}(k+1)'' = \hat{m}(k) - S_o \qquad \hat{m}(k+2)'' = \hat{m}(k)$$

$$\Delta_1' = |m(k+1) - \hat{m}(k+1)'| \qquad \Delta_1'' = |m(k+1) - \hat{m}(k+1)''| \qquad \text{D.1-4}$$

$$\Delta_2' = |m(k+2) - \hat{m}(k+2)'| \qquad \Delta_2'' = |m(k+2) - \hat{m}(k+2)''|$$

$$\Delta_1 = \Delta_1' - \Delta_1'' \qquad \Delta_2 = \Delta_2' - \Delta_2'' \qquad \text{D.1-5}$$

$$e(k) = \begin{cases} 1 & \Delta_1 + \Delta_2 \geq 0 \\ -1 & \Delta_1 + \Delta_2 < 0 \end{cases} \qquad \text{D.1-6}$$

$R_o = s(k)$, $R_1 = e(k-1)$, $R_2 = m(k)$, $R_3 = m(k+1)$, $R_4 = e(k)$, $R_6 = |S(k)|$

Address	Mnemonic	Machine Code	Description
Start	ILR R_o	0001 0000 1010	
	SDR T	0100 1100 0010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	
	SDR R_o	0100 0000 0010	$\|s(k)\| \rightarrow R_o$
	SDR R_6	0100 0110 0010	$\|s(k)\| \rightarrow R_6$, Ac
	CLR T	1001 1100 1010	
	INR T	0111 1100 0010	
	INR T	0111 1100 0010	
	CIA Ac	0011 1111 0010	$-\|s(k)\| \rightarrow$ Ac
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	Jmp Less		$\|s(k)\| < 2^{o}$,
	ILR R_2	0001 0010 1010	$\|s(k)\| >$, $\hat{m}(k) \rightarrow$ Ac
	SDR T	0100 1100 0010	$\hat{m}(k) \rightarrow$ T
	ILR R_o	0001 0000 1010	$\|s(k)\| \rightarrow$ Ac
	ADD T	0000 1100 1010	$\|s(k)\| + \hat{m}(k) \rightarrow$ T, Ac
	ILR R_1	0001 0001 1010	$e(k-1) \rightarrow$ Ac
	ADD T	0000 1100 1010	$\hat{m}(k+1)' \rightarrow$ T, Ac
	CIA T	0011 1110 0010	$-\hat{m}(k+1)' \rightarrow$ T

Address	Mnemonic	Machine Code	Description
	ILR R_3	0001 0011 1010	$m(k+1) \rightarrow Ac$
	ADD T	0000 1100 1010	$m(k+1)-\hat{m}(k+1)' = \Delta'_1 \rightarrow T$, Ac
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	$\Delta'_1 \rightarrow T$
	ILR T	0001 1100 1010	$\Delta'_1 \rightarrow Ac, T$
	SDR R_4	0100 0100 0010	$\Delta'_1 \rightarrow R_4$
	ILR R_2	0001 0010 1010	$\hat{m}(k) \rightarrow Ac$
	SDR T	0100 1100 0010	$\hat{m}(k) \rightarrow T$
	ILR R_0	0001 0000 1010	$\lvert s(k) \rvert \rightarrow Ac$
	CIA Ac	0011 1111 0010	
	ADD T	0000 1100 1010	
	ILR R_1	0001 0001 1010	
	ADD T	0000 1100 1010	
	ILR R_3	0001 0011 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	$-\Delta''_1 \rightarrow Ac$
	ADD R_4	0000 0100 1010	$\Delta_1 = \Delta'_1 - \Delta''_1 \rightarrow R_4$
	ILR R_2	0001 0010 1010	
	INR Ac	0111 1101 0010	

Address	Mnemonic	Machine Code	Description
	AIA Ac	0110 1111 0010	
	SDR T	0100 1100 0010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	$\Delta_2' \rightarrow$ Ac
	ADD R_4	0000 0100 1010	$\Delta_1+\Delta_2' \rightarrow R_4$
	ILR R_2	0001 0010 1010	
	CSR T	0101 1100 1010	
	ADD T	0000 1100 1010	
	AIA Ac	0110 1111 0010	
	SDR T	0100 1100 0010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	$-\Delta_2'' \rightarrow$ Ac
	ADD R4	0000 0100 1010	$\Delta_1+\Delta_2 \rightarrow R_4$
	Jmp cont		
Less	ILR R_0	0001 0000 1010	
	SDR T	0100 1100 0010	
	ILR R_1	0001 0001 1010	
	SPA	0000 1101 1110	
	CLR T	1001 1100 1010	
	ILR R_2	0001 0010 1010	

Address	Mnemonic	Machine Code	Description
	ADD T	0000 1100 1010	
	INR T	0111 1100 0010	
	ILR R_3	0001 0011 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	$\Delta_1' \rightarrow$ Ac
	SDR R_4	0100 0100 0010	$\Delta_1' \rightarrow R_4$
	ILR R_0	0001 0000 1010	
	SDR T	0100 1100 0010	
	CIA T	0011 1110 0010	
	ILR R_1	0001 0001 1010	
	CIA Ac	0011 1111 0010	
	SPA	0000 1101 1110	
	CLR T	1001 1100 1010	
	ILR R_2	0001 0010 1010	
	ADD T	0000 1100 1010	
	CSR T	0101 1100 1010	
	ADD T	0000 1100 1010	
	ILR R_3	0001 0011 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	

Address	Mnemonic	Machine Code	Description
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	$-\Delta_1'' \rightarrow Ac$
	ADD R_4	0000 0100 1010	$\Delta_1 \rightarrow R_4$
	ILR R_2	0001 0010 1010	
	SDR T	0100 1100 0010	
	ILR R_1	0001 0001 1010	
	CIA Ac	0011 1111 0010	
	SPA	0000 1101 1110	
	INR T	0111 1100 0010	
	ILR T	0001 1100 1010	
	AIA Ac	0110 1111 0010	
	SDR T	0100 1100 0010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	$\Delta_2' \rightarrow Ac$
	ADD R_4	0000 0100 1010	$\Delta_1 + \Delta_2' \rightarrow R_4$
	CLR T	1001 1100 1010	
	ILR R_1	0001 0001 1010	
	SPA	0000 1101 1110	
	INR T	0111 1100 0010	
	CIA T	0011 1110 0010	

Address	Mnemonic	Machine Code	Description
	ILR R2	0001 0010 1010	
	ADD T	0000 1100 1010	
	ILR T	0001 1100 1010	
	AIA AC	0110 1111 0010	
	SDR T	0100 1100 0010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	CIA T	0011 1110 0010	$-\Delta_2'' \rightarrow T$
	ILR T	0001 1100 1010	$-\Delta_2'' \rightarrow AC$
	ADD R4	0000 0100 1010	$\Delta_1 + \Delta_2 \rightarrow R4$
cont	ILR R4	0001 0100 1010	
	CSR T	0101 1100 1010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	
	SDR R4	0100 0100 0010	$e(k) \rightarrow R4$
	RRA AC	0001 1111 1010	
	CLR T	1001 1100 1010	
	INR T	0111 1100 0010	
	ANR T	1000 1100 1010	
	ILR R5	0001 0101 1010	
	ADD AC	0000 1101 1010	
	ADD T	0000 1100 1010	

Address	Mnemonic	Machine Code			Description
	SDR R5	0100	0101	0010	
	ILR R_o	0001	0000	1010	
	SDR T	0100	1100	0010	
	ILR R4	0001	0100	1010	
	SPA	0000	1101	1110	
	CIA T	0011	1110	0010	
	ILR T	0001	1100	1010	$\|s(k)\|e(k) \rightarrow Ac$
	ADD R1	0000	0001	1010	
	SDR R0	0100	0000	0010	$s(k+1) \rightarrow R0$
	SDR T	0100	1100	0010	
	SPA	0000	1101	1110	
	CIA T	0011	1110	0010	
	CSR AC	0101	1101	1010	
	ADD T	0000	1100	1010	
	SNA	0000	1101	1110	
	JMP Large				$\|s(k+1)\| \geq s_0$
	CSR T	0101	1100	1010	
	ILR R4	0001	0100	1010	
	SDR R1	0100	0001	0010	delay e(k) as $e(k-1) \rightarrow R1$
	SPA	0000	1101	1110	
	CIA T	0011	1110	0010	
	CIA T	0011	1110	0010	
	ILR T	0001	1100	1010	

Address	Mnemonic	Machine Code	Description
	SDR R0	0100 0000 0010	$s_0 e(k) \rightarrow R_0$
Large	ILR R6	0001 0110 1010	
	SDR T	0100 1100 0010	
	CLR AC	1001 1101 1010	
	INR AC	0111 1101 0010	
	INR AC	0111 1101 0010	
	CIA AC	0011 1111 0010	
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	JMP NOSS		
	ILR R5	0001 0101 1010	$\|s(k)\| = s_0$
	RRA AC	0001 1111 1010	
	SDR T	0100 1100 0010	
	SNA	0000 1101 1110	
	JMP ZEK		$e(k) = -1$
	RRA T	0001 1110 1010	$e(k) = 1$
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP NOSS		
	RRA T	0001 1110 1010	
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP NP1		

Address	Mnemonic	Machine Code	Description
	JMP NOSS		
same	RRA T	0001 1110 1010	
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP NOSS		
	RRA T	0001 1110 1010	
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP NP1		
	JMP NOSS		
NP1	RRA T	0001 1110 1010	
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP NOSS		
	RRA T	0001 1110 1010	
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP NP2		
	JMP NOSS		
NP2	RRA T	0001 1110 1010	
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP NOSS		

Address	Mnemonic	Machine Code	Description
	RRA T	0001 1110 1010	
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP NP3		
	JMP NOSS		
NP3	RRA T	0001 1110 1010	
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP NOSS		
	JMP SS		
ZEK	RRA T	0001 1110 1010	
	ILR T	0001 1100 1010	
	SNA	0000 1101 1110	
	JMP same		
	JMP NOSS		
SS	CLR AC	1001 1101 1010	
	INR AC	0111 1101 0010	
	ADC AC	0000 1101 0010	
	ADD AC	0000 1101 1010	
	INR AC	0111 1101 0010	
	INR AC	0111 1101 0010	
	INR AC	0111 1101 0010	
	INR AC	0111 1101 0010	

Address	Mnemonic	Machine Code	Description
	SDR T	0100 1100 0010	
	ILR R_4	0001 0100 1010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	
	SDR R_0	0100 0000 0010	
NOSS	ILR R_0	0001 0000 1010	
	ADD R_2	0000 0010 1010	
	CMA Ac	1111 1111 1010	
	Out D/A	1101 0000 1000	
	LDI Ac	0100 1111 0010	
	CMA Ac	1111 1111 1010	
	SDR R_3	0100 0011 0010	
	ILR R_4	0001 0100 1010	
	SDR R_1	0100 0001 0010	
	Jmp WAIT	1111 1111 1011	

Appendix E

CVSD Xmitter Algorithm

(R1) = s(k) , (R2) = $\hat{m}(k)$, (R3) = e(k) , (R4) = e(k-1), (R5) = e(k-2)

$\alpha = 0.95$ $S_i = 200$ mV $s_o = 10$ mV $S_i(1-\alpha) = 10$ mV

Address	Mnemonic	Machine Code	Description
Start	ILR R_2	0001 0010 1010	$\hat{m}(k) \rightarrow$ Ac
	AIA Ac	0110 1111 0010	$\hat{m}(k)-m(k) \rightarrow$ Ac
	CSR T	0101 1100 1010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	CIA T	0011 1110 0010	e(k)→ T
	ILR T	0001 1100 1010	
	SDR R_3	0100 0011 0010	e(k)→ R3
	ILR R_1	0001 0001 1010	s(k)→ Ac
	SDR T	0100 1100 0010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	
	ILR T	0001 1100 1010	
	SDR R1	0100 0001 0010	|s(k)|→ R1
	CLR Ac	1001 1101 1010	
	INR Ac	0111 1101 0010	
	ADC Ac	0000 1101 0010	
	ADC Ac	0000 1101 0010	
	ADC Ac	0000 1101 0010	

Address	Mnemonic	Machine Code	Description
	ADC Ac	0000 1101 0010	
	ADC Ac	0000 1101 0010	
	SDR T	0100 1100 0010	0000111111→ T
	ILR R1	0001 0001 1010	$\|s(k)\|$→ Ac
	RRA Ac	0001 1111 1010	
	RRA Ac	0001 1111 1010	
	RRA Ac	0001 1111 1010	
	RRA Ac	0001 1111 1010	
	ANR T	1000 1100 1010	$\frac{1}{16}\|s(k)\|$→ T
	CIA T	0011 1110 0010	
	ILR R1	0001 0001 1010	
	ADD T	0000 1100 1010	$\|s(k)\|-\frac{1}{16}\|s(k)\|=\alpha\|s(k)\|$→ T,Ac
	SDR R1	0100 0001 0010	$\alpha\|s(k)\|$→ R1
	ILR R3	0001 0011 1010	
	SNA	0000 1101 1110	
	Jmp POS1		$e(k)\geq o$
	ILR R4	0001 0100 1010	$e(k)\leq o$
	SNA	0000 1101 1110	
	Jmp Disch		$e(k)<o$, $e(k-1)\geq o$
	ILR R5	0001 0101 1010	$e(k)<o$, $e(k-1)<o$
	SNA	0000 1101 1110	
	Jmp Disch		$e(k)<o, e(k-1)<o, e(k+2)>o$
	Jmp Charge		$e(k)<o, e(k-1)<o$, $e(k-2)<o$

Address	Mnemonic	Machine Code	Description
POS1	ILR R_4	0001 0100 1010	$e(k) \geq o$
	SNA	0000 1101 1110	
	Jmp Pos2		$e(k) \geq o$ $e(k-1) \geq o$
	Jmp Disch		$e(k) \geq o$ $e(k-1) < o$
Pos2	ILR R_5	0001 0101 1010	$e(k) \geq o$ $e(k-1) \geq o$
	SNA	0000 1101 1110	
	Jmp change		$e(k) \geq o, e(k-1) \geq o, e(k-2) \geq o$
Disch.	CLR Ac	1001 1101 1010	
	INR Ac	0111 1101 0010	
	SDR T	0100 1100 0010	$So \rightarrow T$
	ILR R_1	0001 0001 1010	$\alpha \lvert S(k) \rvert \rightarrow Ac$
	CIA T	0011 1110 0010	$-So \rightarrow T$
	ADD T	0000 1100 1010	$\alpha \lvert s(k) \rvert - So \rightarrow T$
	SNA Ac	0000 1101 1110	
	Jmp large		$\alpha \lvert s(k) \rvert \geq S_o$
	CLR Ac	1001 1101 1010	$\alpha \lvert s(k) \rvert < S_o$
	INR Ac	0111 1101 0010	
	SDR R_1	0100 0001 0010	$So \rightarrow R_1$
Large	ILR R_1	0001 0001 1010	
	SDR T	0100 1100 0010	
	ILR R_3	0001 0011 1010	
	SPA	0000 1101 1110	
	CIA T	0011 1110 0010	

Address	Mnemonic	Machine Code	Description
	lLR T	0001 1100 1010	
	SDR R1	0100 0001 0010	s(k+1)→ R1
	Jmp Estimate		
charge	lNR R1	0111 0001 0010	$s(k+1)=\alpha\|s(k)\|+so$
	Jmp Large		
Estimate	SNA	0000 1101 1110	
	Jmp Psk1		$s(k+1)\geq o$
	lLR R2	0001 0010 1010	$s(k+1)<o$
	SNA	0000 1101 1110	
	Jmp PMk		$s(k+1)<o$, $\hat{m}(k)\geq o$
	lLR R1	0001 0001 1010	$s(k+1)<o$, $\hat{m}(k)<o$
	ADR R2	0000 0010 1010	
	SNA	0000 1101 1110	
	Jmp Negsat		$s(k+1)<o$ $\hat{m}(k)<o$ $\hat{m}(k+1)\geq o$
	Jmp Out		
negsat	CLR Ac	1001 1101 1010	
	lNR Ac	0111 1101 0010	
	RRA Ac	0001 1111 1010	
	SDR R2	0100 0010 0010	$\hat{m}(k+1)=1000000000\rightarrow R_2$
	Jmp Out		
PMk	lLR R1	0001 0001 1010	$s(k+1)<o$ $\hat{m}(k)\geq o$
	ADR R2	0000 0010 1010	$m(k+1) \rightarrow R_2$
	Jmp Out		

Address	Mnemonic	Machine code	Description
Psk1	ILR R2	0001 0010 1010	$s(k+1)\geq 0$
	SNA	0000 1101 1110	
	Jmp posmk		$s(k+1)\geq o \quad \hat{m}(k)\geq o$
	ILR R1	0001 0001 1010	$s(k+1)\geq o \quad \hat{m}(k)<o$
	ADR R2	0000 0010 1010	$\hat{m}(k+1)\rightarrow R_2$
	Jmp out		
posmk	ILR R1	0001 0001 1010	$s(k+1)\geq o \quad \hat{m}(k)\geq o$
	ADR R2	0000 0010 1010	$\hat{m}(k+1)\rightarrow R_2$
	SNA	0000 1101 1110	
	Jmp out		$\hat{m}(k+1)\geq o$
	CLR Ac	1001 1101 1010	$\hat{m}(k+1)<o$
	INR Ac	0111 1101 0010	
	RRA Ac	0001 1111 1010	
	CMA Ac	1111 1111 1010	
	SDR R2	0100 0010 0010	$0111111111=\hat{m}(k+1)\rightarrow R_2$
out	ILR R2	0001 0010 1010	
	CMA Ac	1111 1111 1010	
	OUT D/A	1101 0000 1000	
	ILR R4	0001 0100 1010	
	SDR R5	0100 0101 0010	delay e(k) as e(k-1)
	ILR R3	0001 0011 1010	
	SDR R4	0100 0100 0010	delay e(k-1) as e(k-2)
	WAIT	1111 1111 1011	Wait for new sample

Appendix F

Switching Receiver Design

Address	Mnemonic	Machine Code	Description
INITIAL	CLR R_0	1001 0000 1010	
	CLR R_1	1001 0001 1010	
	CLR AC	1001 1101 1010	
	INR AC	0111 1101 0010	
	SDR T	0100 1100 0010	
	ADD AC	0000 1101 0010	
	ADD AC	0000 1101 1010	
	ADD AC	0000 1101 1010	
	ADD AC	0000 1101 0010	
	ADD AC	0000 1101 0010	
	ADD AC	0000 1101 1010	
	ADD AC	0000 1101 1010	
	SDR R_7	0100 0111 0010	(R_7)=0011001100
	ADD AC	0000 1101 0010	
	ADD AC	0000 1101 0010	
	SDR R_8	0100 1000 0010	(R_8)=1100110011
	ILR T	0001 1100 1010	
	ADD AC	0000 1101 1010	
	ADD AC	0000 1101 0010	
	ADD AC	0000 1101 1010	
	ADD AC	0000 1101 0010	
	ADD AC	0000 1101 1010	
	ADD AC	0000 1101 0010	
	ADD AC	0000 1101 1010	
	ADD AC	0000 1101 0010	
	SDR R_9	0100 1001 0010	(R_9)=0101010101

Address	Mnemonic	Machine Code	Description
START	ILR R_1	0001 0001 1010	
	SDR T	0100 1100 0010	
	RRA T	0001 1110 1010	
	CLR AC	1001 1101 1010	
	INR AC	0111 1101 0010	
	RRA AC	0001 1111 1010	
	CMA AC	1111 1111 1010	
	ANR T	1000 1100 1010	
	ILR T	0001 1100 1010	
	SDR R_1	0100 0001 0010	MSB of R_1 is ready to accept LSB of R2
	ILR R_2	0001 0010 1010	
	RRA AC	0001 1111 1010	
	SDR R_2	0100 0010 0010	
	CLR T	1001 1100 1010	
	INR T	0111 1100 0010	
	RRA T	0001 1110 1010	
	ANR T	1000 1100 1010	
	ILR T	0001 1100 1010	
	ORR R_1	1100 0001 1010	
	CLR AC	1001 1101 1010	
	INR AC	0111 1101 0010	
	RRA AC	0001 1111 1010	
	CMA AC	1111 1111 1010	
	ANR R_2	1000 0010 1010	
	LTM AC	0101 1011 1010	
	ANR R_2	1000 0010 1010	
	ILR R_0	0001 0000 1010	
	RRA AC	0001 1111 1010	
	SNA	0000 1101 1110	

Address	Mnemonic	Machine Code	Description
	JMP NO		LSB R_0=0
	ILR R_0	0001 0000 1010	LSB R_0=1
	SNA	0000 1101 1110	
	JMP SVADM		MSB R_0=0
	JMP CVSD		MSB R_0=1
NO	ILR R_1	0001 0001 1010	
	SDR T	0100 1100 0010	
	ILR R_9	0001 1001 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	JMP AGAIN1		
	JMP NOZ1		$(R_1)\neq(R_9)$
AGAIN1	ILR R_9	0001 1001 1010	
	SDR T	0100 1100 0010	
	ILR R_1	0001 0001 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	JMP ZER1		$(R_1)=(R_9)$
NOZ1	ILR R_1	0001 0001 1010	$(R_1)\neq(R_9)$
	SDR T	0100 1100 0010	
	ILR R_8	0001 1000 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	JMP AGAIN2		

Address	Mnemonic	Machine Code	Description
	JMP WAIT	1111 1111 1011	$(R_1) \neq (R_8)$
AGAIN2	ILR R_8	0001 1000 1010	
	SDR T	0100 1100 0010	
	ILR R_1	0001 0001 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	JMP ZER2		$(R_1)=(R_8)$
	JMP WAIT	1111 1111 1011	$(R_1) \neq (R_8)$
ZER2	ILR R_2	0001 0010 1010	
	SDR T	0100 1100 0010	
	ILR R_7	0001 0111 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	JMP AGAIN3		
	JMP WAIT	1111 1111 1011	$(R_2) \neq (R_7)$
AGAIN3	ILR R_7	0001 0111 1010	
	SDR T	0100 1100 0010	
	ILR R_2	0001 0010 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	JMP YES1		$(R_2)=(R_7)$
	JMP WAIT	1111 1111 1011	$(R_2) \neq (R_7)$
YES1	CLR AC	1001 1101 1010	
	INR AC	0111 1101 0010	

Address	Mnemonic	Machine Code	Description
	ORR R_0	1100 0000 1010	LSB R_0=1
	ANR R_0	1000 0000 1010	MSB R_0=0
	JMP SVADM		
ZER1	ILR R_2	0001 0010 1010	
	SDR T	0100 1100 0010	
	ILR R_9	0001 1001 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	JMP AGAIN4		
	JMP WAIT	1111 1111 1011	$(R_2)\neq(R_9)$
AGAIN4	ILR R_9	0001 1001 1010	
	SDR T	0100 1100 0010	
	ILR R_2	0001 0010 1010	
	CIA T	0011 1110 0010	
	ADD T	0000 1100 1010	
	SNA	0000 1101 1110	
	JMP YES2		$(R_2)=(R_9)$
	JMP WAIT	1111 1111 1011	$(R_2)\neq(R_9)$
YES2	CLR AC	1001 1101 1010	
	INR AC	0111 1101 0010	
	ORR R_0	1100 0000 1010	LSB R_0=1
	RRA AC	0001 1111 1010	
	ORR R_0	1100 0000 1010	MSB R_0=1
	JMP CVSD		

APPENDIX G

MARKOV SEQUENCES AND CHAINS

1. Markov Sequences

Definitions:

A sequence $\underline{X}_n$ of random variables is called Markov-1 (first order Markov) if for any n we have:

$$F(x_n|x_{n-1}x_{n-2}\cdots x_1) = F(x_n|x_{n-1}) \quad , \quad x_n \text{ discrete} \qquad \text{(G-1a)}$$

$$f(x_n|x_{n-1}x_{n-2}\cdots x_1) = f(x_n|x_{n-1}) \quad , \quad x_n \text{ continuous} \qquad \text{(G-1b)}$$

Therefore the joint distribution of $\underline{X}_n$ is determined by $f(x_1)$ and the transitional densities:

$$f(x_1,x_2\cdots x_n) = f(x_n|x_{n-1})f(x_{n-1}|x_{n-2}) \cdots f(x_2|x_1)\ f(x_1) \qquad \text{(G-2)}$$

Properties

1. A Markov sequence is Markov also in the reverse direction:

$$f(x_1|x_2x_3\cdots x_n) = f(x_1|x_2) \qquad \text{(G-3)}$$

Proof:

$$f(x_1|x_2x_3\cdots x_n) = \frac{f(x_1x_2x_3\cdots x_n)}{f(x_2x_3\cdots x_n)}$$

$$= \frac{f(x_n|x_{n-1})f(x_{n-1}|x_{n-2})\cdots f(x_2|x_1)f(x_1)}{f(x_n|x_{n-1})f(x_{n-1}|x_{n-2})\cdots f(x_3|x_2)f(x_2)}$$

$$= \frac{f(x_2x_1)}{f(x_2)} = \frac{f(x_1|x_2)f(x_2)}{f(x_2)} = f(x_1|x_2)$$

2. From (G-1) we conclude that:

$$E\{x_n | x_{n-1} \cdots x_1\} = E\{x_n | x_{n-1}\} \tag{G-4}$$

3. In a Markov sequence one might say that if the present is known, then the past is independent of the future. If $n > r > s$, then:

$$f(x_n x_s | x_r) = f(x_n | x_r)\ f(x_s | x_r) \tag{G-5}$$

4. A Markov sequence is called homogeneous if the conditional density $f(x_n | x_{n-1})$ is indepenent of n.

$$f_{x_n}(\alpha | x_{n-1} = \beta) = \text{function}(\alpha, \beta) \tag{G-6}$$

5. A Markov sequence is stationary if it is homogeneous and all of the R. V. X_n have the same density

$$f_{x_1}(\alpha) = f_{x_2}(\alpha) = \ldots = f_{x_n}(\alpha) \tag{G-7}$$

6. The transitional densities of a Markov sequence satisfy

$$f(x_n | x_s) = \int_{-\infty}^{\infty} f(x_n | x_r)\ f(x_r | x_s)\ dx_r\ , \quad n > r > s \tag{G-8}$$

Proof: For any three R. V. X_n , X_r , X_s we have

$$f(x_n | x_s) = \int_{-\infty}^{\infty} f(x_n x_r | x_s)\ dx_r = \int_{-\infty}^{\infty} f(x_n | x_r x_s) f(x_r | x_s) dx_r$$

Since the sequence is Markov-1, we have:

$$f(x_n | x_s) = \int_{-\infty}^{\infty} f(x_n | x_r)\ f(x_r | x_s)\ dx_r$$

Equation (G-8) is known as the Chapman-Kolmogoroff equation.

2. Markov Chains.

If the R.V. are of discrete type taking on the values $a_1, a_2, \ldots, a_n$, and

$$Pr[X_n=a_n \mid X_{n-1}=a_{n-1}, \ldots, X_1=a_1] = Pr[X_n=a_n \mid X_{n-1}=a_{n-1}] \tag{G-9}$$

then the sequence $\underline{X}_n$ is called a Markov-1 chain.

Let

$$p_i(n) = Pr[X_n = a_i] \quad , \text{ unconditional density} \tag{G-10}$$

$$P_{ji}(s,n) = Pr[X_n=a_i \mid X_s=a_j], \text{ conditional density } n>s \tag{G-11}$$

note that small p means unconditional density and capital P means conditional density. It is obvious that:

$$p_i(n) = \sum_j P_{ji}(s,n)\, p_j(s) \tag{G-12}$$

Furthermore:

$$\sum_i p_i(n) = 1 \ , \qquad \sum_i P_{ji}(s,n) = 1 \tag{G-13}$$

For a Markov chain we also have

$$P_{ji}(s,n) = \sum_k P_{ki}(r,n)\, P_{jk}(s,r) \ , \quad n > r > s \tag{G-14}$$

which is the discrete version of the Chapman-Kolmogoroff equation.

With $\underline{P}(s,n)$ the square matrix with elements $P_{ji}(s,n)$ and $\underline{p}(n)$ the column matrix with elements $p_i(n)$ we have from (G-12) and (G-14),

$$\underline{p}(n) = \underline{p}(s)\ \underline{\underline{P}}(s,n) \tag{G-15a}$$

$$\underline{\underline{P}}(s,n) = \underline{\underline{P}}(s,r)\ \underline{\underline{P}}(r,n) \tag{G-15b}$$

If the conditional probability $P_{ji}(s,n)$ depends only on the difference n-s, the chain is called homogeneous. For a homogeneous chain

$$\underline{\underline{P}}(k) = \underline{\underline{P}}(1)^k \tag{G-16}$$

where $\underline{\underline{P}}(1)$ is the one time transition matrix. If, in a homogeneous matrix, the distributions of the first two R.V. are the same, i.e. if

$$\underline{p}(2) = \underline{p}(1) \tag{G-17}$$

then

$$\underline{p}(3) = \underline{p}(2)\underline{\underline{P}}(1) = \underline{p}(1)\underline{\underline{P}}(1) = p(2) \tag{G-18}$$

and by repeated applications

$$\underline{p}(n) = \underline{p}(1) \tag{G-19}$$

We say that the chain is stationary in the narrow or strict sense. If $\underline{p}(1) \neq \underline{p}(2) \neq \underline{p}(3)$ then the chain is not stationary in the narrow sense. However, if $\underline{\underline{P}}(1)^n$ tends to a matrix with identical row elements, and the limit:

$$\lim_{n\to\infty} p_i(n) = p_i \tag{G-20}$$

exists, then we say that the chain is stationary in the wide sense, or asymptotically stationary.

Appendix H

Proof of Transformation

Let $\underline{\underline{A}}$ be the following 8 × 8 matrix:

$$\underline{\underline{A}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Then $\underline{\underline{P}}_T(20) = \underline{\underline{A}}\ \underline{\underline{P}}(20)\ \underline{\underline{A}}^T$, where $\underline{\underline{A}}^T$ is the transpose of $\underline{\underline{A}}$.

APPENDIX I

Using the first part of Eq. (3.5-5)

$$A(k,i) = \Pr[\hat{m}(k-1)=x_{i-1}(k-1),\ m(k-1) \leq x_{i-1}(k-1)] \tag{I-1}$$

$$= \Pr[\hat{m}(k-1)=x_{i-1}(k-1),\ \hat{m}(k) = x_i(k)] \tag{I-2}$$

Using Baye's theorem,

$$A(k,i) = \Pr[\hat{m}(k)=x_i(k) \,|\, \hat{m}(k-1)=x_{i-1}(k-1)] \cdot \Pr[\hat{m}(k-1)=x_{i-1}(k-1)] \tag{I-3}$$

since

$$\Pr[\hat{m}(k-1) = x_{i-1}(k-1)] = A(k-1,i-1) + B(k-1,i-1) \tag{I-4}$$

The transition probability from level x_{i-1} at time k-1 to level x_i at time k is:

$$\Pr[\hat{m}(k)=x_i(k) \,|\, \hat{m}(k-1)=x_{i-1}(k-1)] = \frac{A(k,i)}{A(k-1,i-1) + B(k-1,i-1)} \tag{I-5}$$

References

1. De Jager, F., 'Delta Modulation - a new method of PCM transmission using the 1 unit code.' Philips Research Report, 7, 442-466, December (1952).

2. Van De Weg, N., 'Quantizing noise of a single integration delta modulation system with an N-digit code.' Philips Research Report, 367-385, October 8 (1953).

3. Zetterberg, L.H., 'A comparison between delta modulation and pulse code modulation', Ericsson Technics, I, 45-154 (1955).

4. Greefkes, J.A. and de Jager, F., 'Continuous delta modulation, Philips Research Report, No. 23, 233-246 (1968).

5. Greefkes, J.A. and Riemens, K., "Code modulation with digitally controlled companding for speech transmission', Philips Technical Review, 31, No.11/12, 335-353, (1970).

6. Brolin, S.J. and Brown, J.H., 'Companded delta modulation for telephony', I.E.E.E. Trans.Com.Tech., COM-16, 157-162, (1968)

7. Abate, J.E., 'Linear and Adaptive delta modulation', Proc. I.E.E.E., 55, No.3, 298-308, March (1967).

8. Jayant, N.S., Adaptive delta modulation with a one-bit memory', Bell Systems Tech.J., 49, No.3, 321-342, March (1970).

9. Song, C.L., Garodnick, J. and Schilling, D.L., 'A variable step size robust delta modulator', I.E.E.E. Trans.Com.Tech. COM-19, No.6, 1033-1044, December (1971).

10. Schienberg, N., Tsu, Luen R. lei, Schilling, D.L., 'Adaptive Delta Modulation Systems for Video Encoding', I.E.E.E. Trans.Com.Tech. COM-25, No.11, 1302-1313, November(1977).

11. Taub and Schilling, 'Principle of Communication Systems', McGraw-Hill Book Co., N.Y., 402-406, (1971)

12. Fine, T.L., 'The response of a particular nonlinear system with feedback to each of two random processes,' I.E.E.E. Trans. on Info. Theory, IT-14, No.2, 255-264, March (1968).

13. Slepian, D., 'On delta modulation,' B.S.T.J., 51, No.10, 2101-2137, December (1972).

14. Mills, T.K. and O'Neal, J.B.,Jr., 'Quantizing noise calculations in delta modulation,' I.C.C., Montreal, No.1,(1971).

15. O'Neal, J.G.,Jr., 'Delta modulation quantizing noise and analytical and computer simulation results for Gaussian and television input signals,' B.S.T.J., 45, No.1, 117-141, January (1966).

16. Zetterberg, L.H. and Uddenfeldt, J., 'Adaptive delta modulation with delayed decision,' I.E.E.E. Trans. on Com., COM-22, No.9, September (1974).

17. Uddenfeldt, J. and Zettenberg, L.H., 'Algorithms of delayed encoding in delta modulation with speech-like signals,' I.E.E.E. Trans. on Com., June (1976).

18. Cutler, C. Chapin, 'Delayed encoding:Stabilizer for adaptive coders,' I.E.E.E. Trans. on Com., COM-19, No.6, December (1971).

19. Hans, R. Schindler, 'Linear, Nonlinear, and Adaptive delta modulation,' I.E.E.E. Trans. on com., COM-19, No.6, December (1971).

20. Papoulis, A., 'Probability, Random Variables, and Stochastic Processes,' McGraw Hill Book Co., N.Y., (1965).

21. Steele, R., 'Delta Modulation Systems,' John Wiley & Sons Inc., N.Y., (1975).

22. Wozencraft, J.M., Jacobs, I.M., 'Principles of Communication Engineering,' John Wiley & Sons, Inc., N.Y., (1965).

23. Rosenblatt, M., 'Markov Processes. Structure and Asymptotic Behaviour,' Springer-Verlag, N.Y., Heidelberg, Berlin, (1971).

24. Intel 'Schottky Bipolar LSI Microcomputer set,' Data Sheets.

25. AMD 'Am2900 Bipolar Microprocessor Family,' Data Sheets.

26. Motorola 'High Performance MECL LSI Processor Family,', Data Sheets.