MCR-70-34 Copy No.

Potential Applications of Digital Techniques to Apollo Unified S-Band Communications System

(REVISION 1)

FINAL REPORT

FEBRUARY 1970

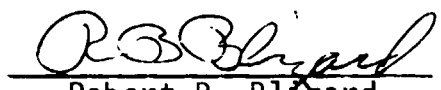
MARTIN MARIETTA CORPORATION

MCR-70-34

POTENTIAL APPLICATIONS OF DIGITAL TECHNIQUES TO APOLLO UNIFIED S-BAND COMMUNICATIONS SYSTEM FINAL REPORT

February 1970

Approved



Robert B. Blizard
Program Manager

MARTIN MARIETTA CORPORATION
DENVER DIVISION
Denver, Colorado 80201

FOREWORD

This report was prepared by the Martin Marietta Corporation in accordance with Contract NAS9-9852, "Study of Potential Applications of Digital Techniques to the Apollo Unified S-Band Communications System" for the Manned Spacecraft Center. This is the final report on all work performed on this contract. This contract spanned the period from 1 July 1969 to 23 February 1970. This report is submitted in accordance with Contract Schedule Article 7.

The NASA-MSC Technical Monitor was Mr. Bartus H. Batson whose direction and encouragement is hereby acknowledged. The Martin Marietta personnel contributing to this program were:

Dr. Robert B. Blizard	Program Manager
John D. Pettus	Assistant Program Manager
Ward B. Anthony	Software Studies
James L. McKinney	Software Studies
David L. Manion	Hardware Design
Leslie A. Pownall	Image Processing
William A. Stevens	Compression Studies
Jack L. Munger	Programer
Louis G. Pinelli	Programer
Glen A. Metgen	Programer
Dr. Harvey M. Gates	Coding Studies
Michael R. Turner	Coding Studies
JoAnn P. Armstrong	Programer

CONTENTS

Appendix

Figure

I. INTRODUCTION AND SUMMARY

The objective of this study was to determine the feasibility of digital transmission of real-time, standard format TV, together with voice and all the other data that must be returned from the Apollo spacecraft. The principal benefit offered by a digital system is a reduction in the RF power required to transmit a good picture and reliable data. To achieve this improvement it is necessary to use two techniques, always considered together, made possible by converting to an all-digital system: error-correcting codes and data compression. Data compression makes each transmitted bit more important so that fewer errors can be tolerated, and error correction gives more improvement at low error rates.

For both error correction and data compression we have selected preferred techniques, simulated these techniques, and made preliminary hardware designs. In the course of this work we have also obtained results and created ideas that may find application in communication links that operate at much lower data rates than the Apollo link.

In selecting techniques for error correction and data compression hardware requirements and performance were considered. For error correction, block codes were eliminated from consideration because hardware size limitations require the use of a relatively small constraint length making performance substantially poorer than can be obtained with convolutional codes.

For decoding convolutional codes two methods were considered, sequential decoding and the Viterbi algorithm. Viterbi decoding is quite new, receiving serious consideration as a practical method only after Heller demonstrated at the Jet Propulsion Laboratory (JPL) that excellent performance with reasonable amounts of computation could be obtained with codes of small constraint length.

We reduced the Viterbi algorithm to a paper hardware design and concluded that it is feasible to operate at 7.5×10^6 bits/sec with TTL components. This design was compared with the Codex Corporation's high speed sequential decoder. Performance is about the same in terms of error rate vs SNR for the current error rate application. The curve is steeper for sequential decoding; if error rates above 10^{-5} can be tolerated the Viterbi performance is better, and if rates below 10^{-6} are needed sequential decoding requires less SNR.

In this performance comparison, the high data rate is an important consideration. The Codex machine uses hard decision detection (rather than multi-level quantizing) to simplify the algorithm and permit operation at 5×10^6 bits/sec. Sequential decoders designed for lower speeds perform better by 2dB or more. A parts count shows that the Codex decoder needs about as many flat-packs as the Viterbi design, but it also requires a core memory unnecessary in the Viterbi.

The Viterbi decoder is recommended for this application for two reasons: the parts cost is lower, and the maximum speed is slightly higher. The development cost for Viterbi decoding should be low because the algorithm is simple and well defined. Unlike sequential decoding, it has no parameters that must be chosen by trial to optimize the performance. Another round of hardware development may give the advantage to sequential decoding, but the Codex machine is designed for high speed operation and a complete redesign would be necessary to improve its speed.

For TV compression we recommend the zero order interpolator (ZOI) algorithm, but results do not show much difference between the ZOI and the zero order predictor (ZOP). Both algorithms are fairly simple to implement, and our design studies show that operation at 5×10^6 pixels/sec is feasible (pixel: picture element). The first order interpolator (FOI) gives pictures without contouring, but it is much more difficult to implement because it requires taking the quotient of two numbers. Simulation of these techniques includes dithering the quantization levels from line to line to decrease the component of contouring that results from the finite quantizing intervals.

Results show that a good TV picture, with standard format and frame rate, can be transmitted with 7×10^6 bits/sec or 235,000 bits/frame. Techniques that can be developed during an extension of this study contract should reduce this rate.

For the voice channels, we recommend delta modulation as an effective but simple compression technique. Since the bit rate required for voice is much less than for TV, a large effort was not invested into this part of the study and our recommendation is based on published results. The experience of investigators at the Manned Space Center agree with our conclusions.

The overall performance comparison between an FM system and the digital system with data compression and error correction shows a difference of 3 to 4 db in required RF power. Further refinements in the compression techniques may increase the margin.

A technique for combining data compression and sequential decoding was also considered. This method uses redundancy in the data to help in the decoding process. The theory was worked out prior to this contract under one of our company funded research tasks. Under the contract operation was verified with computer simulation. It was also determined that too much computation was needed for decoding for the TV data rate. However, there may be valid applications where data rates are lower and the transmitting interval is so austere that conventional data compression hardware can't be supported.

Hardware requirements of the digital system were studied in varying depth. The Viterbi decoder received the most attention since, to our knowledge, none had been designed previously. Two paper designs were made. The first will operate at a data rate of at least 7.5×10^6 bits/sec.and requires about 850 dual in-line packages. The second type is for much lower speeds, less than 5×10^4 bits/sec, and uses about 250 packages. Hardware is saved by performing operations in series and by using MSI.

We did not make any hardware studies for sequential decoding because Codex Corporation made available a parts count of their high speed machine.

Also investigated were the core requirements and decoding speeds for several small general purpose computers used as Viterbi decoders. Many space vehicles will have considerable memory and logic capacity on board for data handling and navigation, and little additional equipment would be required to program a Viterbi decoder to improve the margin, speed, and reliability of the command link.

To support the simulation and hardware studies, we wrote a program that helps to find good convolutional codes. It finds all codes that give maximum-minimum weight and tabulates the number of words of each weight.

In the course of studying the Viterbi algorithm, a way was conceived to use it in a repetitive fashion for codes with very large constraint length. Although the theoretical groundwork has been done, it will be necessary to run some simulations to determine the performance. We expect to be able to evaluate this technique, referred to as iterative Viterbi decoding (IVD), during the extension of this contract.

It is expected to outperform sequential decoding, and we hope the idea will promote a useful exchange of ideas.

During this study we have been fortunate in having Dr. David Forney, Director of Research of the Codex Corporation, as a consultant. We have been pleased by his knowledge and his ability to contribute to all aspects of the work.

II. CODING AND DATA COMPRESSION STUDIES

A. CODING STUDIES

The first decision in choosing a coding-decoding method was to reject block orthogonal codes. A convolutional code was chosen for two reasons: performance is better for a given decoding hardware complexity, and the band expansion can be low.

Convolutional codes may be decoded sequentially using a Fano algorithm* or Gallager algorithm,† or with the Viterbi algorithm.§ A sequential decoder incorporates searchback operations, and a core memory unit is required to store past data that would be called back if noise levels increase. However, searchbacks hamper the speed of the decoding operation. The decoder must be restarted if an erasure occurs, which may require several hundred bits of good data; i.e., data with little noise on it. Thus, restarting the decoder requires several constraint lengths, and constraint lengths for a sequential decoder are 4 to 10 times longer than a Viterbi decoder. The information rates of existing decoders of this type do not exceed 5 Mbps.

One parameter common to both systems is the quantization of the signal at the receiver. Each of the received bits is put into an integrate-and-dump circuit that quantizes the information Q levels. The Viterbi decoder operates at higher efficiencies with larger Q. The high-speed Fano decoders can only operate with hard decision, Q = 2. This is caused by the fact that the branch metric for this sytem can have only three values (e.g., 1, -5, -11), which allows very simple modulo-6 arithmetic. If the Fano

---

*R. M. Fano: "A Heuristic Discussion of Probabilistic Decoding." *IEEE Transactions on Information Theory*, Vol IT-9, April 1963, p 64-73.

†R. G. Gallager: *Information Theory and Reliable Communication*. John Wiley and Son, Inc., New York, N. Y., 1968.

§A. J. Viterbi: "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm." *IEEE Transactions on Information Theory*, IT-13, April 1967.

decoder were to operate at more than two levels of quantization, an appropriate branch metric with more entries would have to be implemented, and the system would be much slower. Thus, the high-speed Fano decoder is tied to the parameters R = 1/2 and Q = 2, and offers very little flexibility. The code for this system is not as critical as with the Viterbi system.

The Viterbi algorithm decoder, which also decodes convolutional codes, operates at a fixed rate with no searchbacks; thus it does not require a memory unit and lends itself to small packaging. Since it has no memory to access, it may be made to run faster than the Fano decoder with about the same number of electronic components. This decoder is based on the state of the first k-1 stages of the shift register of the convolutional encoder. There are 2^{k-1} possible states one must examine to decode 1 bit of information. For this reason the constraint length of a Viterbi system normally does not exceed 8 bits. Short constraint lengths imply less performance. In the Fano system the number of allowable searchbacks restricts the performance of the system.

The Viterbi decoder is selected over the Fano decoder because:

1) It is potentially faster;
2) It is smaller and less complex to implement;
3) It requires no restart after a burst of noise;
4) It is competive and outperforms the Fano at error rates greater than 2×10^{-5};
5) It requires no memory unit;
6) It is potentially more flexible; i.e., can easily run at various input quantization levels, Q, can operate at shorter constraint lenghts, and can easily be designed to run at rates 1/2, 1/3, or 1/4.

2. Choosing a Convolutional Code

For Viterbi decoding, a small constraint length is necessary to keep the amount of hardware within reason. Performance strongly depends on the particular code that is chosen, and this section describes criteria that will eliminate bad codes and help in choosing good ones.

The Hamming distance (or simply "distance") between any two code words is the number of places in which they differ. Thus, 100111 and 110101 differ in two places, the 2nd and 5th, and therefore have a distance of 2. If the distance between two code words A and B is small, it is easy for channel noise to make the receiver think that B was transmitted when in fact it was A, and vice versa. In other words, A plus the noise can easily look more like B than like A.

Thus a code with a large minimum distance between code words is desirable. It is a fundamental property of parity check codes, including convolutional codes, that a list of the Hamming distances between a particular code word A and all other words in the code are independent of A. This simplifies the problem of finding the minimum distance since we can examine the distances from the all-zero code word 000...00. The Hamming distance from this word to a second word is simply the mumber of ones in the second word, and this number is called the Hamming weight (or simply the "weight") of the word. It is much easier to test a given code for the minimum weight than for minimum distance.

Heller* gives a method for determining an upper boumd on the minimum weight of a convolutional code for any constraint length k and rate 1/V. For k = 5 and V = 2, the greatest minimum weight is 8. This minimum weight is obtained only with the code generated by the shift register connections 10111, 11101 (and symmetrical transformations of this code). Unfortunately, it turns out that this code is not a good one. Because there are an even number of connections to each mod-2 adder, the coded bit stream is the same for the complement of the message as it is for the message itself. For instance, there is no way to distinguish between the messages ...000000... and ...11111.... Simulation on a general-purpose computer confirms this prediction. When the message is all zeros, the decoder puts out long strings of zeros and ones. The simulated noise flips the decoder from one state to the other.

More general considerations permit a quick test to eliminate degenerate codes like the one just described. It is convenient to treat the connection vectors as "polynominals with coefficients from the field of two elements." These "elements" are, of course, 0 and 1 and they obey the rules 0 + 0 = 0; 0 + 1 = 1; 1 + 1 = 0; 0 . 0 = 0; 0 . 1 = 0; 1 . 1 = 1. In polynominal form, the connection vectors for the code discussed above are:

---

*J. A. Heller: *Short Constraint Length Convolutional Codes.* JPL Space Programs Summary, Vol III, p 37-54.

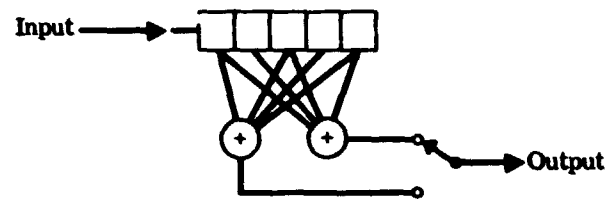
$x^4 + x^2 + x + 1$, and

$x^4 + x^3 + x^2 + 1$.

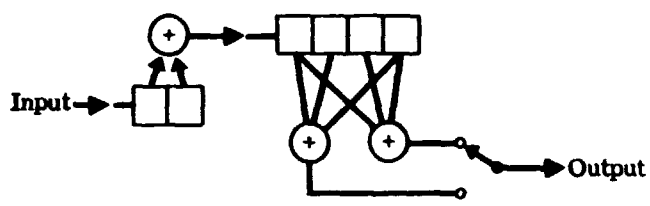
These polynominals can be factored as:

$$(x^4 + x^2 + x + 1) = (x + 1)(x^3 + x^2 + 1),$$

$$(x^4 + x^3 + x^2 + 1) = (x + 1)(x^3 + x + 1).$$



(a) k = 5 Encoder



(b) k = 4 Equivalent

Figure II-1 A Degenerate K = 5 Encoder and Its k = 4 Equivalent

The fact that both polynominals have x + 1 as a factor indicates that the code is degenerate and is equivalent to a k = 4 code. This can be verified by putting messages through the two encoders in Figure II-1. The two-stage shift register that precedes the k = 4 encoder performs a transformation in the message space without affecting the performance of the code.

Since the only k = 5, V = 2 code that gives minimum weight 8 is degenerate, the codes with minimum weight 7 must be considered. There are several of these including the ones listed below and their partners in symmetry.

11101, 11001; 11101, 10011; 11011, 11001; 11011, 10101.

Of these, all except the last perform satisfactorily. The polynominals of this code factor as

$$x^4 + x^3 + x + 1 = (x + 1)(x + 1)(x^2 + x + 1),$$

$$x^4 + x^2 + 1 = (x^2 + x + 1)(x^2 + x + 1),$$

and they have the common factor $x^2 + x + 1$. It is easily verified that with this code the infinite message ... 011011011 ... is encoded as all zeros and is thus indistinguishable from the all-zero message.

For each of the good k = 5, V = 2 codes, the distribution of code word weights for the 256 messages of 9 or fewer bits is given in Table II-1. The best code is probably 11101, 10011 since it has the least number of words of the lowest weights. Simulations run so far do not show a definite difference in performance between the codes, but better information will be available soon.

Table II-1 Numbers of Code Words of Each Weight for Messages of Length 9-Bits or Less

CODE WORD WEIGHTS FOR K = 5, V = 2															
CODE	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
11101 11001	3	3	3	17	22	23	37	36	33	35	23	11	6	3	1
11101 10011	2	3	4	13	20	23	35	44	40	35	24	7	2	3	1
11011 11001	2	4	6	12	20	28	33	40	42	28	16	12	8	4	1

3. Score Control for Viterbi Decoding

In decoding, the score increment ΔS is never negative; and it is necessary to provide some means to prevent overflow of the score registers. Since only the differences between the scores in the various registers are significant, decoding will not be affected if all scores are reduced by the same amount whenever overflow is threatened.

A simple way to prevent overflow is as follows: Whenever all score registers have a "1" for the most significant bit, this bit is reset to zero in all registers. This process subtracts 2^{s-1} from each score, where s is the number of bits in each score register.

It is necessary that no register overflows before all of them reach a count of 2^{s-1}. This condition is fulfilled if the maximum difference S between two scores is always less than $2^{s-1} - \Delta S_m$ where ΔS_m is the largest score increment that is possible. Therefore, it is useful in choosing the score register size to know an upper bound on the difference between any two scores. Such a bound is demonstrated next.

Let the message of lowest score (not necessarily the correct message) be the all-zero message. (Because of the properties of group codes, this assumption does not sacrifice generality.) Because the decoding procedure chooses the lower score at each mode, any state is always reached via the path that yields the lowest score. Therefore, if an upper bound is established for the score if a particular path is assumed, the score cannot be larger for the true path.

For any path the maximum score difference is equal to the Hamming weight W of the path times the maximum score change for a single channel bit or $\Delta S_m/V$ where V is the reciprocal of the rate.

From any state to any other state there is a unique path of k-1 steps and there are two paths of k steps. The messages corresponding to the two paths of k steps differ only in a single bit. Therefore, the Hamming distance between the code words is equal to W_o, the total number of connections between the mod-2 adders and the shift register.

Consider now the two paths of k steps leading from the all-zero state to a given state. The code words will have kV bits of which W_o are different and $kV-W_o$ are alike. We are interested in the maximum minimum weight that is possible for this pair. This will be achieved when the similar bits are 1's and the dissimilar bits are split as evenly as possible between 1's and 0's. Thus the maximum minimum weight is bounded by

$$W \leq kV - W_o + W_o/2 = kV - W_o/2.$$

The maximum score difference between states is

$$S = \Delta S_m W$$

and to assure that there will be no overflow in a score register of s stages, we require

$$S < 2^{s-1} - \Delta S_m.$$

As an example, consider the k = 5, V = 2 code with $W_o = 7$ and a decoder with q = 8 levels of quantization so that $\Delta S_m = 7$.

$$W \leq 10 - \frac{7}{2}$$

or, since W is an integer

$$W \leq 6,$$

$$S = 7W = 42.$$

$$42 < 2^{s-1} - 7.$$

$$7 \leq s.$$

Thus 7 stages are adequate.

4. Sequential Decoding for Data Compression

This section describes a method for using redundancy in data, such as images, in the sequential decoding process. The theoretical work was done under a company-funded program,†† and the simulations were run under this contract. From the amount of computation required for decoding, this method was found not applicable to real-time TV.

However, for extremely austere space probes it has the advantage of requiring no additional equipment at the transmitter. When transmitter costs are much greater than receiver costs, as in space-to-earth or air-to-ground link, or where there are many transmitters and a single receiver, this method is likely to be cost-effective and may be the only possible choice.

For the space-to-earth link the savings are in producing software for general purpose computers on the ground rather than hardware in space. In addition to the obvious savings in reliability, and in power and weight on the spacecraft, cost and development time can be reduced by avoiding hardware design and qualification tests. It is even possible to increase the information rate of vehicles already flying by modifying the decoding program to exploit data redundancy.

The results of the simulation studies are reported here with only an abreviated description of the theoretical background, which may be found in the program report.†

---

†R. B. Blizard, H. M. Gates, and J. L. McKinney: *Convolutional Coding for Data Compression.* Martin Marietta Research Report R-69-17, November 1969.

Figure II-2 shows how a sequential decoder can be modified to use source redundancy. The Message Branch Metric Subprogram is added to modify the branch metric according to the likelihood of the decoded message. This type of decoder is effective against channel noise, like an ordinary decoder, but the additional information derived from the likelihood of the possible decoded messages permits it to decode with a lower SNR, and the gain achieved in this way can be credited to data compression. It is also possible to operate in a true compression mode, shifting more message bits into the encoder than channel bits taken out. This mode was used in the simulation because it is simpler to program, cheaper to run, and is more direct proof that compression can be achieved.

The first simulation was of a binary assymetrical source compressed 2 to 1. Two bits are shifted into the encoding register (Figure II-3) and a single bit is read out of the mod-2 adder during each unit of time. The source is assymetrical in that a 0 is many times more likely than a 1. As was shown in the report, the computational limit R comp is reached when

$$D = \log_2 \left(\sqrt{P(0)} + \sqrt{P(1)}\right)^2$$

where D is the number of channel bits per input symbol. In this case 1/D is the compression ratio.

The system described in Figure II-3 was programed on the CDC 6400 computer in Ascent. The system test consisted of the Fano algorithm decoder, encoder, and random number generator, which generated the binary information stream so there would be control over the number of binary ones. The system performs 12,000 calculations per second. The CDC 6400 has a 60-bit word, which represents the maximum constraint length possible for this system.

Simulations for this system were obtained for 8- and 48-bit constraint lengths at rate 2/1 convolutional codes and sequential decoding 300-bit blocks with a 12-bit tail. The decoder uses hard decision (Q = 2) input. The code, in octal, used for k = 8 is 716 as shown in Figure II-3 and 7162610413051275 for k = 48. The computer simulations were made for information content P(1) = 4%, 4.6%, 5%, and 5.5% where the theoretical R_{comp} is at 4.6%. The results of this simulation are shown in Figure II-4, where P_r (C ≥ L) is the probability that the number of computations C is greater than the number of computations per bit L when moving through a single branch.

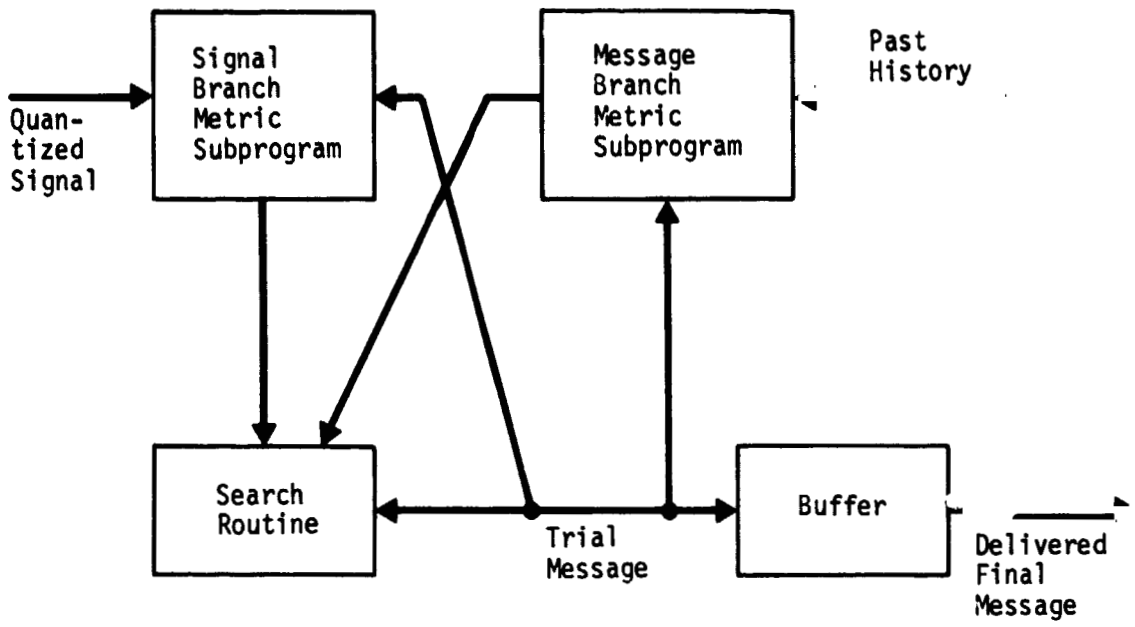


Figure II-2 Decoder Organization to Use Data Redundancy

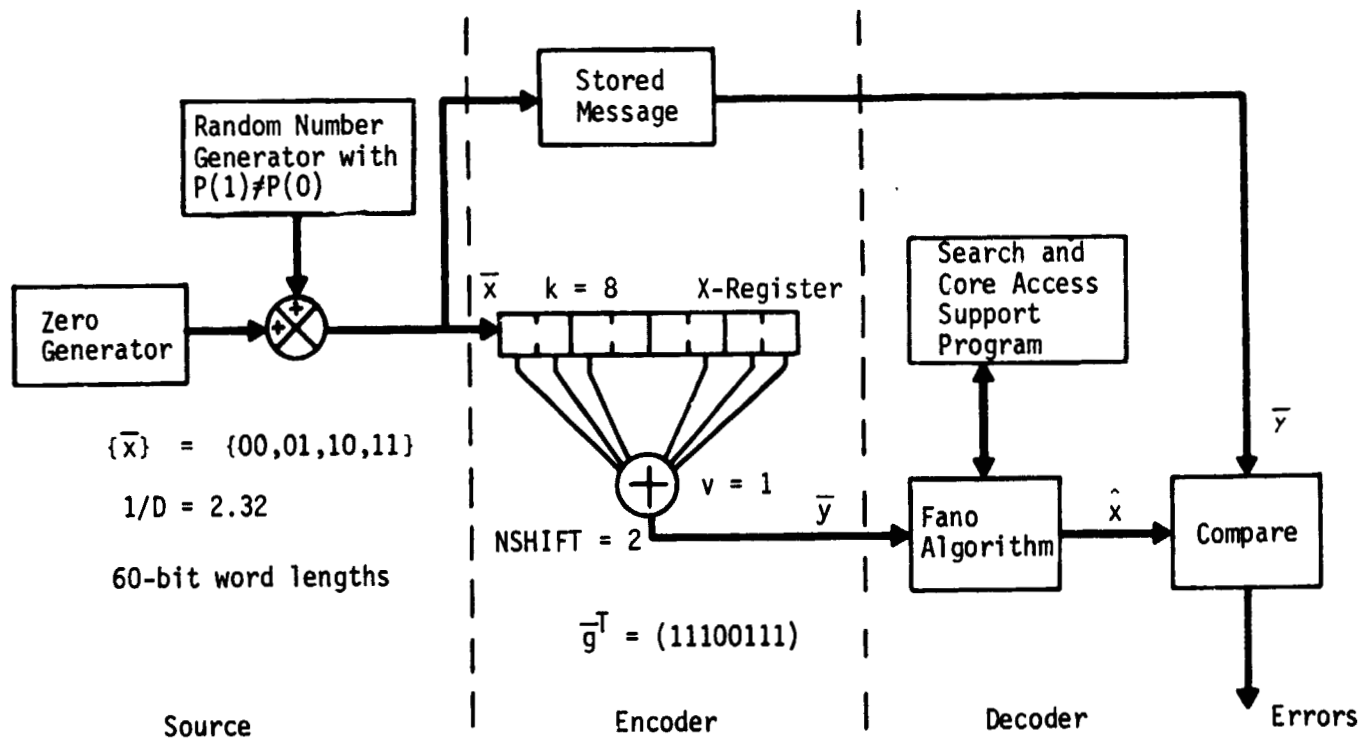
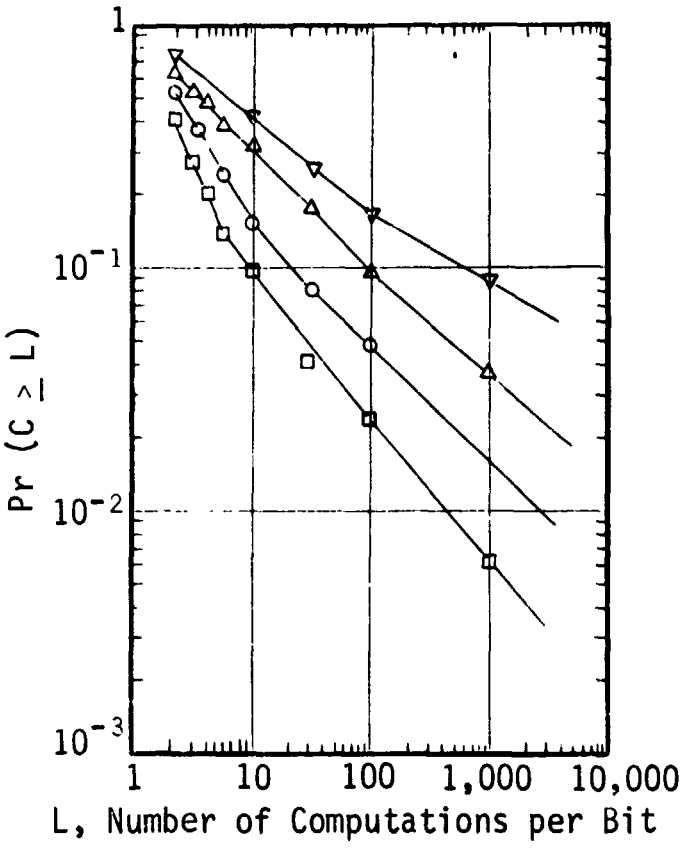


Figure II-3 Simulation of 2 to 1 Compression with Sequential Decoding

The second simulation was much more difficult, but more significant because it applies directly to a typical space-probe problem, the efficient transmission of digitized images. To verify the theoretical performance predictions for this technique we have experimented with seven digitized pictures that were supplied by R. Rice of JPL. These pictures vary in activity from typical flyby data, which generally are inactive, to pictures taken from landers, which are quite detailed and very active (i.e., difficult to compress).



Note:

Source Activity		Errors at k=8
▽	5.5%	4.85×10^{-3}
△	5.0%	5.4×10^{-3}
○	4.5%	6.7×10^{-3}
□	4.0%	9.5×10^{-3}

Figure II-4 Simulation of Convolutional Data Compression for Binary Symmetric Channel

The pictures are quantized to 6 bits, and vary in activity from an entropy of 1.9 to 4.67 bits/pixel on a first difference calculation of one picture element to the element immediately behind it. Each picture has a field of 684 x 600 elements. Picture data are summarized in Table II-2.

Table II-2 Summary of Picture Data

PICTURE	ENTROPY	AVERAGE GRAY LEVEL	STANDARD DEVIATION	D
1	1.90	50.3	2.48	2.3
2	2.60	54.9	9.22	2.85
3	3.06	34.8	9.85	3.4
4	3.33	50.8	12.45	3.65
5	3.65	41.5	10.08	3.9
6	4.00	48.5	11.79	4.35
7	4.67	41.2	11.61	4.95

Note: 1. Picture fields are 684 elements/line, and no less than 598 lines/frame.

2. Each pixel is 6 bits.

The key to the data compression decoding problem, whether using sequential decoding or block decoding, is to accurately predict what picture element value should be given the values of its surrounding neighbors. More explicitly, as the decoder moves through the picture from left to right and from top to bottom, the value sought is that of the next pixel given past data points.

For this simulation, the predicted value of the next gray level was a linear combination of the levels of the preceding pixel in the same line, and the three nearest pixels in the preceding line. With this technique we are making use of the line-to-line correlations as well as the correlation between adjacent pixels in a line.

A 3-to-2-compression ratio was used for operation since this is about all that can be expected to work with more active pictures. As Table II-2 shows, the parameter D exceeds 4 bits/pixel for picture 7, based on using only the preceding pixel for prediction. In this case, D is the number of bits/pixel that corresponds to the computational limit.

Again the system was programed on the CDC 6400, which provided sufficient core storage to store the data fields required to make proper evaluation of the base pixel. Searching time, however, is slow (100 searches per second) due in part to the

evaluation of the branch metrics. The results of the simulation are shown in Figure II-5 where the average number of searches per line is shown as a function of first difference picture entropy. Each picture contains 684 elements per line and we are simulating 683 coded elements. Our programs have been limited to 40,000 searching operations. We attempted to decode picture 6, first difference entropy = 4.00, with some interesting results. The system started decoding properly, but after 50 or so lines the maximum number of searches was exceeded and an erasure occurred. The decoding of successive lines deteriorated rapidly where erasures occurred sooner on each line than the line before. Complete erasures occurred 7 to 8 lines after the first erasure was detected.

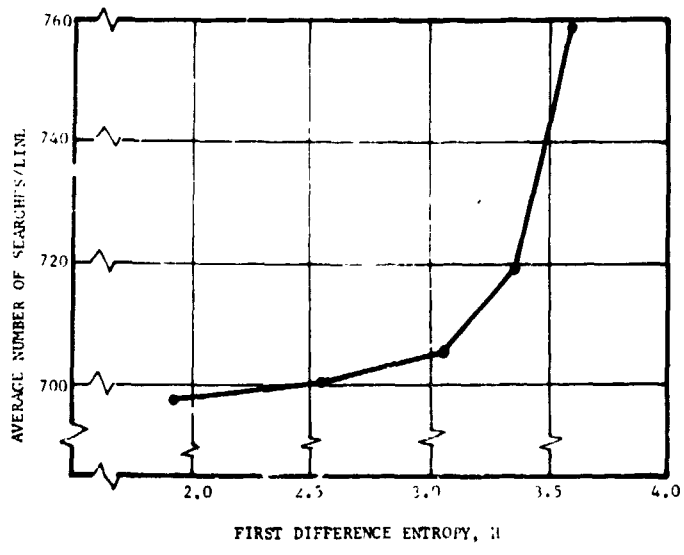


Figure II-5 Performance of the 6/4 Decoder-Compressor

The system was forced to restart halfway down and the same phenomena occurred after several lines. The decoder simulation of picture 7 would exceed the 40,000-calculation limit with every line. Only a small portion of picture 7 was simulated because of the long run times encountered.

The results imply that once we exceed an entropy of a value in the neighborhood of 3.65, we exceed R_{comp}. From Figure II-5 we see that for H = 3.65, $\sigma = 3$. For $\sigma = 3$, D = 3.9, which is very near the output value of the convolutional encoder of 4. Theoretically, at least, activity any higher than H = 3.6 or so should be difficult to decode. This was verified by pictures 6 and 7. The simulation uses a pattern of 5 elements but the entropy was computed on two patterns (first differences between the preceding element and the base pixel). Thus some information should be decoded above the first difference entropy of 3.65.

The agreement with theory indicates that performance can be predicted for any data source with known statistics.

5. Iterative Viterbi Decoding (IVD)

The following pages are included in this report to document the progress made on a new idea for a coding and decoding scheme. The technique may permit a closer approach to channel capacity than is possible with sequential decoding. Not enough has been done, either analytically or with simulation, to permit a valid estimate of the ultimate performance, but accumulated results in the near future should help determine whether further work along these lines is warranted.

Sequential decoding of convolutional codes has been very successful in improving the efficiency of power-limited communication links with a modest amount of decoding equipment. However, sequential decoding behaves badly at information rates above half the channel capacity, and some new method must be found if we are to make further progress. The combination of block codes with convolutional codes and sequential decoding is one approach that is being worked now. Another approach based on short constraint-length convolutional codes and the Viterbi decoding algorithm, is outlined below.

The Viterbi algorithm has been shown to be optimum in that it chooses the most likely message. The problem is that, like correlation decoding of block codes, the Viterbi algorithm is impractical for large constraint lenghts. Some means must be found to increase the constraint length without making the decoding job impractical.

The natural approach is to use several channels of convolutional codes with a block code interconnecting the channels. If this is done, it is important to get efficient cooperation between the two types of codes. For instance, if the convolutional channel outputs are connected with a BCH code, the BCH code would then correct the errors and erasures of the convolutional channels. This procedure is inefficient because it does not utilize the information received by the convolutional channels when they are unable to decode correctly. Only a small amount of additional information is ordinarily needed to make the difference between failure and success in a coded channel, but the BCH code would have to furnish the entire message without making use of the information in those convolutional channels that are in error.

Iterative Viterbi Decoding (IVD) supplies additional information in small increments to help in decoding the troublesome parts of the message. A single Viterbi decoder makes several passes over the received signal improving the message reliability each time.

From preliminary paper studies, it appears that hardware Viterbi decoders can be built with integrated circuits to run at message rates of several megabits per second, and thus the IVD should be capable of performance in the neighborhood of a megabit per second.

a. Configuration - Figure II-6 shows a simple convolutional encoder for IVD. At the left is an ordinary k = 5, V = 4 encoder. The connections to the four mod-2 adders are shown in matrix form for clarity. The outputs from the first two mod-2 adders are sampled by the switch in the ordinary way. The remaining two are combined (mod-2) with a 30-bit relative delay in the second shift register. Thus we have a rate 1/3 code.

The IVD decoder is shown in Figure II-7. The incoming signal is converted to digital form in the integrate and dump (I & D) circuit. Each received channel symbol is now represented by a digital number (3 bits is sufficient), and these are stored in the buffer for periodic discharge into the R register. The contents of the R register are circulated at a speed several times the channel speed so that the Viterbi decoder makes several passes over each part of the I & D output. A good choice for Q and its efficient use will be important in obtaining high performance. Two possibilities for Q are being considered. The first is simply the rate of increase of the "score" used in the decoder. The score in the Viterbi algorithm is the correlation of the received signal with the assumed message. It may be sufficient to use a single bit for Q indicating either reliable or unreliable results; but since Q would only have to be determined at intervals of several message bits, the use of two bits would not add very much to the memory requirement. Another criterion is the difference between the scores of the selected message and that of the message it defeated in the comparisons that occur with each message bit. This difference indicates the reliability of each binary choice in selecting a message. If the difference is large it indicates that the message is likely to be correct.

As soon as some message is stored in the M register, the decoder can start making use of the third symbol in each triple. This symbol was formed as the mod-2 sum of the third

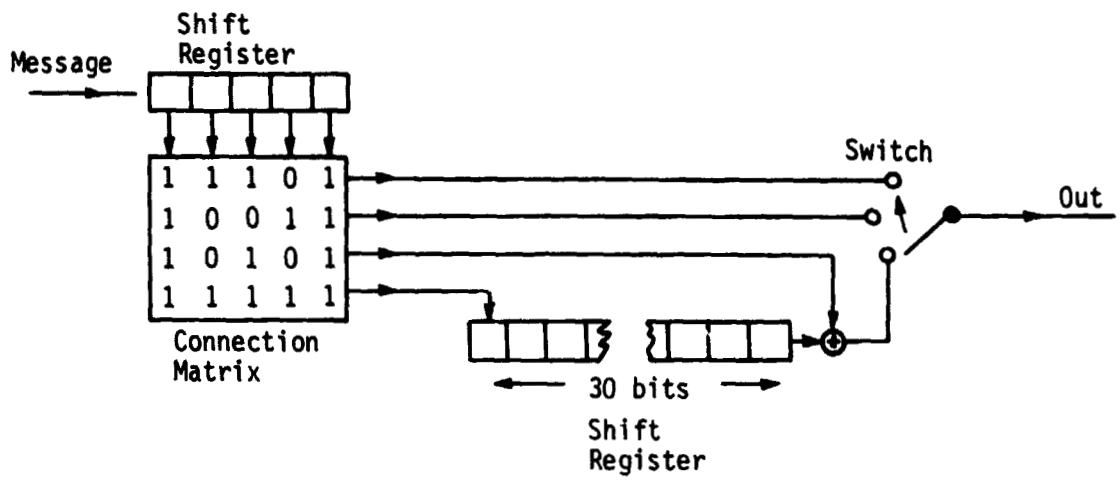


Figure II-6 A Simple Convolutional Encoder for IVD

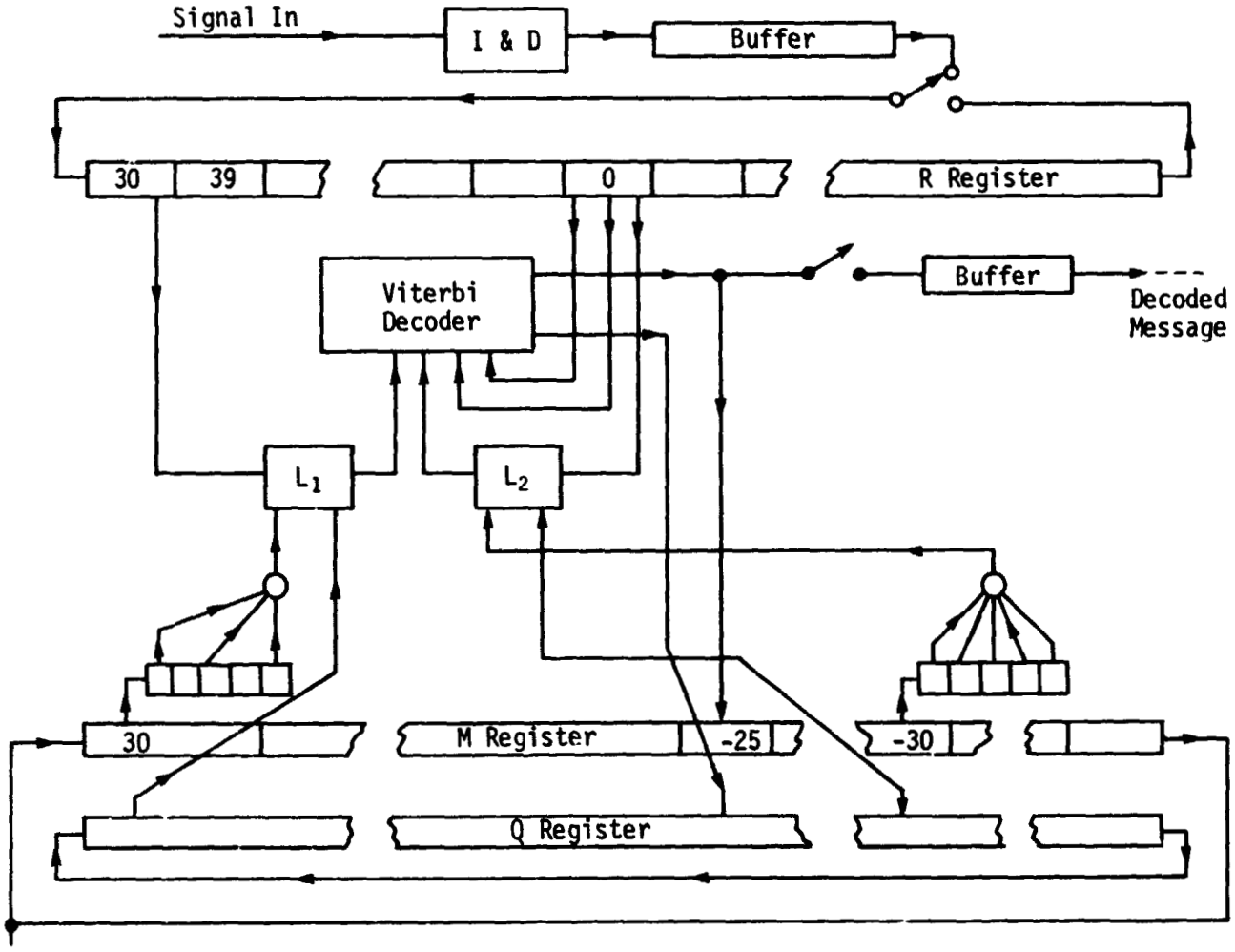


Figure II-7 Decoder for Coder of Figure II-6

check symbol out of the connection matrix (the one with connection vector 10101 in Figure II-6) and the fourth (with vector 11111) from a time 30-message-bits earlier. To make use of the third check symbol, the decoder takes the previously decoded message from the M register, at a position 30 bits downstream, forms the fourth check symbol (vector 1111) and, if this is a 1, it changes the sign of the I & D number taken from the R register. This operation is performed in the logic box L_2. Another function of L_2 is to use the quality function from the Q register to make the I & D number 0 when it is unreliable.

After one pass has been made, the fourth check symbol (11111) can be used. The I & D number is found in the R register 30 bits upstream and its sign is corrected according to the assumed message in the M register. The logic box L_1 is identical to L_2.

As successive passes are made, the third and fourth check bits are available a greater fraction of the time until some limit is reached. The probability is high that both the third and fourth check symbols will be available for decoding, and therefore, the probability of error will approach the value for a V = 4 (rate 1/4) code at the same energy per channel symbol. Thus only 3/4 as much power is required, or an improvement of 1.5 dB is achieved over the corresponding V = 4 code.

b. More Powerful Codes - If Viterbi decoding is to be improved sufficiently to compete with sequential decoding, many more code vectors must be combined in the encoding process. In the first stages of decoding only a small amount of signal (the vectors that are not combined) will be available, and the crucial question is whether decoding will bog down before most of the check bits are available. To ge insight into this problem, a simplified system was considered. This system consists of an erasure channel with "message units" or words consisting of several message bits each. The results of this analysis give the conditions for successful decoding and indicate an approach to deciding how many vectors to combine in pairs, triplets, etc.

c. Analysis of an Erasure Channel - A coded channel with an indication of the reliability of the decoded message units can be treated as an erasure channel if fuzzy results are acceptable. In an erasure channel, some of the message units are known with certainty and the others are known no better than they would have been without the received signal. To make the real channel with reliability indication look like an erasure channel, a threshold

T is chosen for the reliability indicator Q at a level high enough so that when T < Q, the message unit is almost certainly correct. "Almost certainly" depends on the use to be made of the message, which will not be considered in this section.

1) The code has a low rate and each message unit is encoded into a large number of channel bits;

2) The code structure is flexible enough so that the effect of changing the number of channel bits that are available for decoding a given message unit is the same as changing S by an equivalent amount. That is, the effective value of S is proportional to the number of bits available for decoding the message unit;

3) A coding superstructure will be added by transmitting channel bits that link the message units. Each of these bits is the mod-2 sum of N bits that would normally appear in the codes for N different message units chosen from a large amount of message material so that any two message units will not be linked by more than one of these bits. The message units might, for instance, be words of an orthogonal code, but of course, for IVD each represents a message segment a few times k bits.

A certain fraction A_1 of the channel symbols is used in the ordinary way, that is, they are not shared by two or more message units. A fraction A_2 is shared by two units, and so on up to A_N which is the fraction shared by N units.

At the first decoding pass, the deletion rate is

$$D_1 = D(SA_1)$$

where S is the total SNR for the channel and SA_1 is the amount available without any previous decoding.

On the second and subsequent passes, more signal is available and D_i, the deletion rate after the i'th pass, approaches a limit as "i" increases. For the i'th pass, the deletion rate that determines the amount of signal (or number of channel symbols) available for decoding is D_{i-1}. If a symbol is shared by j units, its probability of being available is $(1-D_{i-1})^{j-1}$. Since it is

shared by j units, it has a chance of being used in j places; and thus the contribution of all symbols shared by j units is $jSA_j\ (1-D_{i-1})^{j-1}$.

The deletion rate following the i'th pass is therefore

$$D_i = D\left[S\sum_{j=1}^{N} jA_j\ (1-D_{i-1})^{j-1}\right]$$

To optimize the ultimate performance, the A_j should be chosen to maintain $D_i < D_{i-1}$ to as low a value of D_i as required with $\Sigma\ A_j$ as small as possible. However, in a practical system, a limit will be placed on the number of passes u, and in this case the objective is to minimize D_u.

As an example that is mathematically tractable, consider $D(S) = \exp(-S)$. If we choose $A_j = 1/j(j-1)S$, then in the limit for large N

$$\lim_{N\to\infty}\left[S\sum_{j-2}^{N} j\ A_j\ (1-D_{i-1})^{j-1}\right] = \ln D_{i-1},$$

and $$D_i = D_{i-1}\exp(-SA_1).$$

Thus, letting $N\to\infty$ permits a choice of the A_i that gives a deletion rate that decreases by a factor $\exp(-SA_1)$ with each decoding pass. The total amount of signal S that is required is easily determined by recalling that

$$1 = \sum A_j = A_1 + \sum_{j=2}^{\infty}\frac{1}{j(j-1)S}$$

$$= A_1 + \frac{1}{S}$$

Thus $$S = \frac{1}{1-A_1}.$$

d. The Use of the Reliability Indicator Q - In the previous discussion it was assumed that a threshold T for the reliability indicator Q was chosen to give negligible probability of error with T < Q. No attempt was made to use channel symbols that required knowledge of any message unit with Q < T, and full weight was given to those that met the criterion. It is important to have a comprehensive understanding of the use of Q.

Let the normalized received voltage for a particular detected channel bit on a memoryless Gaussian channel be x so that $p(x|1)$, the probability of x when 1 was transmitted is given by

$$p(x|1) = \frac{\exp\left[-(x - D)^2/2\right]}{\sqrt{2\pi}}.$$

This leads to the following equation for the probability ratio C:

$$C = \frac{P\ (1|x)}{p\ (0|x)} = \exp\ 2x.$$

When the channel bit is shared by two or more message units, if an odd number of the bits from the other message units are in error, the effect is the same as a change in the sign of x. If this probability is J, then

$$p\ (x|1) + \frac{1}{\sqrt{2\pi}}\left\{(1-J)\ \exp\left[-(x-1)^2/2\right] + J\ \exp\left[-(x-1)^2/2\right]\right\},$$

and

$$C = \frac{(1-J)\ \exp\ x + J\ \exp\ (-x)}{(1-J)\ \exp\ (-x) + J\ \exp\ x}$$

The logarithm of C is shown in Figure II-8 for various values of J.

The proper quantity to use in determining the score increment ΔS is ln C, and for J = 0 the output of the integrate-and-dump (I & D) circuit is a quantized representation of K + ln C, where K is chosen to make ΔS non-negative. For other values of J the I & D output should be modified to conform as closely as possible to the actual value of ln C without introducing too much hardware complexity. The shape of the curves in Figure 8 suggests a combination of a change in slope and limits on the extreme values of ΔS.

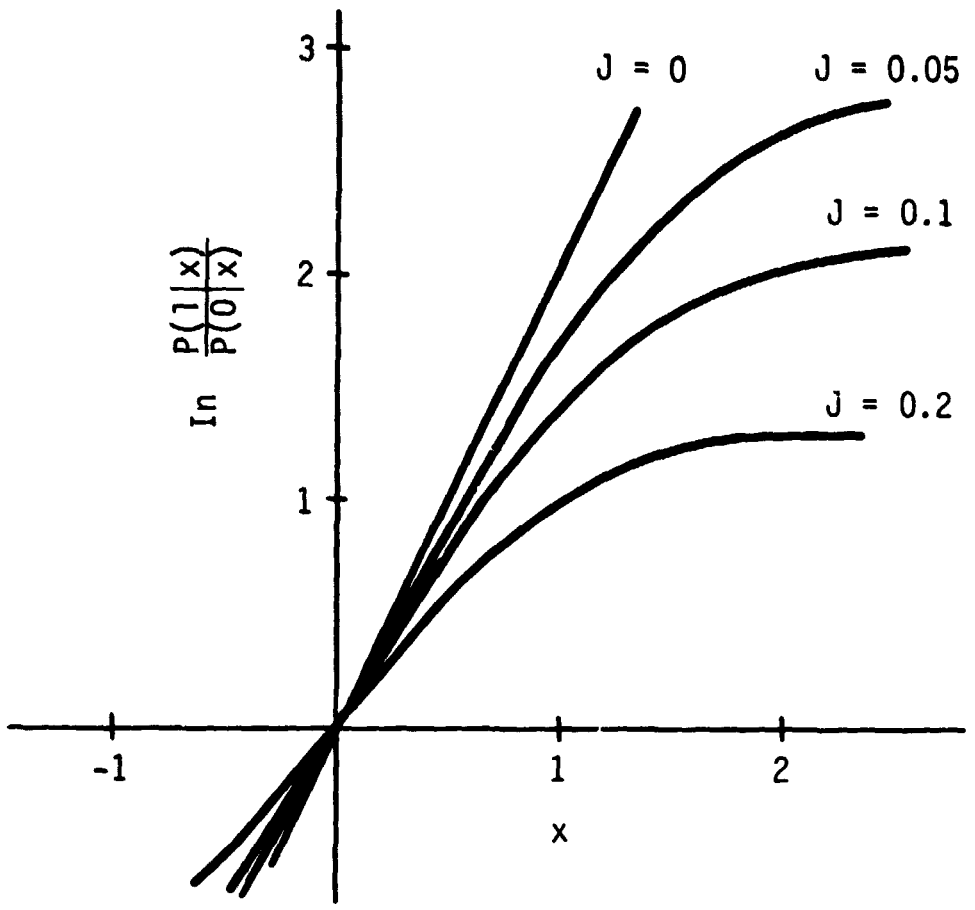


Figure II-8 The Logarithm of the Probability Ratio as a Function of the Signal-to-Noise Ratio for Various Values of J.

e. Projected Performance Improvement - Under the extension of this contract we will simulate decoding with very low signal-to-noise ratio and various reliability indicators. The results should permit a choice of reliability indicator and also a better basis for predicting performance. If IVD still looks good, simple and then more elaborate schemes will be programed.

If IVD is a valid means for approaching channel capacity, careful design of the code and the decoding algorithm will be required to realize its potential. The throwing away of many tenths of a dB cannot be afforded.

If, however, it turns out that there is a fundamental difficulty that imposes a limit similar to the computational limit of sequential decoding, the idea will be abandoned and time will be devoted to other things.

6. Random Number Generators

Two random number generators have been prepared for the CDC 6500 and the IBM 360/44. The numbers generated are used to simulate random noise which would occur in a data transmission channel.

The basic idea for both generators is to generate, by congruence methods, a sequence of numbers (from 0 to 999) in such a way that the sequence will not repeat (cycle) until a very large amount of numbers have been generated (this large amount of numbers is called the "period" of the cycle). The numbers should be as unrelated as possible.

The basic formula for X_n (the nth number generate is

$$X_{n+1} \equiv aX_{n+b} \text{ (modulo T)}.$$

The symbol "$\equiv$" may be interpreted as: $a \equiv b \pmod{C}$ only if C is an exact divisor of $a - b$.

1) The formula used for the CDC 6500 uses a = 23, b = 0, X_o = 47594118, and T = 1000000001;

2) The formula used for the IBM 360/44 uses a = 23, b = 0, X_o = 47594118, and T = 10001009.

The reason that the second formula had to be devised is that the IBM 360/44 is a 32-bit word machine, and the CDC 6500 is a 60-bit word machine. The IBM 360/44 could not handle the magnitude of the numbers which are involved in the first formula.

The numbers generated in both formulas vary from 1 to magnitudes far larger than 999. Therefore, in both formulas, the second, fourth, and sixth digits (from the left) are chosen. This generates a sequence of random numbers between 0 and 999 inclusively.

The cycle period for the first formula is approximately 5×10^6, and the cycle period for the second formula is approximately 10^7.

The first formula was suggested by a publication by the National Bureau of Standards* and has been tested for randomness in several ways.

The second formula is more advantageous in that it has a cycle period of almost twice the cycle period of the first formula.

For the second formula, three randomness tests have been made The first test was a simple distribution test, the second was a Chi-square test, and the third was an auto-correlation between the nth and (n + k)th elements for k varying from 1 to 4000. The strongest auto-correlation coefficient was 0.2, which means there is virtually no linear correlation between the elements. The Chi-square test resulted in a probability of 0.987 that a random sample gives no better fit.

7. Burst Effects of Viterbi Decoders

Error bits on the output of a Viterbi decoder tend to occur in bursts as verified by simulation. Three million information bits were processed by our Viterbi K = 5 V = 2 simulated decoder on the CDC 6500 computer. Input data was assumed as all zeros, and simulated Gaussian noise was impressed upon it in the form of 3-bit quantized input from a pseudo-random-number generator. The quantizing scheme was suggested by Jacob.†

---

*National Bureau of Standards Applied Mathematics Series 55, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, p 950.

†E. M. Jacob: "Sequencial Decoding for Efficient Communication from Deep Space." *IEEE Transaction for Informations Theory*, Vol. COM-15, No. 4, August 1968, p 492.

Output of the program onto magnetic tape were the patterns of error bursts. These bursts were counted by weights (number of errors per burst) and burst length (where k-1 zeros after the last one bit constituted an end of the burst). Table II-3 presents these counts for simulation runs using two encoder connection codes; one is an optimum connection with maximized minimum word weights, and one is not. Figure II-9 shows the number of burst totals with its uncertainty vs signal-to-noise ratio for the optimum connection. Although the table and graph cannot show a difference due to encoder connection large enough to be statistically significant, the data implies that the optimum connection gives a higher number of bursts but a lower bit rate. It is therefore necessary to consider the effects of bit burst error rates in the decoder application to make a choice of connection codes.

Table II-3 Each Run 500,000 Bits (K = 5, V = 2)

Bursts of Weight:	2.5 db	3.0 db	4.0 db
1	112	58	9
2	30	11	3
3	120	68	10
4	55	22	1
5	46	23	1
6	46	24	2
7	47	28	3
8	15	4	0
9	31	13	3
10	13	6	0
11	12	7	0
12	5	1	0
13	12	5	0
14	3	2	0
15	3	1	0
16	3	0	0
17	1	0	0
18	0	1	0
19	1	0	0
20	3	0	0
21	0	0	0
22	0	0	0
23	0	0	0
24	0	0	0
25	0	0	0

Count of error bursts of weights from 1-25 for 0.5×10^6 bits processed by Viterbi decoder K = 5, V = 2

Bursts of Length:	2.5 db	3.0 db	4.0 db
1	112	58	9
2	25	9	2
3	7	2	0
4	103	63	10
5	37	19	2
6	38	20	1
7	20	8	0
8	44	22	2
9	23	17	2
10	16	6	0
11	18	6	1
12	13	7	0
13	14	5	0
14	13	11	3
15	11	4	0
16	10	6	0
17	2	0	0
18	8	0	0
19	10	3	0
20	8	3	0
21	1	1	0
22	0	0	0
23	4	3	0
24	1	1	0
25	0	0	0

Count of error bursts of lengths from 1-25 for 0.5×10^6 bits processed by Viterbi decoder K = 5, V = 2

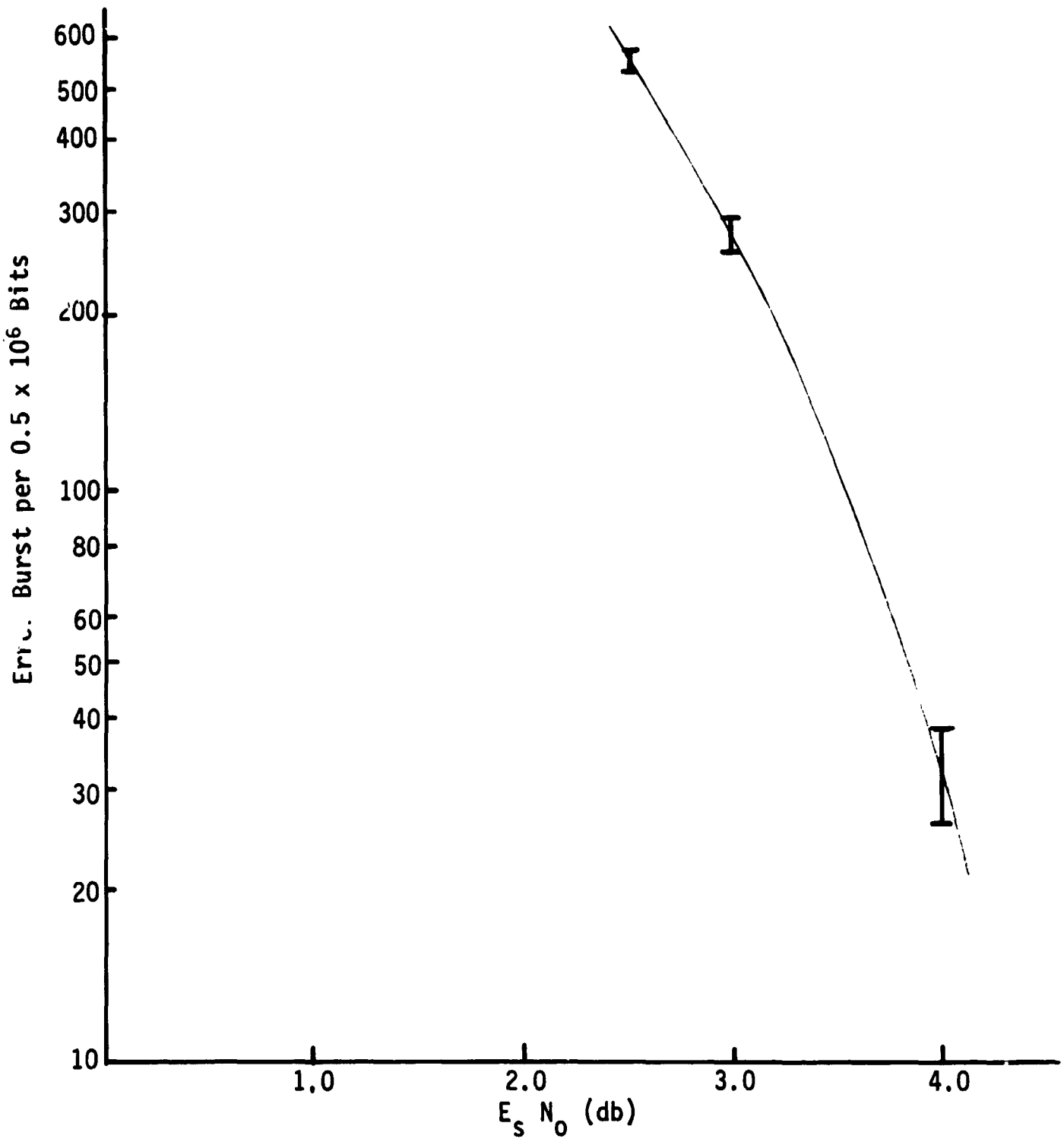


Figure II-9 Burst Statistics

B. DATA COMPRESSION STUDIES

Delta modulation, polynomial prediction and interpolation, and the use of data redundancy in sequential decoding have been considered as ways to compress the TV and voice signals.

For voice, delta modulation was chosen because of its simplicity and effectiveness and because there is considerable work already done in developing the technique.

Delta modulation was also considered for TV, but was rejected as not being powerful enough for the high data rate needed for real-time image transmission.

The polynomial algorithms were narrowed down to zero- and first-order predictors and interpolators, and these were programed in many variations for the TV. The algorithms were tested on a picture that may be typical, and subjective evaluations were made. The method scoring highest also requires more hardware, and it may be difficult to make it run at the required speed. The studies have led to ideas for improving on the algorithms, and the ideas will be evaluated during the extension of this contract.

The use of data redundancy in sequential decoding was studied before starting the contract. Under the contract we have verified the theory through computer simulation, and also determined that decoding would be too slow for TV even with a hardware design. There are, however, other applications that could benefit from this technique.

The data compression studies are described in detail below.

1. Voice Digitization and Compression

This section describes the results of a study to determine acceptable methods of digitizing real time and delayed voice transmissions; the objective being to determine the necessary voice channel bit rate requirements and other pertinent signal characteristics for the purpose of integrating voice and telemetry into a single time division multiplex digital communications channel.

The study included an intensive literature search as a basis for the description and recommendations. Dr. Denmer Baxter of Martin Marietta's Orlando Division has been particularly helpful in furnishing unpublished material in this area of specialization.

Much of the available literature pertains to military voice communications in which voice channel bit error rates are higher, and intelligibility goals lower, than for the USB application.

This plus the fact that present work was done without the benefit of experimental verification has led to some conservatism in recommendations of bandwidth requirements and implementation techniques. However, since both real time and playback voice data represents only a small fraction of the total data compared to TV, the effect on the overall channel requirements is relatively slight.

a. Voice System Summary - The general constraints on the voice handling system include high intelligibility, moderate channel bandwidth, and simple mechanization. This study has defined a system configuration that is predicted to meet the USB needs with modest design and development effort. The system contains the following components:

1) Analog preprocessing (band limit and probably square-root-law syllabic compression);

2) Delta modulation using any one of 3 approaches: a 20 Kb/s clock with double integration and prediction, delta-sigma integration, Winkler's high information technique delta modulation (HIDM);

3) Demodulation, post filtering, and square law expansion to compensate for any pre-modulation syllabic compression.

The study indicates delta modulation is superior to PCM for system bit rates under 30 Kb/s and that vocoders would be too complicated for this application. Voice data redundancy reduction is difficult to justify in view of the small part voice channels would required of the total USB data bit rate, and the complications to modulation and demodulation that would arise.

The 3-delta modulation approaches suggested as candidates are listed in order of increasing complexity and performance. Any of the three should provide the desired performance assuming bit error rate equal to 10^{-4} and a 20-Kb/s data rate allocation for the real time voice channel. A 640-Kb/s rate is assumed for the recorded voice channel played back with a speed-up of 32 times.

The system development area most open to question, meeting the 90% intelligibility requirement, has not closely matched USB requirements with the little experimental work reported to date. Intelligibility measures are of course dependent on both the system parameters and the test material; assuming a Fairbanks-Rhyme-articulation test would be used in development evaluation work.

The use of companding and demodulator filtering are two important design areas that strongly affect intelligibility. Although the parametric variations in these areas are a developmental problem (for example, compander dynamic range specification, and post-filter cutoff frequency and rate), the design of such hardware does not appear to be critical.

b. USB Voice Channel Requirements - The performance specified in the NASA-furnished guidelines is 90% word intelligibility (for analog FM voice, this corresponds to an rms/rms post-detection signal-to-noise ratio of 14 db).

Other considerations should include moderate power consumption, bandwidth, and equipment complexity. Approaches should be limited to current technology requiring only modest design and development cost.

c. Digital Coding Techniques - There are several possible methods of coding voice signals, which may be broadly grouped into: (1) PCM, (2) differential or delta modulation, and (3) information content (e.g., vocoders).

When speech is used, either the waveform or the information content can be digitized. The speech waveform is represented by a signal whose nominal upper frequency limit is 3500 Hz, although the information content is only 25 to 50 bits per second.[1] The low data rate corresponding to information content can be approached with vocoders.

For USB the bandwidth compression techniques of the vocoders cannot be used because their intelligibility is usually not as high as the waveform digitization techniques, nor is the voice quality as natural.

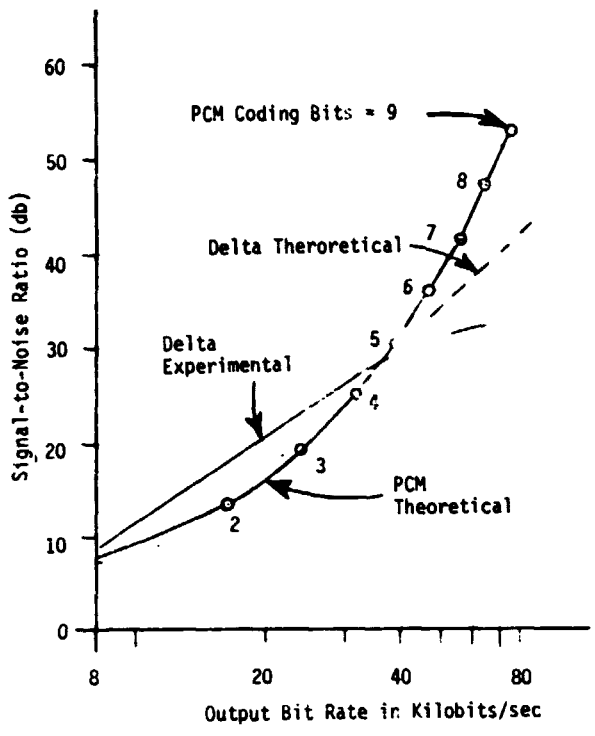


Figure II-10 Signal-to-Noise Ratio for Delta Modulation and PCM

Figure II-10, shows signal-to-noise ratio for the range of output bit rates of interest for PCM and delta modulation. The experimental delta modulation used was a double-integration delta, and the comparison shows the 5-db improvement of delta modulation over PCM at a 20-Kb/s line rate.[2] There is no significant signal to noise difference for the two systems at 8000 and 40,000 bits per second (1 and 5 bit per sample PCM samples at 8000 samples per second), however, the delta modulator hardware would be simpler to implement and the one bit per sample encoding easier to decommutate. Therefore only delta modulation techniques will be treated further.

d. Delta Modulation Implementation - For an audio bandwidth of 3 KHz, probably little more than 18 Kb/s of delta modulation is required to render a signal equal to any practical economical PCM system.[7] However, to assure meeting the 90% intelligibility requirement, 20 Kb/s will be allocated to the real-time voice channel and 640 Kb/s to the high-rate voice playback.

With this sampling rate, three delta modulation candidates, in order of priority and increasing complexity, can be named as suggested techniques. These are:

1) Limited double integration (Ref. 2);

2) Delta-sigma integration (Ref. 3, 8 and 9);

3) HIDM (Ref. 4).

Each is briefly described below following a description of the basic operation of a delta modulator.

Delta Modulation - Basic Operation - The block diagram, Figure II-11,[11] shows a conventional delta modulation scheme. The letter symbols in this diagram corresponds to those on the pulse trains and clock train a, pass through or are inverted by the pulse generator, depending on the waveform at the gating input c. The output pulse train b, which, in the case shown, has eight inverted pulses sent out over the communications channel to the decoder. Here a waveform f is developed by integrating pulse train b. The integrator at the encoder produces an identical stepped waveform at point e. The reconstructed waveform can be viewed as a "quantized" or incremental approximation of an analog signal.

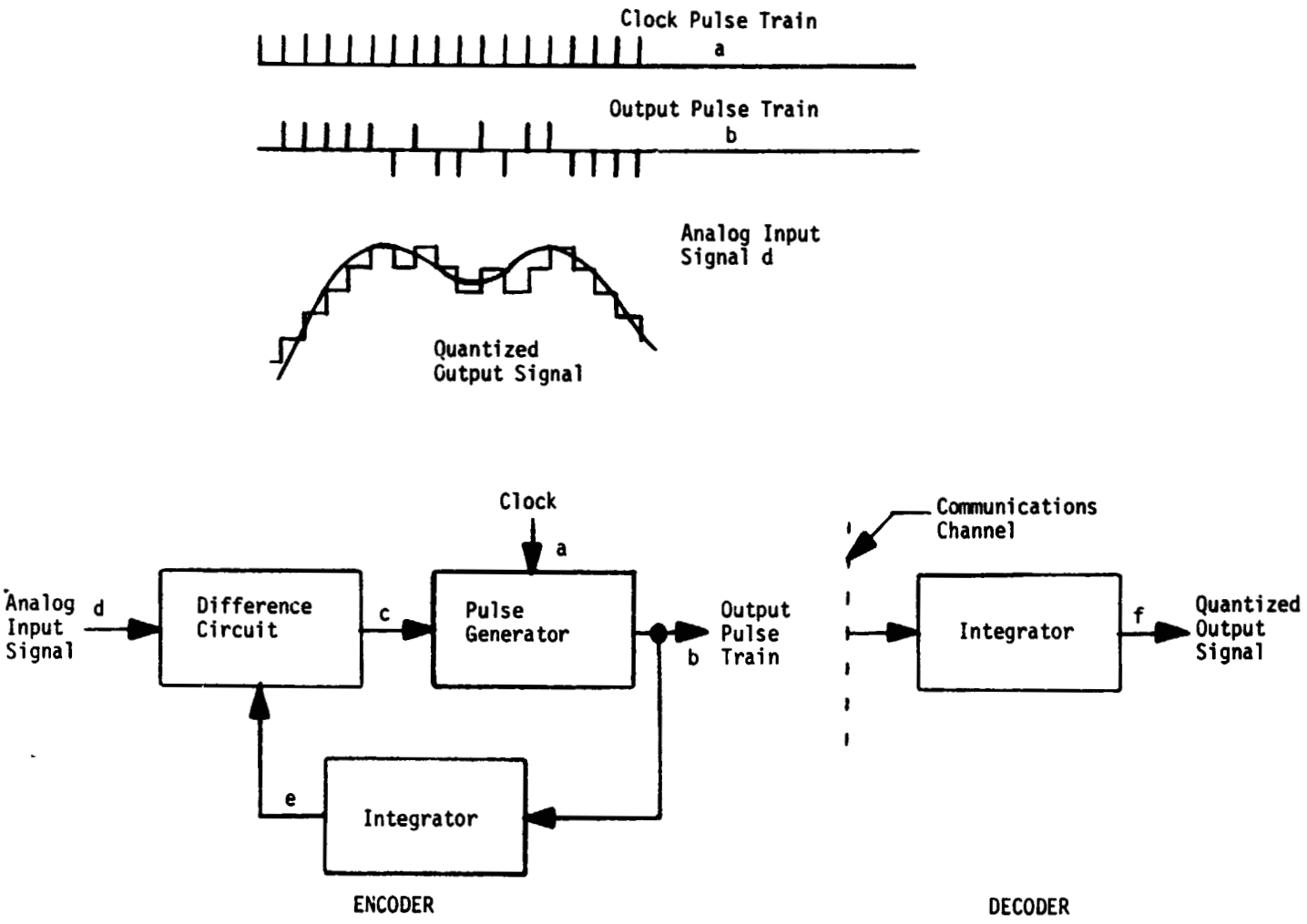


Figure II-11 Basic Delta Modulator Block and Response Timing Diagram

At each clock-pulse interval a comparison is made between the analog input signal d and the local approximation e. The comparison is, in effect, between the present value of the analog input signal and its value approximately one clock pulse interval earlier. If the present analog input voltage is greater than the delayed quantized output signal, the pulse generator is triggered by a positive step waveform and a positive output pulse appears at b. If the present analog input voltage is less than the stepped signal at e, a negative step triggers the pulse generator and a negative pulse appears at output b. For circuit simplicity in practice, the positive and negative pulses would be encoded as the presence and absence of a unipolar pulse at each sample clock time.

In this manner the reconstructed waveform is continually compared to the analog input signal, and quantized differentials are encoded to minimize the error.

Limited-Double Integration Delta-Modulation - When the period of the integration approaches or becomes less than the period of the lowest frequency of interest, the system characteristics change significantly. This type of modulation is called "Limited-Integration Delta Modulation." Since the voltage from the limited integrator decays continuously, the memory of the integrator is relatively short. Consequently, errors due to noise in transmission have only short-term effect. For conventional delta modulation with large-time constants, such errors may affect the system for a prolonged period.

A better approximation to the input signal is obtained when the integration is double rather than single. In this system, called "Double Integration," every input pulse has the effect of changing the slope instead of the amplitude of the output signal. This results in a smoother approximation to the input.

However, additional delay is introduced in the feedback circuit, causing the system to be somewhat sluggish. To remedy this, the difference circuit is fed from a point slightly ahead of the output of the double integrator thus introducing some prediction. Comparison shows that the approximation to the original signal is better with prediction than without. Double integration has the effect of breaking long steady-state patterns into higher frequency components which can easily be filtered.

Studies of double integration have shown a useful improvement in signal-to-noise ratio.

Delta-Sigma Modulation - A modulation technique called "Delta-Sigma Modulation" employs the same functional components as conventional delta-modulation but arranged somewhat differently, 4, 9, 10. The integrator is placed in the forward loop while the input signal and feedback pulse train are subtracted from each other ahead of the integrator. A low-pass filter or short-term integrator is used in the output decoder (see Figure II-12). The remainder of the circuit is essentially the same as the other delta-modulation methods described.

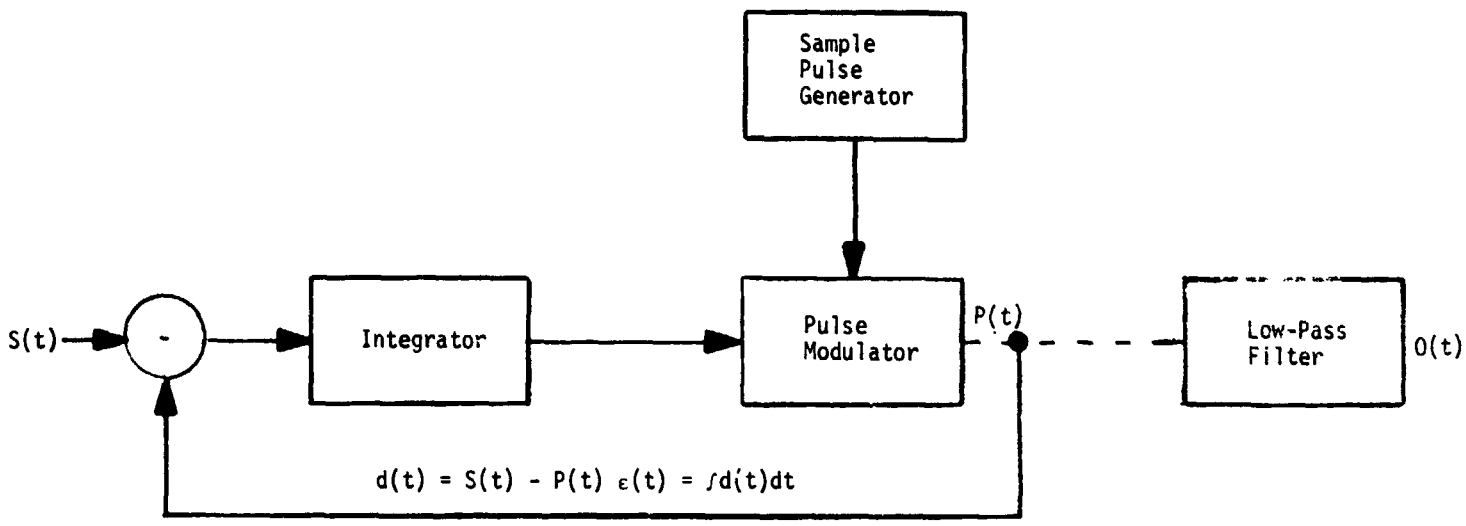


Figure II-12 Delta-Sigma Modulator Block Diagram

Its advantage over conventional delta-modulation is that cumulative transmission errors are not possible since signal-amplitude information is directly transmitted. Only a partial integrator at the output is necessary to provide dc level information.

High Information Delta Modulation - High Information Delta Modulation (HIDM) is a method suggested by Marion R. Winkler.[4] By weighting the contribution of successive pulses it provides more information per pulse. Excellent voice communications at 20,000 pulses per second have been demonstrated.

The essential difference between HIDM and conventional delta-modulation is in the manner of counting amplitude levels. Counting for HIDM is in binary steps and proceeds exponentially for the duration of a sequence of pulses of one polarity.

Initially, with the first pulse following a delta sign change, a transmitted level would be one unit. If this is insufficient, another pulse adds another unit, doubling the quantity.

The total count proceeds exponentially with pulses transmitted, according to 2, such as 1, 2, 4, 8, 16, 32, 64, etc.

Should the increment be too large, the pulse and count directions reverse, reducing in magnitude by a factor of two. In two steps it retraces the "over-correction," searching for the correct magnitude (see Figure II-13).

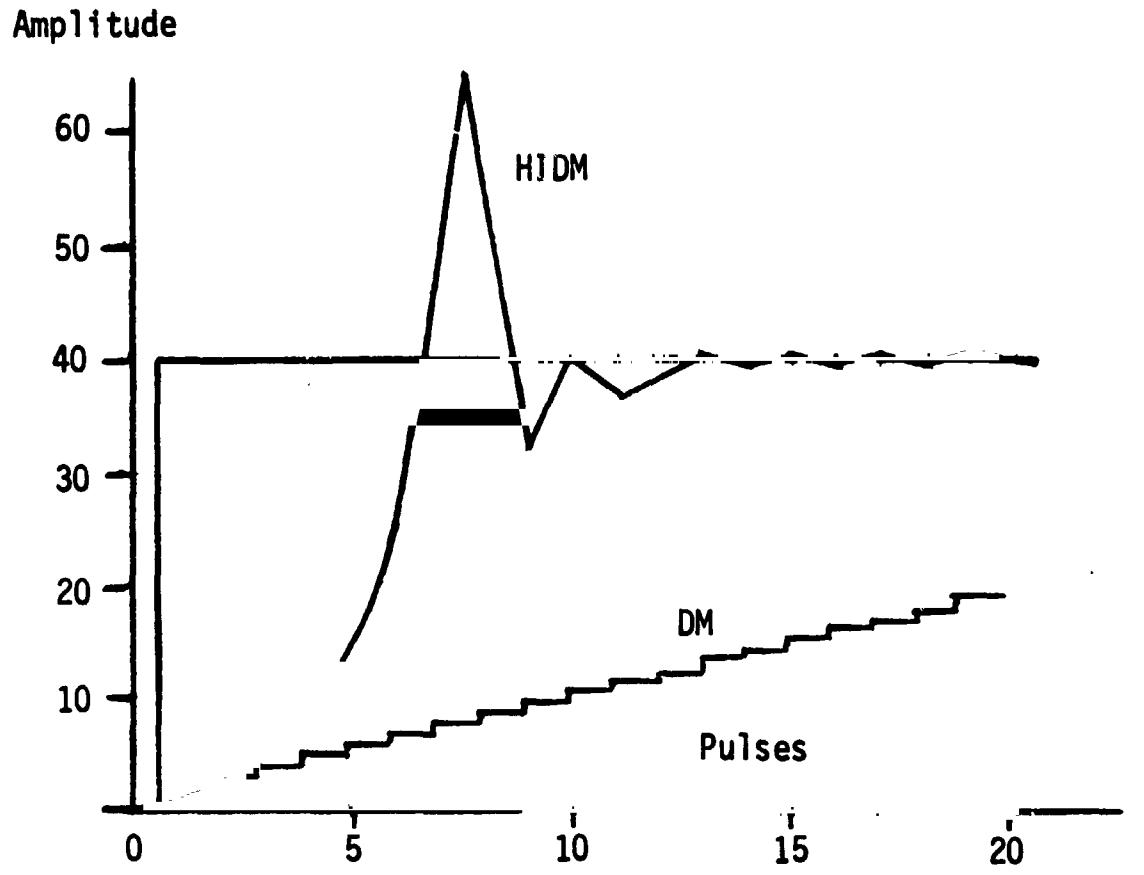
Other Delta Modulation Methods - The 2-bit delta modulators (2 bits per sample) and the exponential modulators have been excluded from further consideration due to increased complexity.

Analog Preprocessing/Demodulation and Post Processing - Each entire candidate system includes the following sections:

1) The analog preprocessing - this section includes both band limiting filtering and amplitude compression and expansion. The prefiltering is required to assure that at least four to six sample clock pulses will occur for the highest analog frequency of interest.

   Amplitude compression and expansion are used on most speech digitization systems to increase speech energy in comparison to a constant noise power of the transmission channel and quantizing noise. Compression minimizes the effects of threshold-induced center clipping. Licklider has shown that only a few db of center clipping can reduce word articulation scores by 20%.[5] Companding may be carried out by changing the delta modulator step size in accordance with the input signal level. Reference 6 outlines a delta modulator with a square-root-type compression characteristics, and a matching demodulator with a square law characteristic. The range of compression is 26 db in this system. Companding may be carried out either by analog signal processing or by the transmission of a variable step size reference.[10]

2) Delta modulator -- in considering the several types of delta modulators, the similar components include the comparator or difference amplifier, the sample clock generator, and the encoding pulse generator. The unique components for each specific delta modulator generally involve the demodulation method; the same demodulator would be used at both the transmitting and receiving terminals.



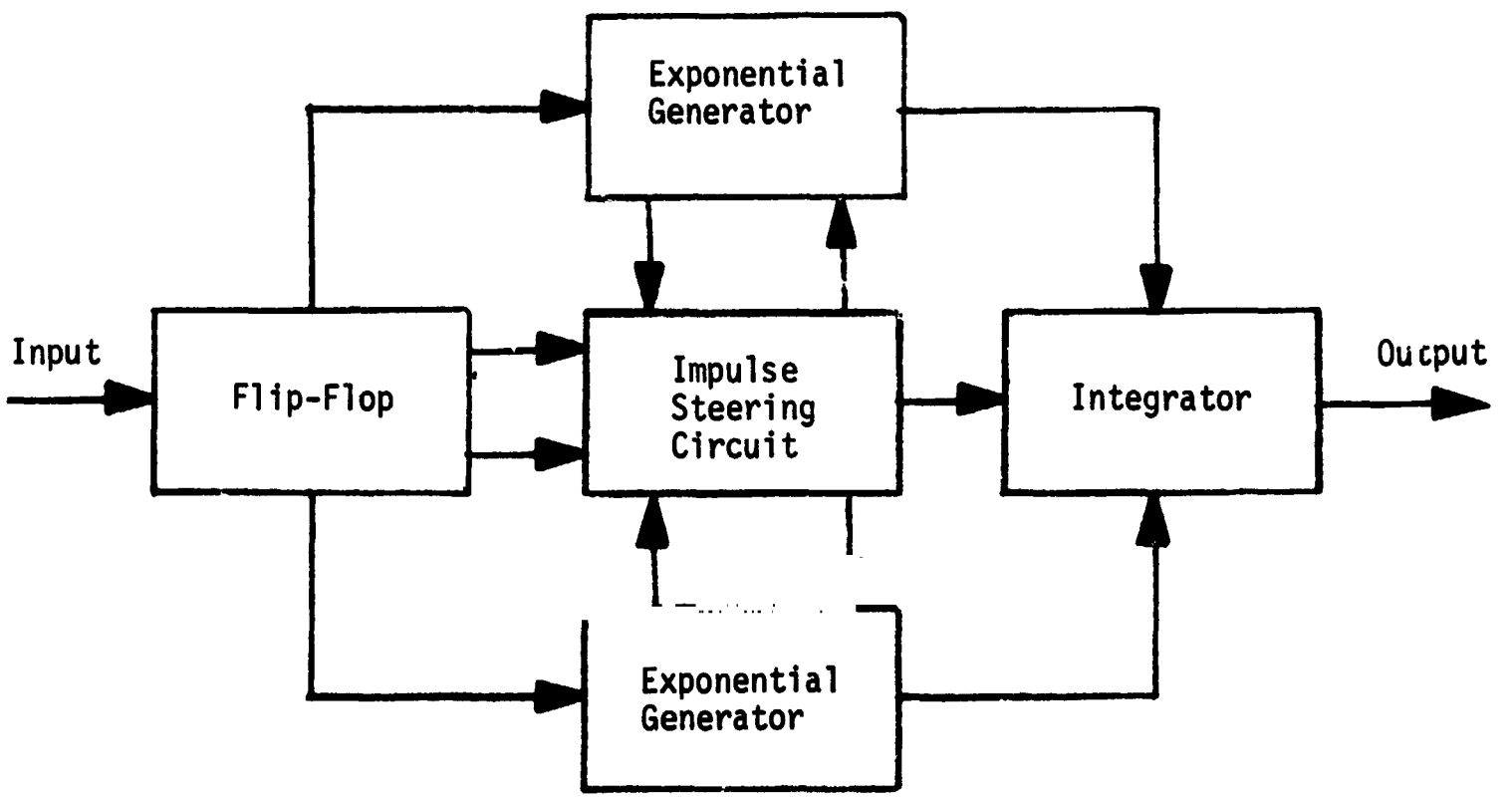


Figure II-13 HIDM Step Function Response and Demodulator Block Diagram

3) Demodulation and post processing -- the receiver demodulator is normally the same as that used at the transmitter; post filtering to reduce idle circuit and quantizing noise for example and to significantly increase intelligibility. The addition of an 18 db per octave 3 KHz post filter increased intelligibility by about 8% over using a 6 db per octave 25 KHz filter (20 Kb/s ample clock).[7] Additionally, a post-process expansion circuit is suggested to compensate for any amplitude compression before modulation.

Conclusions - The actual performance of a delta modulator may differ significantly from that indicated by analyses alone. As Gerber points out, the literature abounds with words like "good," and "high quality" but nowhere can intelligibility data be found.[7] Figure II-14 maps the little evaluation data available on delta modulation intelligibility. In particular, the data and claims of References 2, 4, and 7 in the region of USB application corroborate the selection of 20 Kb/s delta modulation that has been made to a achieve the intelligibility requirements.

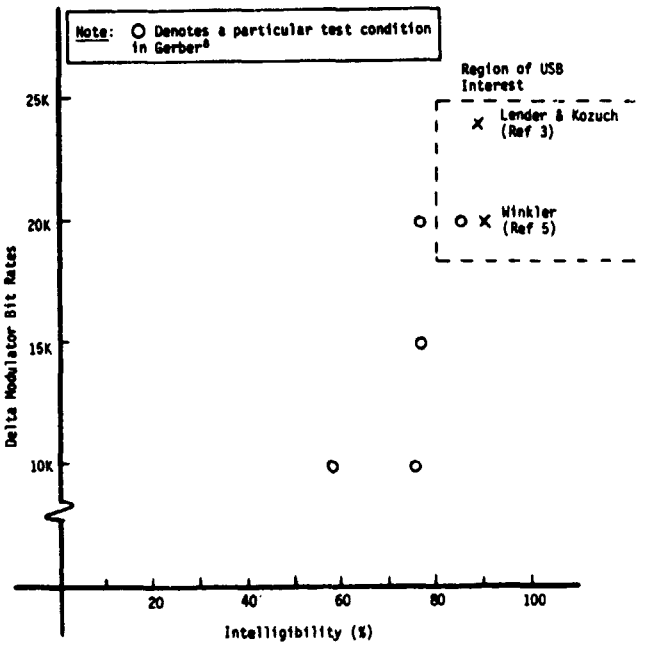


Figure II-14 Experimental Speech Intelligibility vs Delta Modulator Bit Rate

For the 32-times playback voice, a rate of 640 Kb/s should be allocated. When playback voice is not being used the 640 Kb/s allocation might be used to improve the TV quality by allowing a higher picture information transmission rate.

Since the primary objectives of this portion of the study have been met (allocating bit rate and selecting a basic voice coding technique) no additional study in this area is planned.

2. Delta Modulation as a Means of TV Data Compression

In addition to polynomial algorithms, ten types of models of delta modulation as a means of compressing TV data have been studied. These systems include:

1) Single integrator delta modulator (Ref. 12);
2) Double integrator delta modulator (Ref. 12);
3) Double integrator with predictor delta modulator (Ref. 12);
4) Exponential integrator (t = 3fs) delta modulator (Ref. 17);
5) Linear exponential integrator delta modulator (Ref. 18);
6) Multibit "delta modulator" (Ref. 16);
7) Sigma delta modulator (Ref. 13);
8) Companded delta modulator (Ref. 14);
9) Delta-delta modulator (Ref. 15).

We have selected as possible data compression delta modulators the single linear integrator delta modulator with options on the levels of the quantizer within the feedback loop for multilevel or single level values. Predictor delta modulators produce excessive overshoot for pictures which in turn produce contouring. These devices include the double integrator delta, exponential integrators, and the delta-delta modulation techniques, and others.

The delta modulators, though easy to implement at the source, still will require some core memory units for buffering data for frame and line retrace, and for the multiplexing of other data. If core is required, the predictors and interpolators make more efficient use of available memory. In other words, the principle advantage of delta modulation is the absence of core at the source which seems imperative for this high speed mixing and transmission.

The way statistics of pictures are used is not favorable for delta modulators, compared to algorithms that have a large buffering capability. Pictures typically have broad dull regions with patches of detail. The eye focuses on the detail and demands that it be reproduced sharply, if not with perfect brightness fidelity. It is impossible for a system without a buffer to encode this type of source efficiently, and therefore concentration on polynomial algorithms began early in the study.

3. Polynomial Data Compression Techniques

The group of algorithms called "predictors" might more descriptively be called extrapolators because they do in fact predict by extrapolation. The group of algorithms identified as "interpolators" also use prediction when compressing data as a means of determining which pixels are nonredundant. However, in the reconstruction process the interpolators make use of the two widely spaced transmitted values to determine the pixels in between; i.e., reconstructing the redundant pixels by interpolation.

Zero Order Predictor (ZOP) - The ZOP algorithm has two basic forms,* one of which is called "fixed aperture" and the other called "floating aperture." Only the floating aperture method will be described because of its superior performance over the fixed aperture method. The aperture location of this algorithm is based on a prediction that the new pixel value will be the same as the preceding pixel value as shown in Figure II-15. The new pixel value is compared with the predicted value and if the difference is no more than the allowable tolerance, the pixel is considered redundant and dropped. If, however, the difference is greater than the allowable tolerance, the new pixel value is tagged and sent to the buffer for transmission.

First Order Predictor (FOP) - The FOP is much the same as the ZOP floating aperture. In this case, however, the predicted value lies on a straight line drawn through two preceding points.† Here again, if the difference between the predicted pixel value and the new pixel value is less than the allowable tolerance the pixel is eliminated; if the difference exceeds the limit, the pixel is tagged and sent to the buffer which is shown in Figure II-16.

Predictor Reconstruction - Description of reconstruction of the zero order and first order predictor algorithms is exactly the same. Reconstruction proceeds on a point by point basis from left to right by drawing straight lines between adjacent pixel values.

---

*W. A. Stevens: *Data Compression Concepts and Philosophies.* American Astronautical Society, Rocky Mountain Resources for Aerospace Science and Technology, July, 1967.

†J. E. Medline: *The Comparative Effectiveness of Several Telemetry Data Compression Techniques.* International Telemetry Conference, 1963.

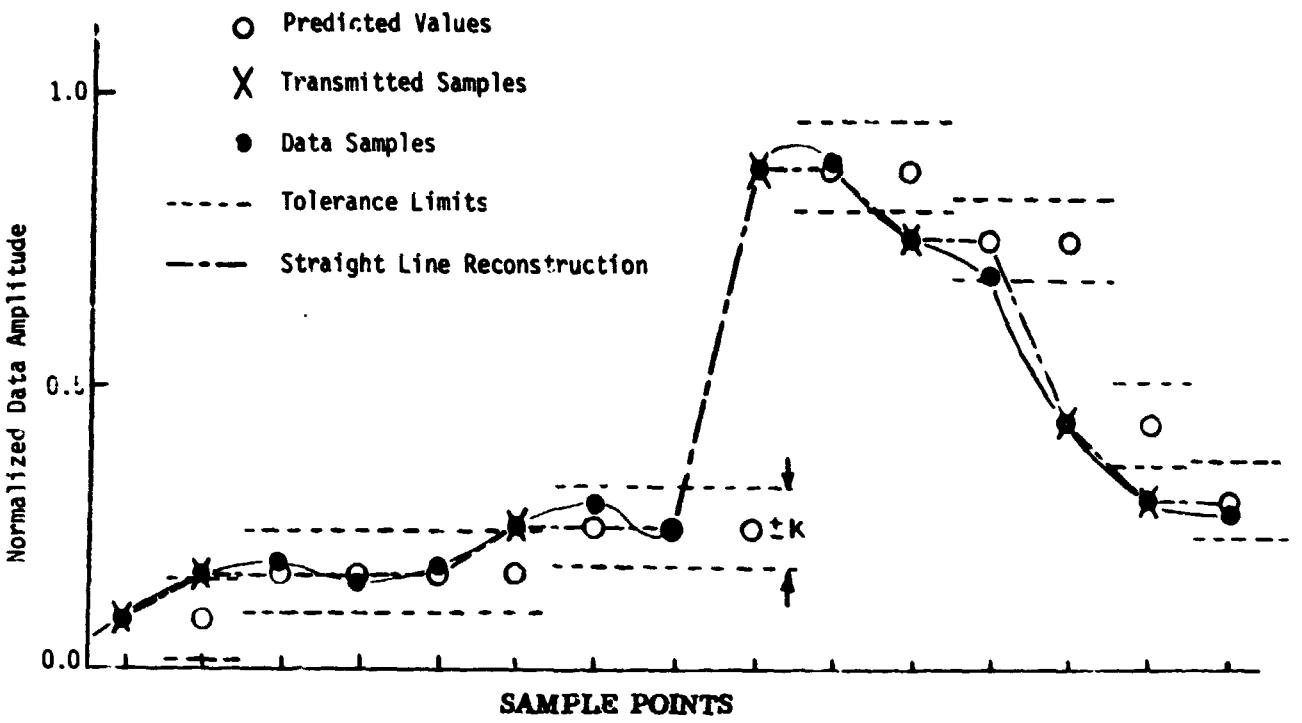


Figure II-15 Zero Order Predictor, Floating Aperture

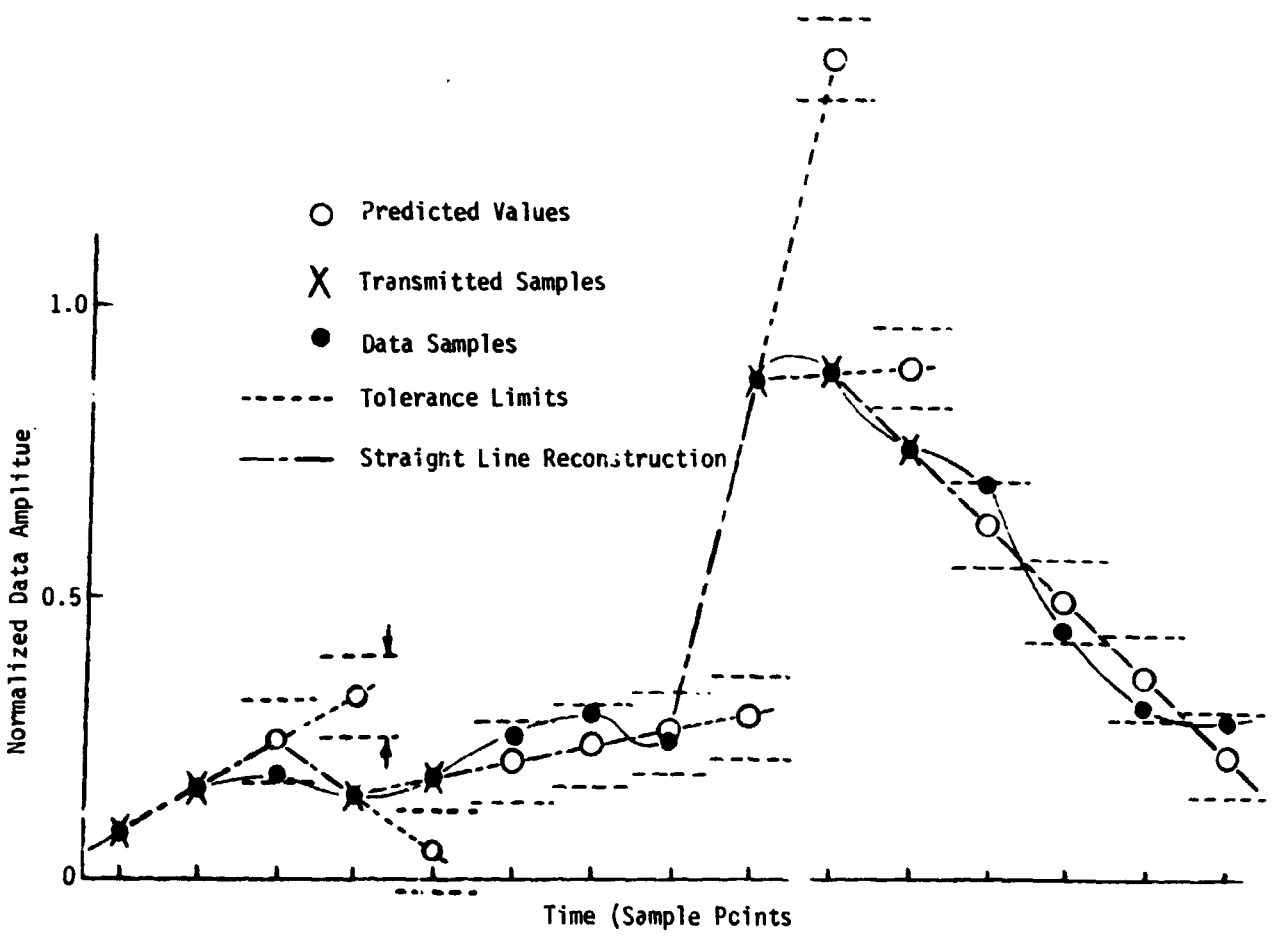


Figure II-16 First Order Predictor

Interpolators - The interpolator group of algorithms works on an "after the fact" type of curve fitting basis. Several types of interpolators can be used depending on the error criteria desired, e.g., peak, RMS or average error. The interpolator differs from the predictor in that the reference from which the tolerance is measured may change according to trends in the pixel values since the last transmission. At such time as a pixel value is outside the predicted tolerance band a value equal to the calculated reference is transmitted. To demonstrate this difference the operation of a zero order interpolator with peak error criteria, Figure II-17, is described next.

Zero Order Interpolator (ZOI) - Each new pixel is successively checked for a maximum or minimum value since the last transmission. When either of these values is exceeded by a new pixel value, the new maximum-minimum difference is compared to 2K. If the difference is less than 2K the pixel is redundant. A difference of greater than 2K results in the calculation of the maximum-minimum average. This average is then tagged with the time of the last "in tolerance" pixel and sent to the buffer. The new series of peak error comparisons begins with the last "out-of-tolerance" pixel value. In summary, the tolerance band of +K is not tied to a predetermined value, but is allowed to float so as to include all of the previous pixels which will be within +K of the last transmitted value. Using the peak error criteria the transmitted value is then a point which best represents the amplitude of all pixels since the last transmission which, of course, is the average of the maximum and minimum values. The time tag placed on the transmitted value corresponds to the time of occurrence of the last in-tolerance pixel.

The point by point graphical reconstruction of the data when this algorithm is used is best accomplished by drawing straight lines from transmitted values toward the left. As long as no value was transmitted at a specific sample time, it is known that the pixel value did not change.

First Order Interpolator - The FOI approximates the function with a straight line drawn between data points as far apart as possible so that no data points in between are more than a specified tolerance above or below the line. It is illustrated in Figure II-18. Reconstruction is by drawing straight lines between transmitted points.

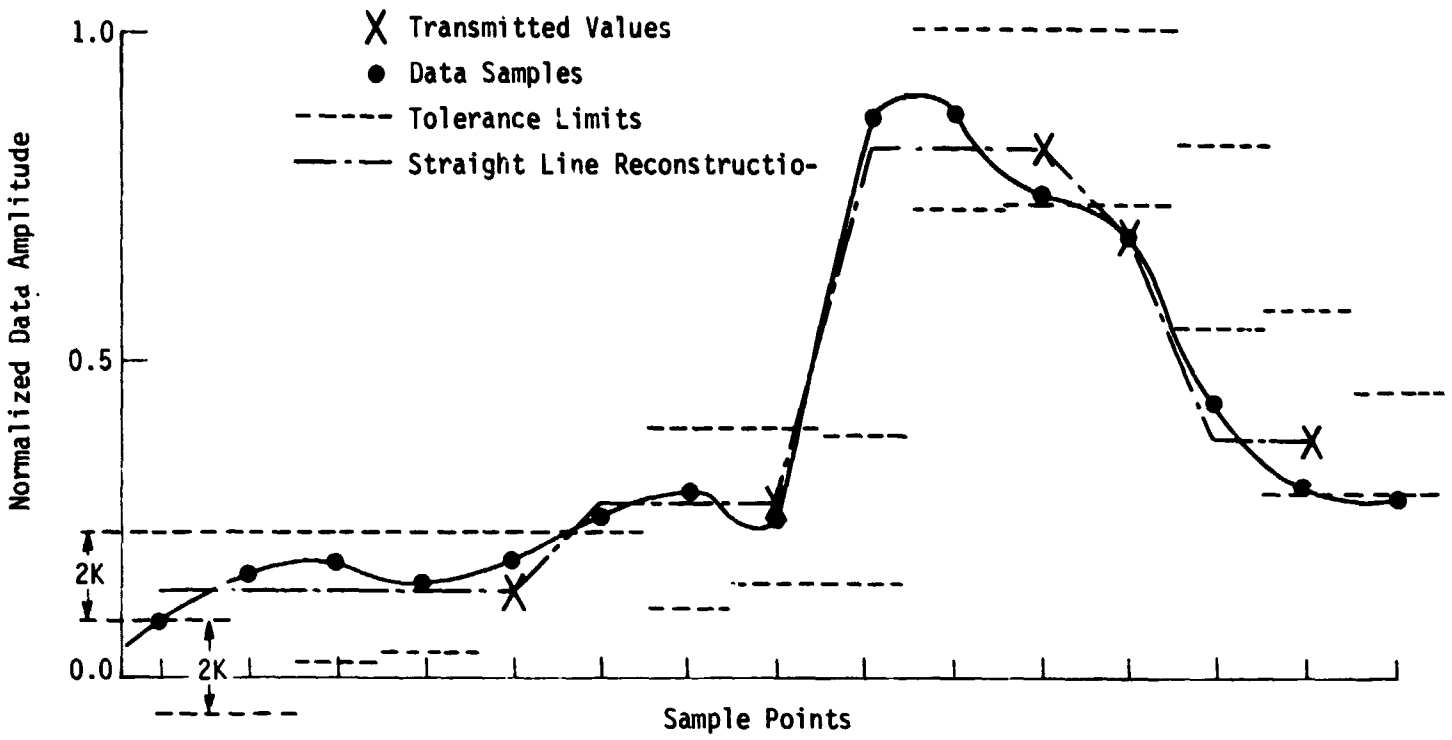


Figure II-17 Zero Order Interpolator (Peak Error Criteria)

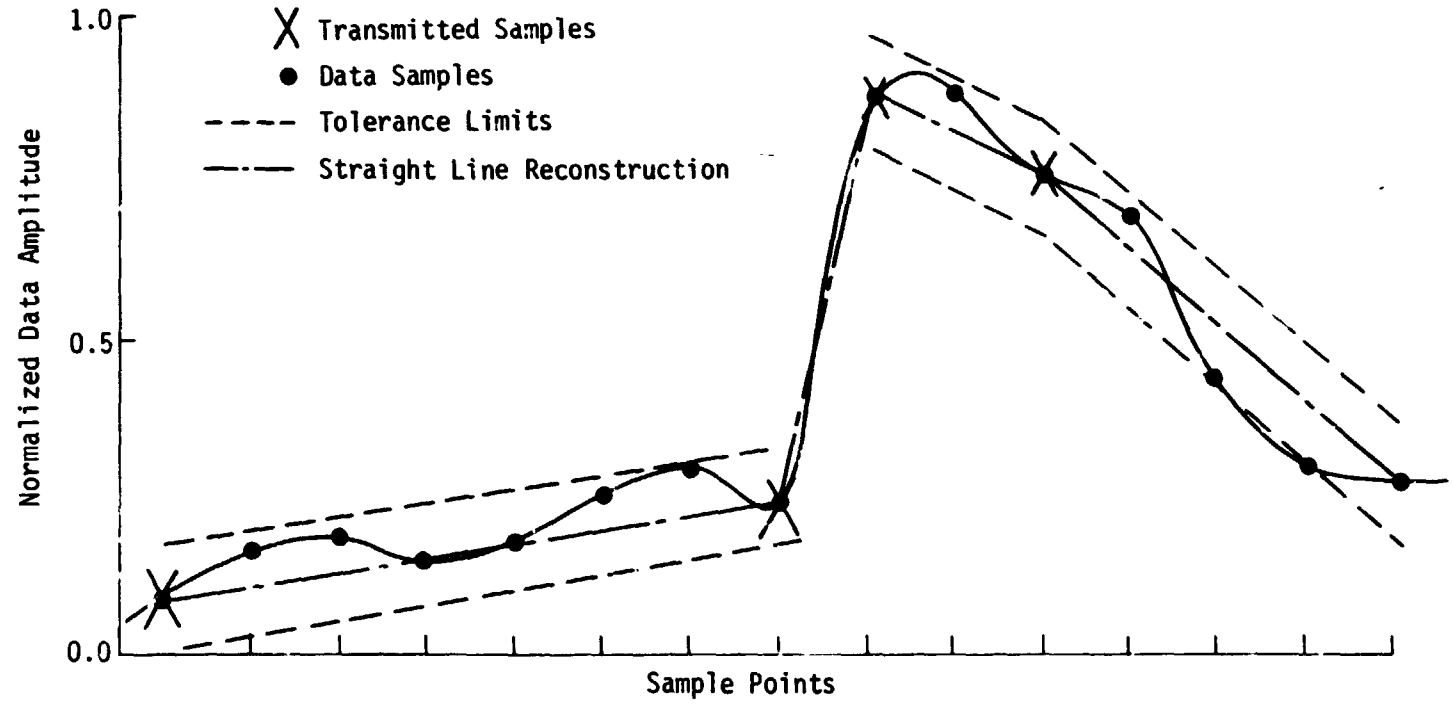


Figure II-18 First Order Interpolator

Run Length Coding - In their ordinary form, w. each of the polynomial algorithms, the transmitted message consists of a sequence of words each of which has a gray level code and a run length code. The run length tells the receiver how many pixels in a row are to be painted with the gray level. Short runs are more frequent, and it is efficient to use a code that has short words for short run lengths and long words for long runs.

If we assume that run length, probability distribution for our video data follows the equation

$$P(RL) = Pq^{RL-1},$$

where q = 1 - p and RL is the run length, we can use an efficient run length code.* From histograms made during the compression evaluation runs, it is apparent that the run length probability distribution does in fact follow this equation.

The run length coding consists of a base word of "m" bits plus overflow bits followed by a comma bit. For example, if the average run length is 4, a base code of 2 bits (i.e., m = 2) would be selected. For a run length of 7, the code word would read a base code of 11 followed by one overflow bit (1), and terminated by a comma bit (0) (i.e., 1110). Again, for m = 4 and a run length of 37, there would be 4 bits base code plus 2 overflow bits and a comma, resulting in a complete code word of 0101110.

In our case, where we want an average run length of between 3 and 4 the optimum run length code consists of a base code of m = 2.† This code word configuration was used for all of our simulations.

---

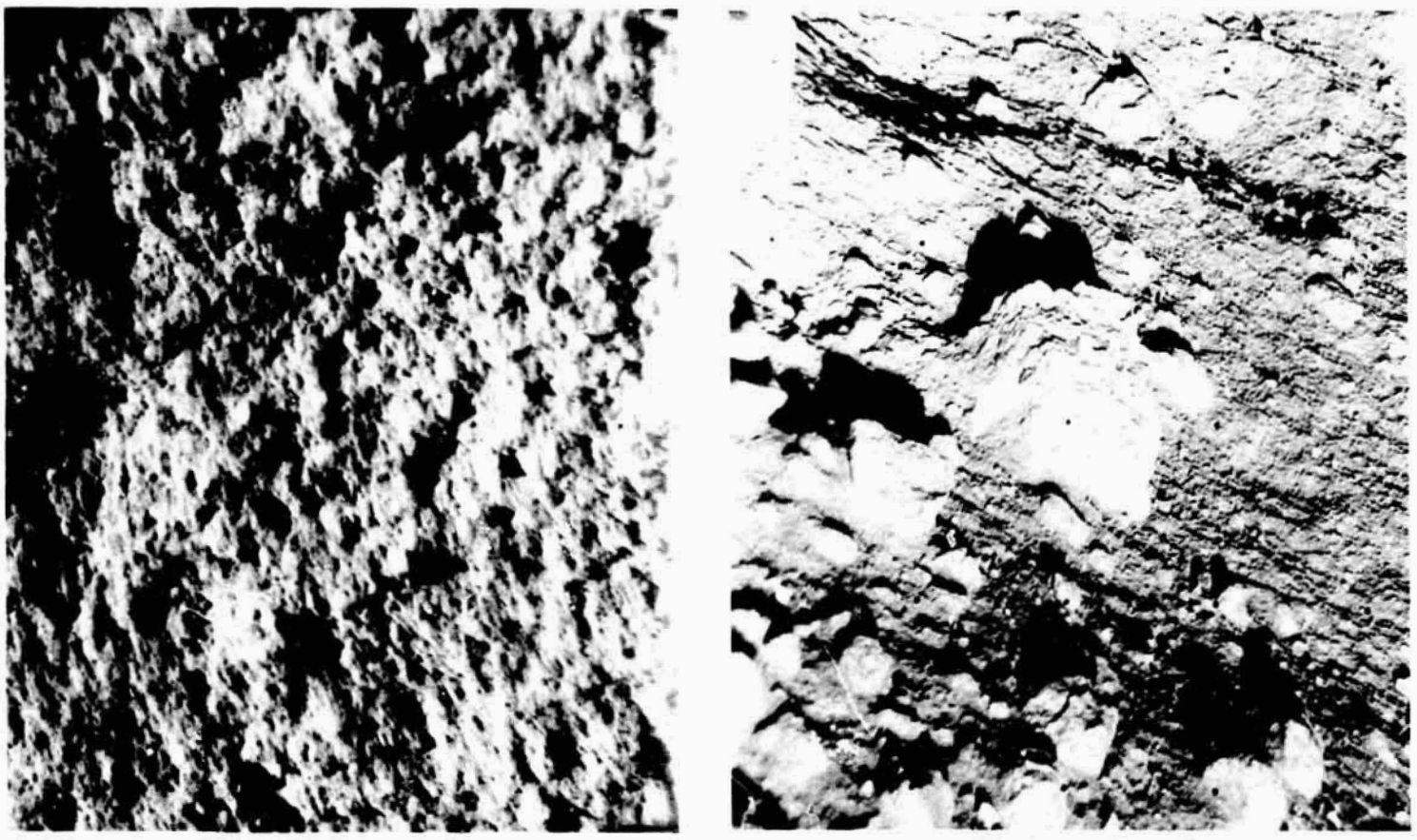
*C. Cherry: *An Experimental Study for the Possible Bandwidth Compression of Visual Image Signals.* Proceedings of IEEE, November 1963.

†S. A. Sheldahl: *Channel Identification Coding for Data Compressors.* EASCON Convention Record, 1968.

4. Preliminary Test of Polynomial Algorithms

For a preliminary test of these compression algorithms, two digital Surveyors moon scenes of uniform high data activity were used as shown in Figure II-19. Figures II-20 and II-21 show performance plots for the ZOP, ZOI, FOP, and FOI algorithms. The abscissa of these plots is given in bits/pixel. The term bits/pixel was chosen so that the number of quantization levels of any one picture could be varied and yet have the results directly related from one picture to the other. This is not possible when using compression ratio as the abscissa. Bits/pixel is found by dividing the total number of coded non-redundant pixel bits by the number of original pixels.

When examining the results it should be remembered that these two pictures are very active and are not typical video pictures. The runs were made with fixed apertures, as noted, and used no buffer control. From earlier runs it was noticed that the signal-to-noise ratio and bits/pixel measurements did not change significantly after the first 15 lines of Pictures 1 or 2. Therefore, the runs represented in Figures II-19 and II-20 terminated at the end of 15 lines.



Picture No. 1
$H_5^* = 3.65$

Picture No. 2
$H_7^* = 4.67$

Figure II-19 Digital Surveyor Moon Scenes

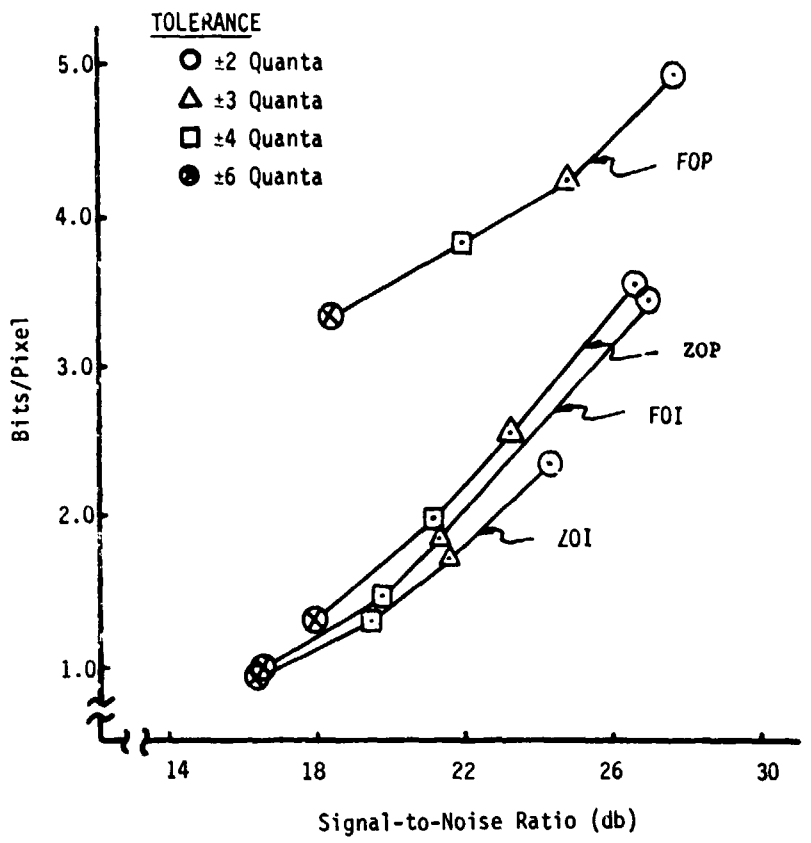


Figure II-20 Algorithm Comparison (Picture 1, 6-Bits)

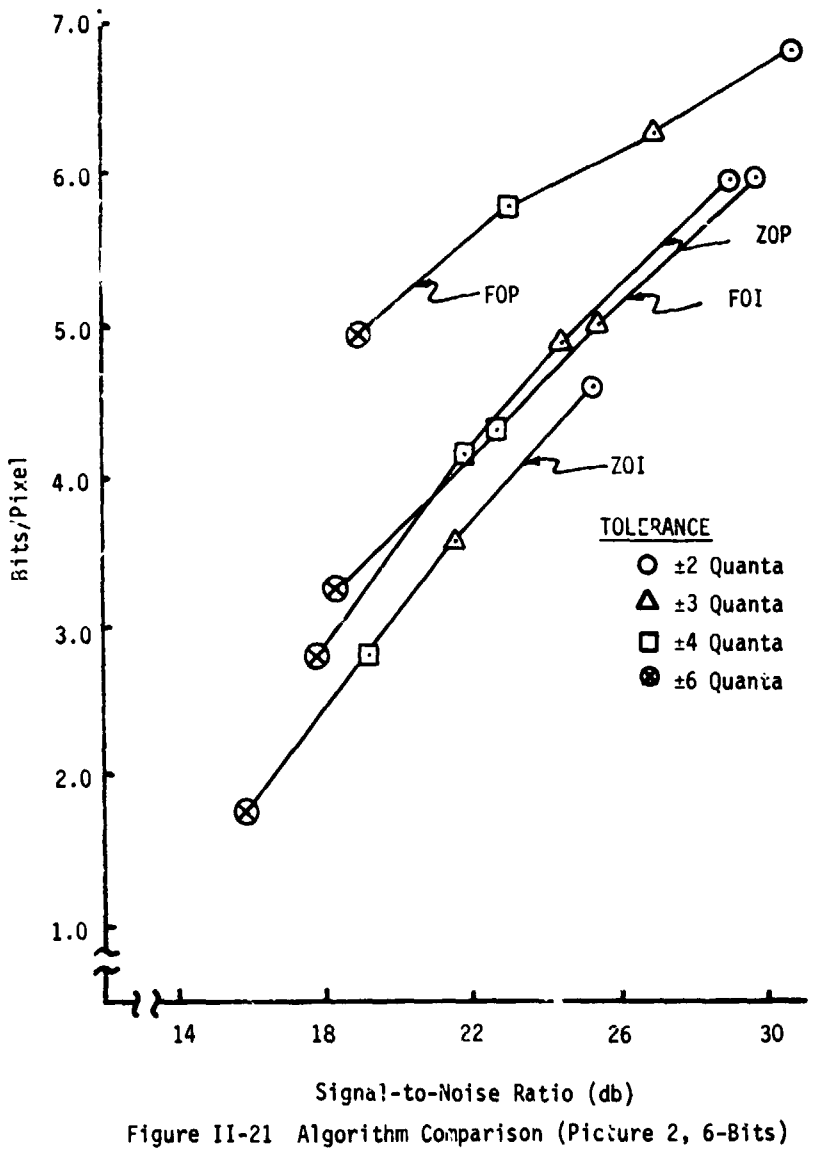


Figure II-21 Algorithm Comparison (Picture 2, 6-Bits)

Both figures show that the ZOI algorithm performed the best by a small margin and the FOP algorithm performed much poorer than the other three algorithms. The comparative performance of the FOI algorithm was somewhat better for the less active picture, Picture 1, than it was for Picture 2.

As expected all of the algorithms performed considerably better on Picture 1 than on Picture 2. In general, this was done with little sacrifice in error signal-to-noise ratio.

In most previous studies, the first-order predictor algorithm has shown rather poor performance; results confirm this.

5. Line-to-line Dither

The criteria of picture quality for TV are not such things as rms error, resolution, and numer of distinguishable gray levels, which are useful because they are fairly well defined and not too hard to measure. The ultimate criteria is whether the picture is satisfactory to the viewer.

A study of the capabilities of the eye cannot give the whole answer either, but it does suggest some approaches to reducing the number of bits that are transmitted without appreciable subjective degradation of the TV image.

In particular, large differences between adjacent lines and between subsequent frames can be detected by the eye, but small differences will not be seen, and the eye will tend to see the average gray level. This effect can be exploited by two kinds of dither as explained below.

a. Gray-Level Dither - To avoid contouring, it is desirable to quantize to a large number of levels, but this increases the number of bits that must be transmitted. Gray-level dither can reduce the contouring without increasing the bit rate; but, if applied too ambitiously, visible flicker will result.

Gray-level dither can best be explained by describing a particular case. Assume that initial quantization is to 6 bits or 64 levels from 0 to 63. We wish to transmit 3 bits of gray level per pixel, so we simply round off to three significant bits. A gray level of 13, binary code 001101, will be rounded to 16 and transmitted as 010. The receiver adds three zeros to make 010000 and puts gray level 16 on the screen. This process will produce noticeable contouring on the flat portions of the picture since the reproduced gray level jumps from 8 to 16 where the original gray level goes from 12 to 13.

To give the effect of finer quantization and soften the contours, the next field (which is interlaced with the first one) is treated differently. Four is added to each gray level before rounding. Thus, 13 + 4 = 17, 010001, which is again rounded to 16. This time, however, the receiver subtracts 4 and puts 12 on the screen. The contours in this field occur midway between those in the previous one; the transition from 4 to 12 occurs where the original goes from 8 to 9, and the transition from 12 to 20 is where it changes from 16 to 17.

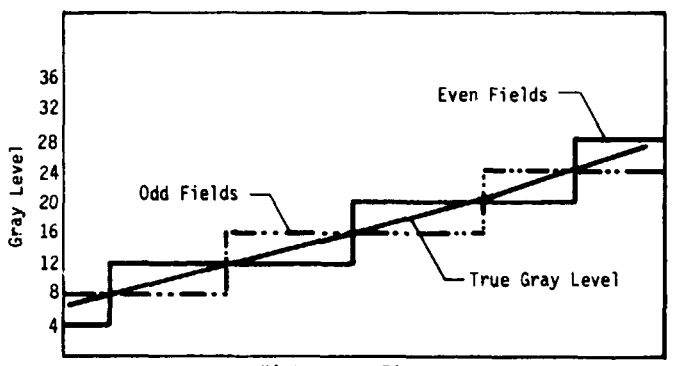


Figure II-22 Superposition of Fields with Gray-Level Dither

Figure II-22 shows how a slowly changing gray level would be reproduced in odd and even fields. The eye tends to average the gray levels of adjacent fields and gives the effect of twice as many quantization levels.

The dither scheme described uses only two different quantizing rules: it rounds to the nearest multiple of 8, or to the nearest odd multiple of 4 (i.e., 4, 12, 20, 28, etc). More elaborate dither schemes can also be used, with more quantizing rules. These require a large number of fields and a longer time to complete the cycle. If the cycle time is too long, the eye will average the brightness, and flicker will be apparent.

To minimize the flicker, the low-frequency components of the gray level versus time function should be kept small. For an 8-level dither, the amount added before rounding could be increased by 3 for each new field. If the original gray level is 13, the reproduced level will go through the cycle 16, 13, 10, 15, 12, 17, 14, 11, to give a relatively small low-frequency components.

b. Horizontal Dither - One way to reduce the required bit rate for TV is to increase the horizontal distance between sampling points. The resolution is reduced and the picture becomes grainy as the separation of the sampling points is increased. However, some of the loss can be regained by using horizontal dither.

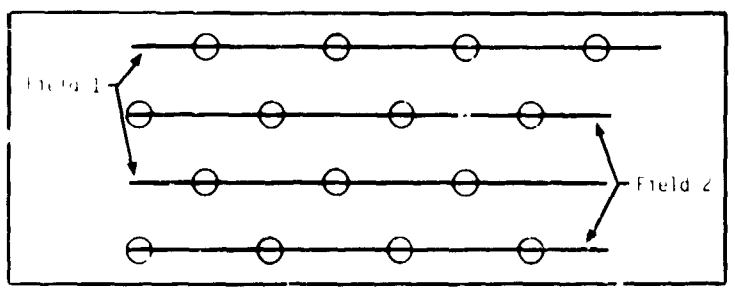


Figure II-23 Staggered Sampling Points in Successive Fields

The simplest type is shown in Figure II-23 where the sampling points are staggered to get a more uniform coverage of the original picture.

The practical limit of horizontal dither as of gray-levels dither, will depend on the flicker sensitivity of TV viewers. All of our compression simulations described in the following section use two levels as shown in Figure II-23.

6. Simulation Results

Figures II-24 and II-25 show the results of processing a typical picture with variations of the zero and first-order algorithms. The original picture has 512 x 512 pixels with 6 bits quantization. Figure II-24 shows the distribution of gray levels.

Each compressed picture is reproduced from about 240,000 bits. Horizontal dither is used in each compressed picture so that 256 pixels are used in each line staggered as previously described.

A subjective evaluation of the pictures with 5 and 6 bits quantization by a naive observer ranked them as shown in Table II-4.

Table II-4 Subjective Ranking of Compressed Pictures (Best Picture is Ranked 1)

ALGORITHMS	QUANTIZATION	INTENSITY DITHER	RANK
ZOI	6		5
	5		3
	5	Yes	4
ZOF	6		9
	5		6
	5	Yes	7
FOI	6		8
	5		1
	5	Yes	2

The pictures ranked 8 and 9 have streaks caused by the last-ditch buffer control mode in which the run length is forced to be at least 6. The two unstreaked FOI pictures are rated best followed by the ZOI and ZOP. The choice of FOI as best seems to be based on the absence of contouring, in spite of the fact that some detail is lost that is preserved in the zero-order algorithms.

It is disappointing that the pictures with intensity dither ranked below the corresponding pictures without. The difference is admittedly slight. but we feel that intensity dither does improve the picture at least for 3 and 4 bits of quantization.

7. Results and Conclusions

The results of the compression simulation show that good pictures can be transmitted with 240,000 bits per frame or 7.2 X 10^6 bits/sec. The only noticeable degradation of the pictures is contouring with the ZOI and ZOP algorithms, and loss of low contrast detail with FOI. Detail with contrast equal to more than a few gray levels is reproduced with resolution equal to that of the original digital picture. FOP is apparently not a competitive method.

Five bits of gray level quantizing (32 levels) is adequate for the test picture that we used. The histogram (Figure II-24) shows that all but 4 levels, out of the original 64, are used. Poor adjustment of the A/D interface could result in fewer active levels, and increase contouring. A simple level-and-gain control circuit could be incorporated in the operational equipment to compensate for changing light levels and contrast.

The small benefits of intensity dither are disappointing. The reason is that contouring is not caused primarily by the size of the quantizing steps but rather by the size of the tolerance in the zero order algorithms. The smallest tolerance is ±1-level while the maximum quantizing error is ½-level. To exploit the closeness of the TV lines, a modified algorithm should be devised. This is discussed in the following section.

Buffer control was satisfactory except that the last ditch mode, used when the buffer was nearly full, produced easily visible flaws. In this mode the run length is forced to be at least six. The effect can be seen on the FOP picture and near the bottom of the 6-bit ZOP picture. While this mode is effective in positively preventing overflow; other methods, including some mentioned in the following section, should be tried.

A forced minimum-run-length is unsatisfactory for a last ditch method of preventing buffer overflow, and something better should be worked out. One possibility is to delete a line and send a special code word to order the repetition of the preceding line.

Our work so far has been with a linear quantizing scheme. That is, each gray level represents an equal increment of light intensity. It is generally considered that a roughly logarithmic relationship is better, giving smaller steps in the darker portions of the picture.

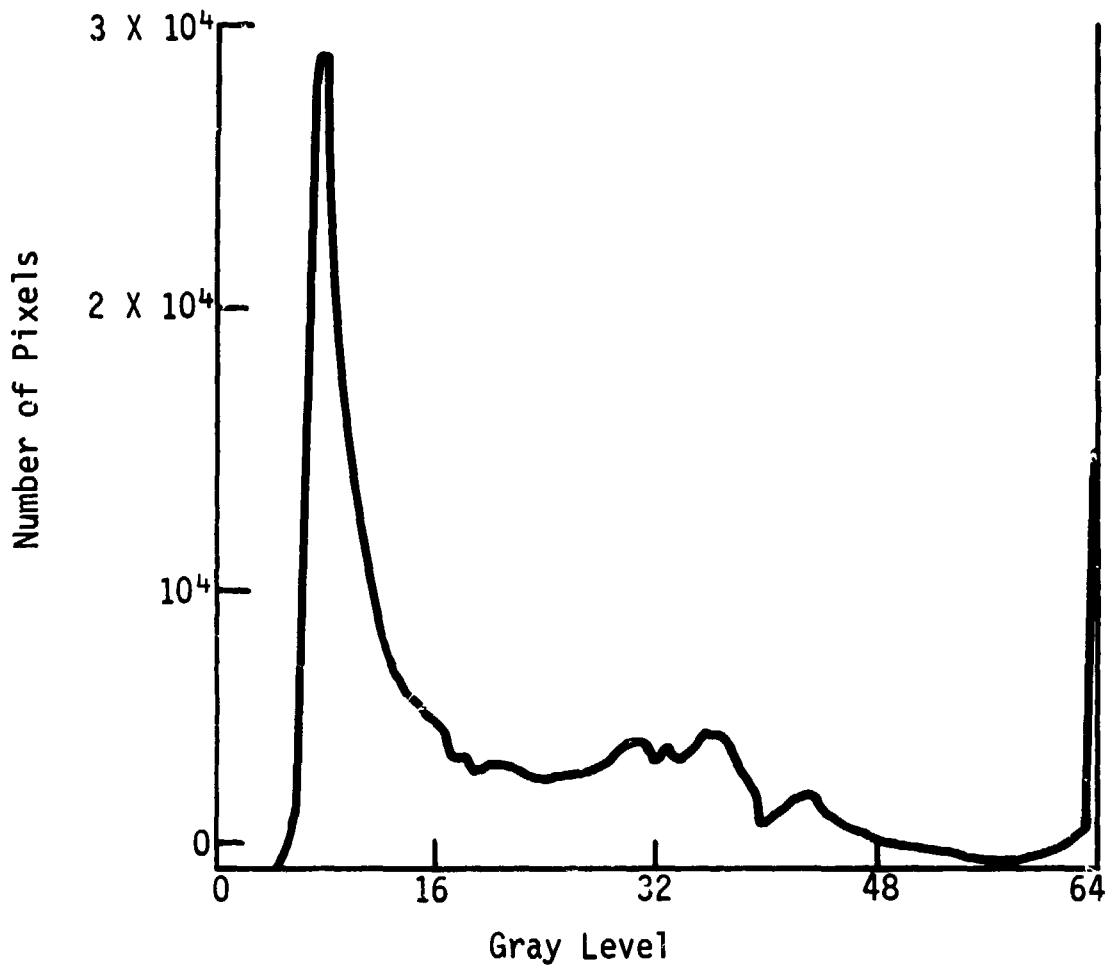
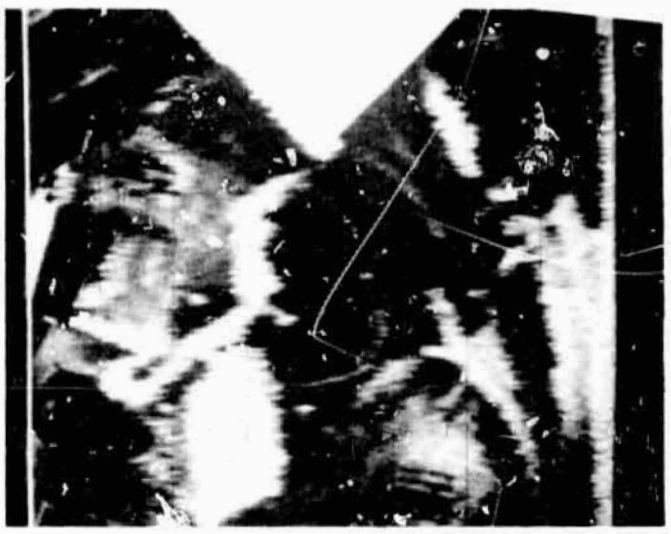


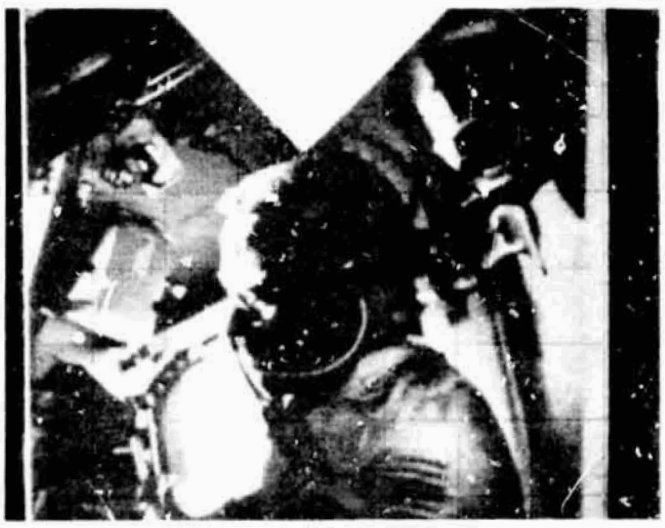
Figure II-24 Histogram of Test Picture

Original Picture
6 Bits/Pixel, 512 Pixels/Line,
512 Lines/Frame

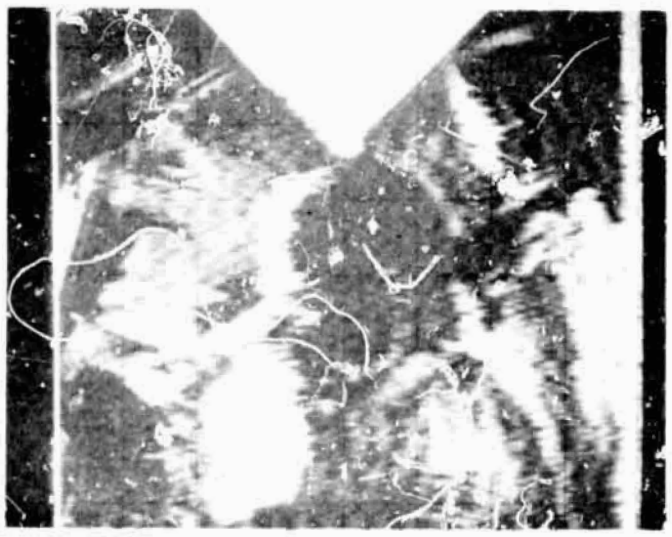


First-Order Predictor
6 Bits, Horizontal Dither

First-Order Predictor
5 Bits, Horizontal Dither



Best Picture by Consensus
First-Order Interpolation
5 Bits/Pixel, Horizontal Dither,
Intensity Dither

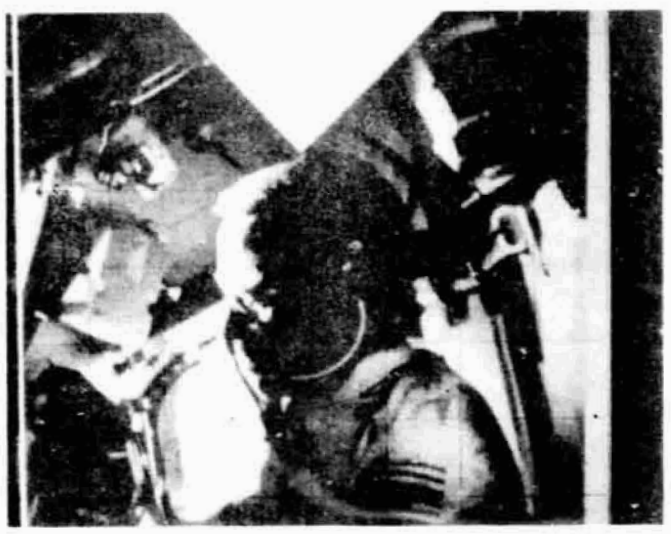


First-Order Predictor
5 Bits, Horizontal and
Intensity Dithers

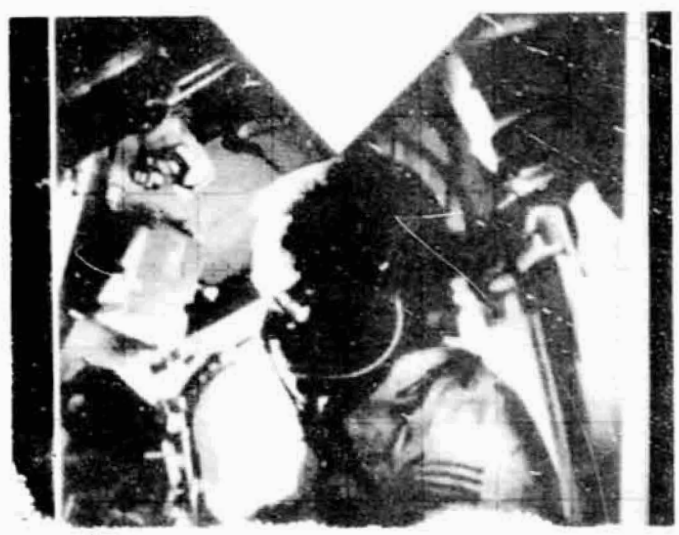
Figure II-25 Typical Picture with Zero and First-Order Comparisons

Zero-Order Interpolator
6 Bits/Pixel, Horizontal Dither

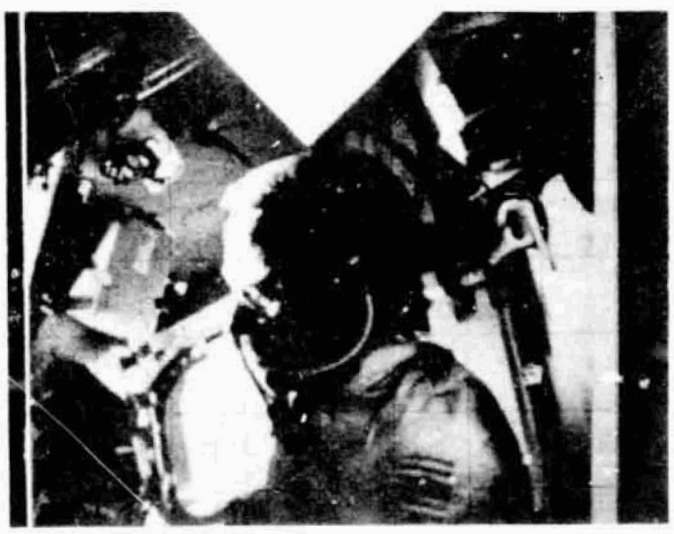
Zero-Order Predictor
6 Bits/Pixel, Horizontal Dither



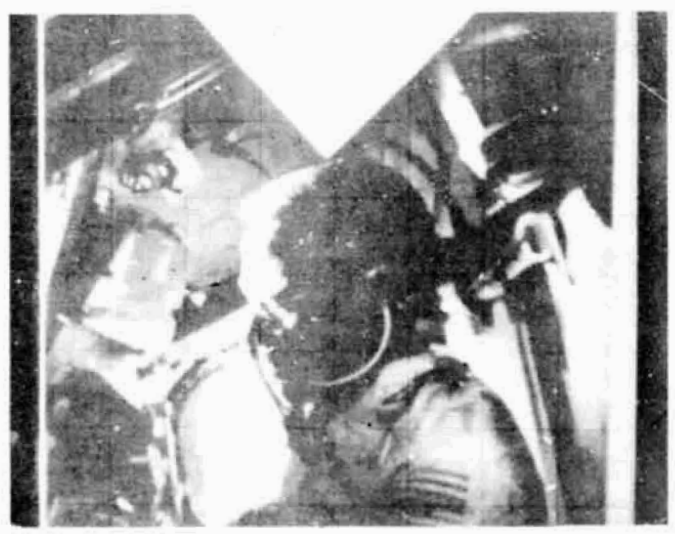
Zero-Order Interpolator
5 Bits/Pixel, Horizontal Dither



Zero-Order Interpolator
5 Bits/Pixel, Horizontal
and Intensity Dithers



Zero-Order Interpolator
6 Bits/Pixel, Horizontal Dither



Zero-Order Predictor
6 Bits/Pixel, Horizontal
and Intensity Dithers

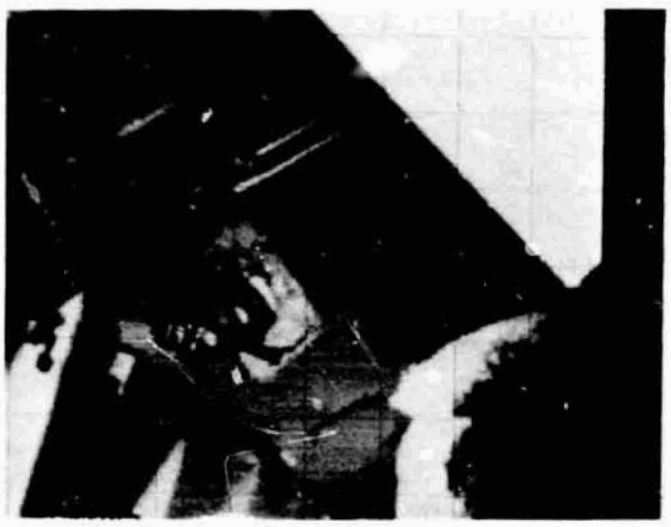
Figure II-25 (cont)

Zero-Order Interpolator
6 Bits/Pixel, Horizontal Dither
Digital Enlargement (Upper LH Corner)

Zero-Order Predictor
6 Bits/Pixel, Horizontal Dither
Digital Enlargement (Upper LH Corner)

Zero-Order Interpolator
5 Bits/Pixel, Horizontal Dither
Digital Enlargement (Upper LH Corner)

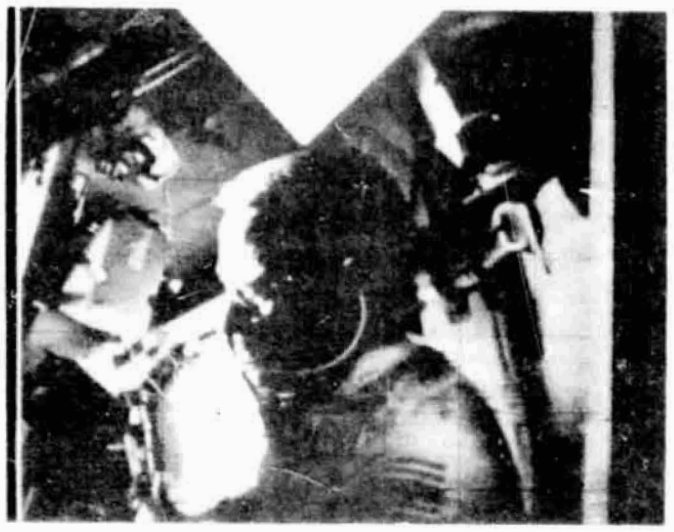
Zero-Order Interpolator
5 Bits/Pixel, Horizontal and
Intensity Dithers Digital
Enlargement (Upper LH Corner)



Zero-Order Predictor
5 Bits/Pixel, Horizontal Dither
Digital Enlargement (Upper LH Corner)

Zero-Order Predictor
5 Bits/Pixel, Horizontal and
Intensity Dithers Digital
Enlargement (Upper LH Corner)

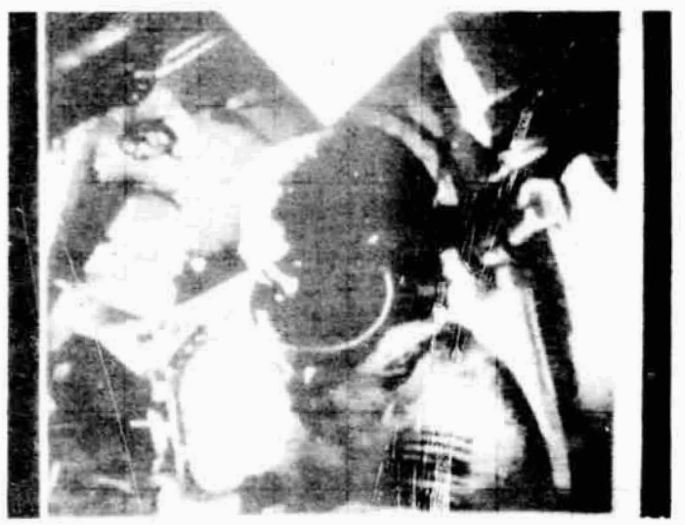
Figure II-25 (cont)



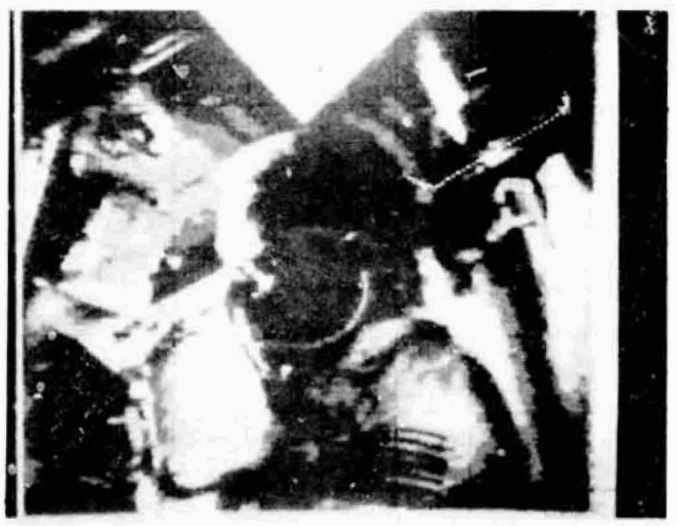
First-Order Interpolator
6 Bits/Pixel, Horizontal Dither

First-Order Interpolator
5 Bits/Pixel, Horizontal Dither

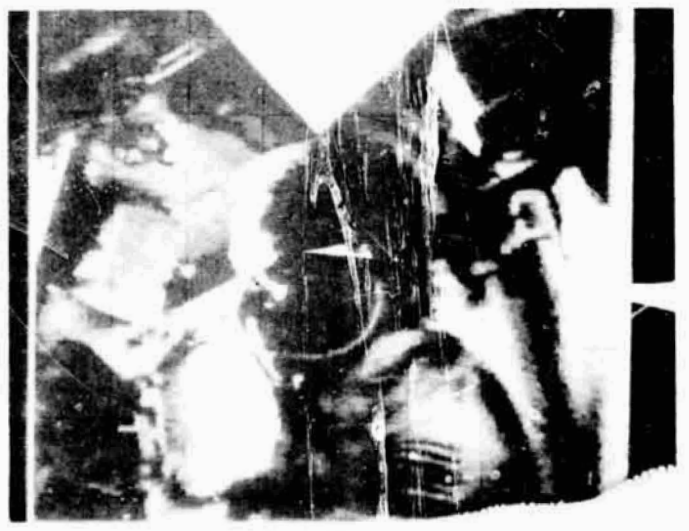
Zero-Order Interpolator
4 Bits/Pixel, Horizontal Dither



Zero-Order Interpolator
4 Bits/Pixel, Horizontal
and Intensity Dithers



Zero-Order Interpolator
3 Bits/Pixel, Horizontal Dither



Zero-Order Interpolator
3 Bits/Pixel, Horizontal
and Intensity Dithers

Figure II-25 (concl)

8. Recommendations for Continued Study

The FOI algorithm avoids contouring, but it is not enough better than the ZOP and ZOI to justify the greater hardware complexity. Therefore, we recommend that further work be done on zero order algorithms with the objective to be increasing the compression ratio while maintaining quality.

A decrease of about 10% in the required bits, with no reduction in quality, can be achieved by transmitting changes in gray level instead of the levels themselves; and using an efficient variable-word-length code similar to the one we have used for run lengths. This will save about 1-bit per word over the present scheme. Both the run-length and the gray-level difference codes should be tailored to the actual statistics of the expected subject matter. A slight improvement might be realized by modifying our run-length code to fit the actual distribution of run lengths in typical pictures. To get the advantage from intensity dither, large quantizing intervals (few levels) should be used with a small tolerance. If the tolerance is 0, small fluctuations that carry the intensity signal back and forth across a quantizing boundary will require many bits to be transmitted without conveying useful information. To avoid this difficulty, we can introduce some hysteresis in the A/D convertor so the gray level is not changed until the intensity signal has moved some selected voltage beyond the quantizing boundary, or has remained in the new quantizing zone for some selected time. The selected voltage and time can be varied to control the buffer queue in the same way that the tolerance is varied in the present programs.

9. Section B References

1. *State of the Art, Speech Compression and Digitization.* ITT Communications Systems, Report ESD-TDR-64-460, (AD 451 036L). Paramus, New Jersey, September 1964.

2. A. Lender and M. Kozuch: "Delta Modulating Systems." *Electronics*, November 17, 1961, pp 125-129.

3. Hircshi Inose, Yasuhiko Yasuda, Junzo Murakami and Hiro. Fujita: "New Modulation Technique Simplifies Circuits." *Electronics*, January 25, 1963, pp 52-55.

4. Marion R. Winkler: "High Information Delta Modulation." *IEEE International Convention Record*, pt. 8, 1963, pp 260-265.

5. J. C. R. Licklider: "Effects of Amplitude Distortion Upon the Intelligibility of Speech." *Journal of the Acoustical Society of America*, Vol 18, No. 2, October 1946, pp 429-434.

6. Atsushi Tomozawa and Hisashi Kaneko: "Companded Delta Modulation for Telephone Transmission." *IEEE Transactions on Communication Technology*, Vol COM-16, NO. 1, February 1968, pp 149-156.

7. Sanford E. Gerber: "The Intelligibility of Delta Modulated Speech." *IEEE Transaction on Audio and Electroacoustics*, Vol AU-14, No. 2, June 1966, pp 93-96.

8. H. Inose, Y. Yasuda, and J. Murakami: "A Telemetering System by Code Modulation – $\Delta - \varepsilon$ Modulation." *IRE Transaction on Space Electronics and Telemetry*, September 1962, pp 204-209.

9. Hiroshi Inose and Yasuhiko Yasuda. "A Unity Bit Coding Method by Negative Feedback." *Proceeding of the IEEE*, November 1963, pp 1524-1535.

10. Stephen J. Brolin and James M. Brown. "Companded Delta Modulation for Telephony." *IEEE Transactions on Communication Technology*, Vol COM-16, No. 1, February 1968, pp 157-162.

11. Robert H. Maschoff: *Delta Modulation.* Electro-Technology, January 1964, pp 91-97.

12. A. Lender and M. Kozuch: "Delta Modulating Systems." *Electronics*, November 17, 1961.

13. H. Inose, Y. Yasuda, and J. Murakami: "A Telemetering System by Code Modulation - Δ - ε Modulation." *IRE Trans. on Space Electronics and Telemetery*, September 1962.

14. A Tomozawa and H. Kaneko: "Companded Delta Modulation of Telephone Transmission." *IEEE Transactions on Communication Theory*, Vol COM-16, NO. 1, February 1968.

15. J. Das and P. Chatterjee: "Optimized Δ - Δ Modulation System." *Electronics*, Vol 3, No. 6, June 1967.

16. H. Weg: "Quantizing Noise of a Single Integration Delta Modulation System with an N-Digit Code." *Philips Res. Rpt. 8*, 1953, pp 367-385.

17. J. Holzer: *Exponential Delta Modulation for Military Communications.* Final Report, Proj. No. 3-99-12-021, Signal Corp. Task No. 132A, June 1956.

18. R. Cotton, et al.: *Demonstration of the Feasbility of Using Delta Modulation for Pictorial Transmission.* Final Report, Contract AF33(657)-8478, Philco Advanced Technology Lab., ASD Tech Doc. Report ASD-TDR-63-528, Blue Bell, Pennsylvania, May 1963.

C. PERFORMANCE COMPARISON

For this application, certain ground rules must be established to compare the performance of a digital system with the existing analog system. The great bulk of transmitted information is in the TV picture, and any comparison will be based principally on the ability to transmit a good picture with a minimum signal-to-noise ratio in a given band. The FM system gives an acceptable picture when the video signal-to-noise ratio is greater than 22 db, and the video band is 2.5 MHz.

Our compression studies show that equal resolution and good picture quality can be achieved with ZOI image compression using 270 pixels/line and an average of 1.8 bits/pixel. However, decoding errors will produce several types of blemishes in the image. More than half of the errors will occur in the gray-level code and will produce errors in the brightness of a few pixels, the number depending on the particular run length. Errors occurring in the run-length codes will affect the rest of the line if the error inserts or removes a comma that indicates the start of the next gray-level run-length pair.

Much less frequently, an error will destroy an end-of-line code or produce a false end-of-line code. This type of error will usually make the rest of the field one line high or low. A very rare occurrence is that of an error in the end-of-field (or frame) code, which will usually produce a vertical error of several lines in the next field.

To simplify the comparison, it is assumed that every error spoils one line. However, with Viterbi decoding, errors tend to occur in bursts (i.e., in closely spaced groups) which can do no more damage to a line than can a single error. Therefore the assumption is modified to state that every burst of errors spoils one line.

Figure II-26 is constructed on the following basis.

1) The RF bandwidth of the digital system is 15 MHz. Required RF signal-to-noise ratio is referenced to the noise in this band.

2) The threshold for the FM link is 9 db. The base band for the FM system is 2.5 MHz, and the I.F band is 2(M+1) times as wide, where M is the modulation index. Thus, for M=2 the RF bands of the two systems are equal. For smaller M, the RF band is narrower, and the threshold is shown as less than 9 db over the 15 MHz band, although it is still 9 db over the actual RF band.

3) The video signal-to-noise ratio is relative to a sine wave with negative and positive peaks at full-black and full-white levels.

4) The video signal-to-noise ratio for the FM system is

$$\text{Video SNR} = 3M^2S \Big/ 2N_oB_v$$

where M is the modulation index, S the RF power, N_o the noise spectral energy density, and B_v is the video bandwidth.

If the RF bandwidth is

$$B_f = 2(M+1)B_v,$$

the relation between video signal-to-noise ratio and RF signal-to-noise ratio is

$$\frac{\text{Video SNR}}{\text{RF SNR}} = 3(M+1)M^2.$$

For M=2 this ratio is equivalent to 15.5 db.

5) For digital transmission, Viterbi decoding is used with k=5 and V=2.

6) Burst error frequencies are for simulations as shown previously in Table II-2 and Figure II-9.

7) An average of 486 bits is needed for each line (270 pixels, and 1.8 bits/pixel).

8) An additional 400,000 bits/second of voice and data are carried by the digital system. The same information is transmitted by the FM system during retrace time without requiring any increase in carrier power. For the digital system the total bit rate is 7.5 megabits/second.

9) Quad-phase modulation is used for the digital system. Thus with V=2, 7.5×10^6 quad-phase symbols are transmitted each second. The 15-MHz band limitation truncates the power spectrum at the first minimum with a loss of 1 db. Since the band is twice the bit rate, E_b/N_o is 2 db more than carrier signal-to-noise ratio (3 db for the factor of two, 1 db subtracted for the truncation).

10) Each burst produces one defective line.

The interpretation of Figure II-26 depends on a subjective assessment of the relative merits of a picture with a certain fraction of the lines defective one with a certain video signal-to-noise ratio. If one bad line in 100 is equivalent to a 22-db video signal-to-noise ratio, the digital system needs about 5.6 db less signal-to-noise ratio than the FM system. If one bad line in 100,000 is taken as the equivalent of 25 db, the difference is reduced to 4.5 db.

If the RF band could be increased, the digital system would regain the 1-db spectrum truncation loss and it would also be possible to get an additional 0.4-db coding gain by using V=3 instead of V=2.

In summary, the digital system has an advantage that is between 4 db and 6.4 db. Improvements in the compression techniques might add 1 db.

If a nominal figure of 5 db is assumed it is pertinent to consider how much this will buy in terms of relaxed requirements on the total system; 5 db is equivalent to any one of the following:

1) A reduction of 44% (or more if surface imperfections are considered) in the diameter of the receiving antenna. If cost is proportional to volume, it is reduced by 82%;

2) A similar reduction in transmitting antenna diameter with 82% less weight and pointing tolerance increased by 78%;

3) A 78% increase in range;

4) A reduction of 68% in radiated RF power.

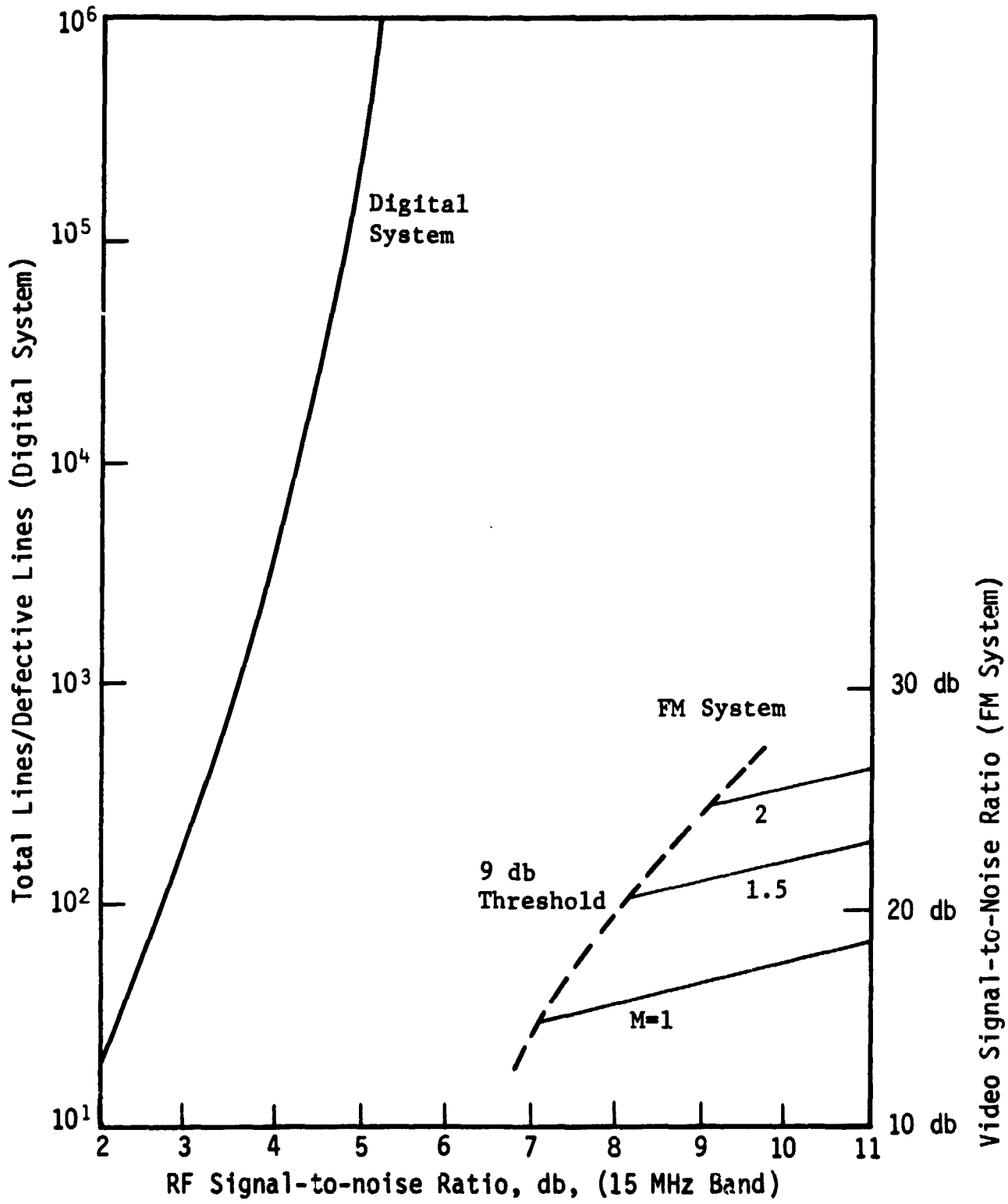


Figure II-26 Performance Comparison of Digital and FM Systems

III. SOFTWARE STUDIES

A. FANO ALGORITHM

Computer simulation of the Fano algorithm was accomplished in two versions, both coded in Fortran for the CDC 6000 series machines. First was a 2/1 data compressor for studies of performance with variations of the threshold, metric, and computational limit parameters. Second was the 6/4 data compressor in combination with an aposteriori probability predictor for picture application. Both compressors are discussed in Subsection 4 of Section A, Chapter II.

B. VITERBI SIMULATION

Simulation of the Viterbi algorithm on the computer was accomplished in three versions: (1) Metasymbol assembler programing for the Sigma-7, for hardware study purpose; (2) Fortran and Compass assembler for the CDC 6000 series, for performance verifications; and (3) Fortran for the IBM 360 for further studies, hardware design, and further development of variations to the Viterbi algorithm.

1. Metasymbol

The Metasymbol version of the program was developed as an aid in hardware design of the algorithm. It can be run on an XDS Sigma-5 or -7, and uses 2150 words of core. It will decode at approximately 300 bits per second at a constraint length of 8. Algorithm operation is almost the same as in actual hardware with the same register shifts, etc.; except that in the hardware fewer registers are used, because transfers can be done in parallel. Because this program was not designed for performance verification runs, the bits decoded are arranged in blocks, and thus have starting and ending effects which alter the statistics of errors.

2. Fortran for the CDC 6000

This version provided statistics for algorithm performance for different constraint lengths, encoder connections, and quantization schemes. This program is usable on a CDC 6400 or 6500

without alteration except for reaction to compilers, which increment their control variable in a DO loop before a bottom exit. The program is convertible to any machine in Fortran by adjustment to different word lengths. Decoding speed is approximately 25 bits per second at a constraint length of 8, and the speed increases approximately by a factor of 2 for each reduction of the constraint length by 1-bit. Limits of table size restrict use to rate 1/2 and 1/3, and constraint length of 3 to 8. This program blocks the input and output bits into groups of 300, but does not clear registers between blocks, therefore it simulates an endless stream of data without beginning and ending effects. This program is coded in a manner not to exactly simulate a hardware decoder, but to run at the maximum speed in Fortran without seriously affecting adaptability to other machines. In order to do this the registers are not actually shifted; but instead, pointers modified to simulate shifts are kept to indicate which bits represent the first or last bit of the register.

The Compass assembler version for the CDC 6000 is a direct conversion of the Fortran program to assembly language, and runs at approximately 80 bits per second at a constraint length of 8. It does not differ from the above program except in speed of execution. Figure III-1 shows a comparison of the results from this version with other coding methods.

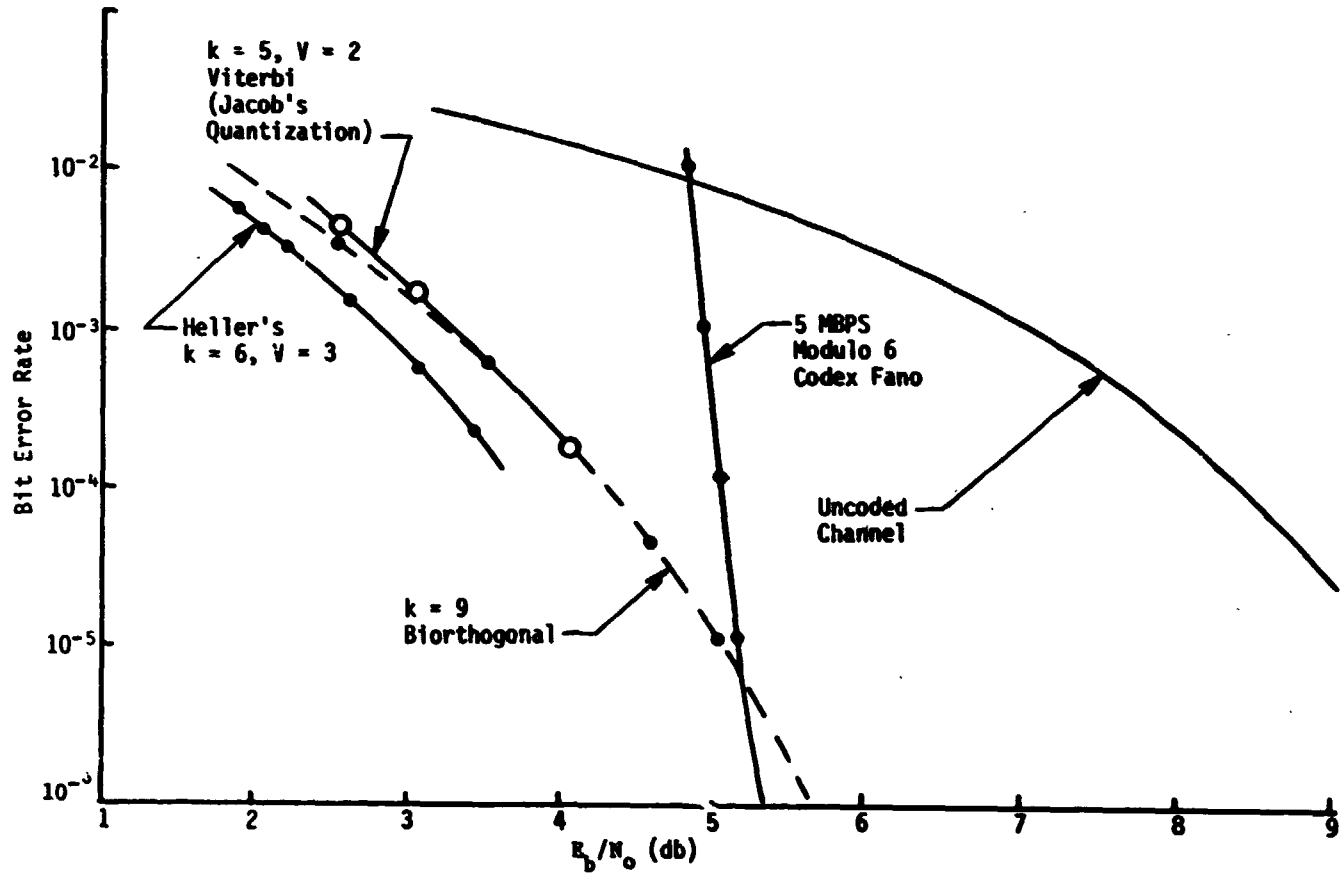


Figure III-1 Viterbi Performance Comparison

3. Fortran for the IBM 360

The Fortran for the IBM 360 varies as to the output options for use in aiding hardware design. These options include: (1) diagnostic printouts so that decoding details can be observed, and (2) register lengths to prevent overflows in the hardware. Also, metric or scoring studies can be run. Running speed is approximately 15 bits per second at a constraint length of 5.

C. COMPRESSION SIMULATION

We have written a Fortran program for use on the CDC 6500 to simulate four compression algorithms as they would be used to compress TV picture data. The algorithms included are ZOI, ZOP, FOI, and FOP. In addition, horizontal and intensity dither can be added individually or in combination.

Input to the program is a 7-track digital tape with up to 6-bits/pixel, and of variable record length. Output is the compressed version of the tape of the form of a 5- or 6-bit/pixel, a 2- or 3-bit run length code, and a variable number of multiple bits to the run length code, terminated by a zero comma. Output for each record is a compression summary for each record or line as the thresholds for when a level will be changed is varied with the buffer contents to prevent overflow or underflow of the buffer, thus producing a variable compression factor.

This program has been used extensively to process picture tapes generated on our image facility.

IV. HARDWARE DESIGN

A. CONVOLUTIONAL ENCODER

A shift register encoder is most generally used to generate a convolutional code. It is the hardware implementation of a mathematically described technique of generating codes. Of the numerous methods of generating these codes in hardware, the one selected and described here can operate at the highest frequency with the greatest reliability.

This hardware design consists of a shift register of K bits where K is the constraint length of the code, and V mod-2 adders where V is the denominator of the rate fraction N/V (in this case N = 1). The connections between the shift register and the mod-2 adders are determined by an optimization procedure described in Subsection 2 of Section A, Chapter II. The output of the mod-2 adders are time-multiplexed to produce a continuous bit stream for transmission. The block diagram for this encoder is shown in Figure IV-1.

The encoder is designed using TTL with MSI wherever possible. The encoder register (Figure IV-2) is a dynamic-serial static-parallel shift register of flipflops packaged 2 per pack and clocked by DACL or data clock, which is a clock signal at 1/V the rate of encoder clock ECL. Each of the outputs ER (0), thru ER (7), are presented to the mod-2 adders. The mod-2 adders are MSI modules that generate odd or even parity, dependent on the state of PARS (parity select) which, in this case, is made a logical "1" for even parity. The inputs ER (0) thru ER (7) are connected for a valid code, and the outputs PAR (N) are presented to the Encoder-Multiplexer as shown in Figure IV-3. The Encoder-Multiplexer in Figure IV-4 consists of a sum of products gating from parallel to serial conversion of the PAR(N) signals, and a synchronous ring counter for generating the commutation signals COM (N) for this gating. The ring counter can be operated for a 1/2 or 1/3 rate by use of the mode control MODE', and is synchronized with the incoming data by the power up reset PUR. If MODE' is held to logic "0", the ring counter holds COM3 to reset and only two counts are used. This counter always resets itself to the 001 state on this first clock pulse ECL, or via PUR so that regardless of what state it comes up in it always starts up correctly.

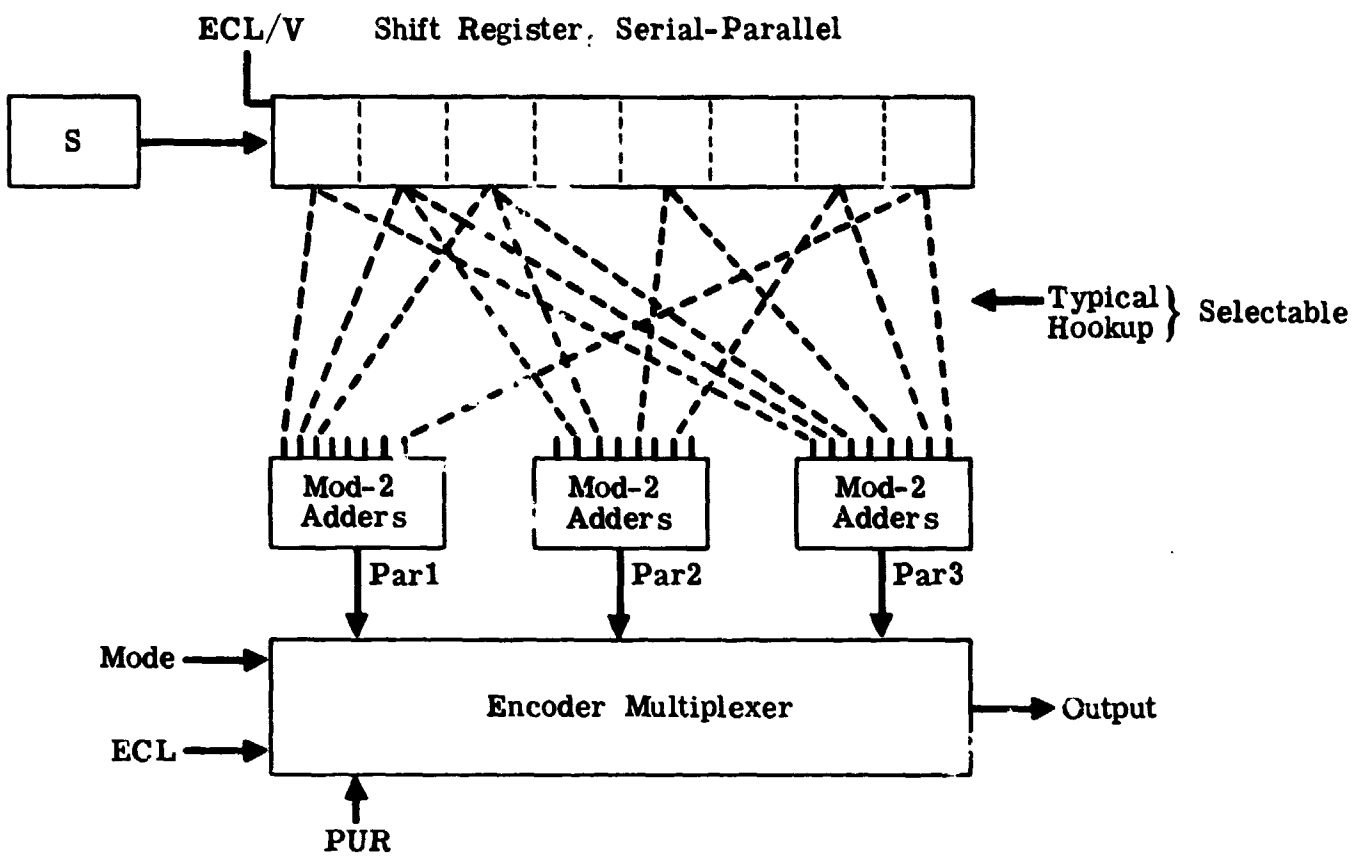


Figure IV-1 Convolutional Encoder

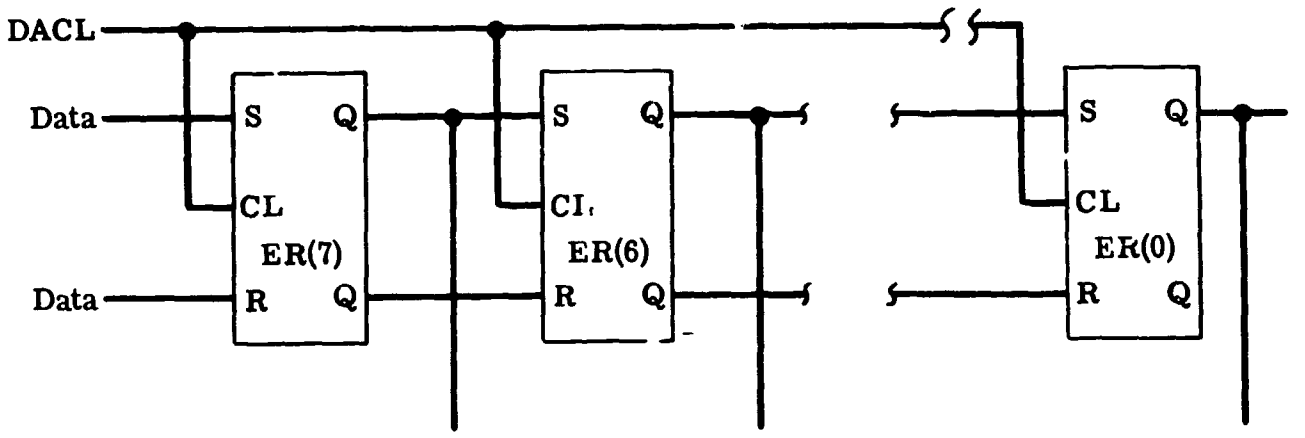


Figure IV-2 Encoder Register

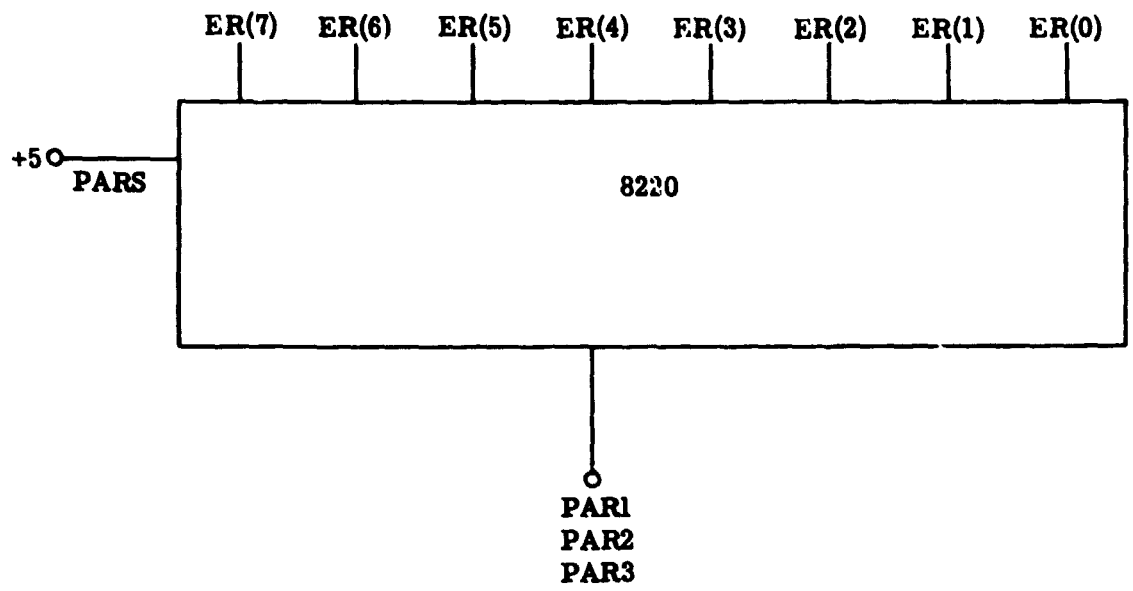


Figure IV-3 Mod-2 Adders

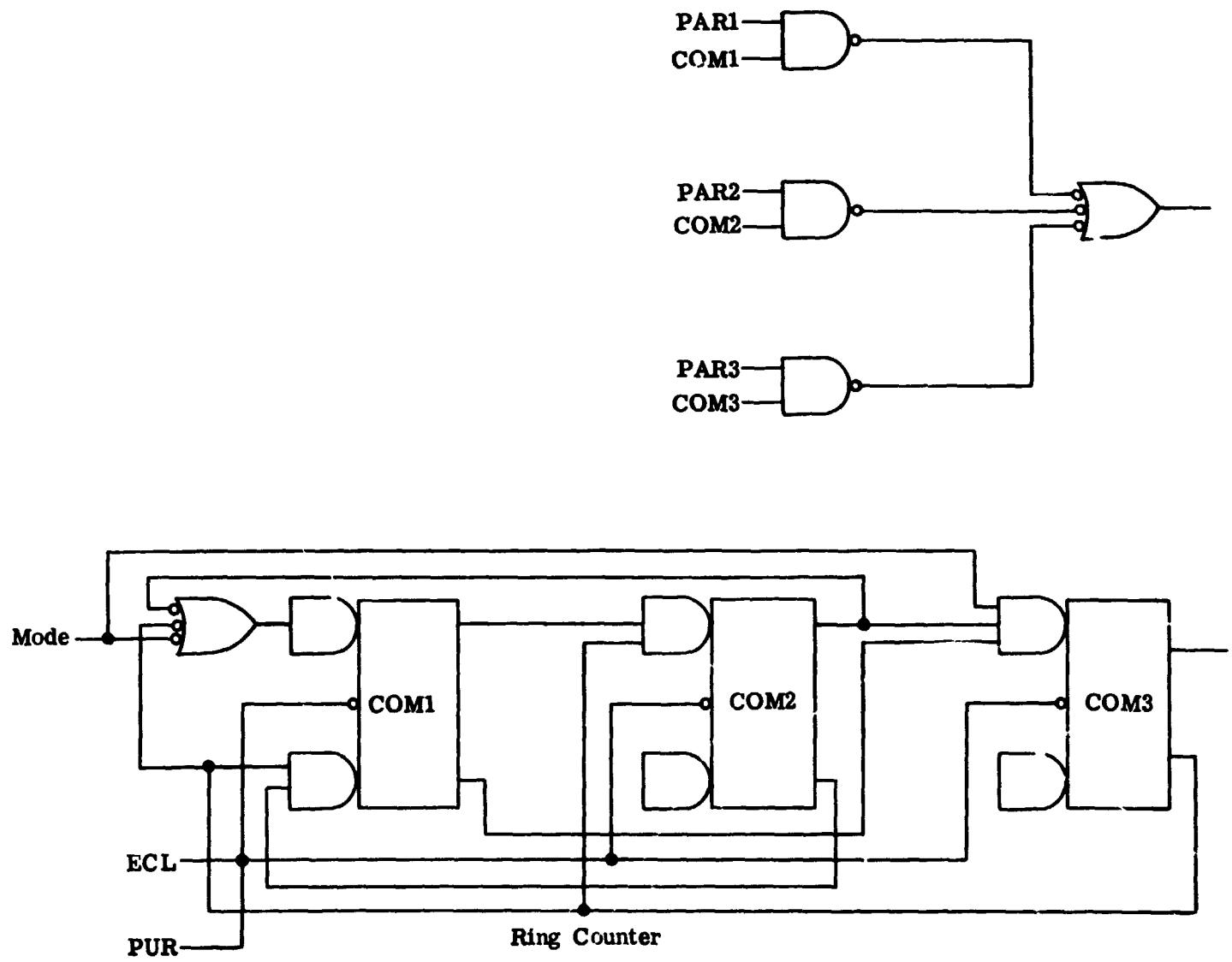


Figure IV-4 Encoder-Multiplexer

B. HIGH-SPEED VITERBI DECODER

In the beginning of this study, we evaluated several algorithms for sequential decoding of convolutional codes, and selected the Viterbi altorithm.

Dr. G. D. Forney was consulted in regard to the existing decoder, which was based on the algorithms by R. M. Fano. The Codex decoder is operated at 5 Mbps. This rate can be increased by about 2 or 3 kbps using an emitter-coupled logic, such as Motorola's MECL III, but cannot be increased by using TTL.

At the outset of this study, Dr. H. M. Gates and Mr. W. B. Anthony of Martin Marietta, and Mr. P. Batson of NASA-MSC, made a trip to NASA's Jet Propulsion Laboratory (JPL) to discuss the sequential decoder algorithm conceived by A. J. Viterbi. The JPL personnel involved in this discussion were J. Heller, E. Posner, and S. Butman. Dr. Gates and Mr. Anthony related the contents of their talks at JPL to the others on the study team.

They concluded that the JPL personnel had a thorough understanding of the Viterbi algorithm, a working computer program to simulate the algorithm, and a conceptual design for a low-speed (100-MHz) hardware decoder. A decision was made not to use their hardware decoder design; but, by using a fully parallel system and selecting a family of logic that was consistent with our needs, we could design a high-speed decoder that could operate above 10 Mbps, and at a speed as great as 30 Mbps if necessary, to implement the algorithm.

The design was based, where possible, on a fully parallel approach with no dependence on logic family to allow the use of any family of logic elements from DTL to MECL III. This created problems, since, in some areas, family dependence was mandatory. The original logic family chosen was MECL III because it was the fastest available, had very few elements, and was the most restrictive in hardware configuration. In September 1969, Mr. D. L. Manion (Martin Marietta) met with personnel from Motorola's Semiconductor Products Division to discuss applications of MECL III and other ECL lines and the potential use of Custom LSI in the design and construction of a system like the Viterbi decoder. After reviewing the speed requirement for the decoder, and the complex packaging problems of MECL logic, it was decided to base the designs on the high speed TTL family. From this point, all subsequent study was directed to the use of TTL with maximum use of MSI and LSI to increase speed and reduce costs.

The block diagram of Figure IV-5 shows the Viterbi algorithm implemented as parallel hardware systems for K = 3 and V = 2. K = 3 and V = 2 are chosen for simplicity of drawing and can be expanded by parallel addition of adders, comparators, and registers.

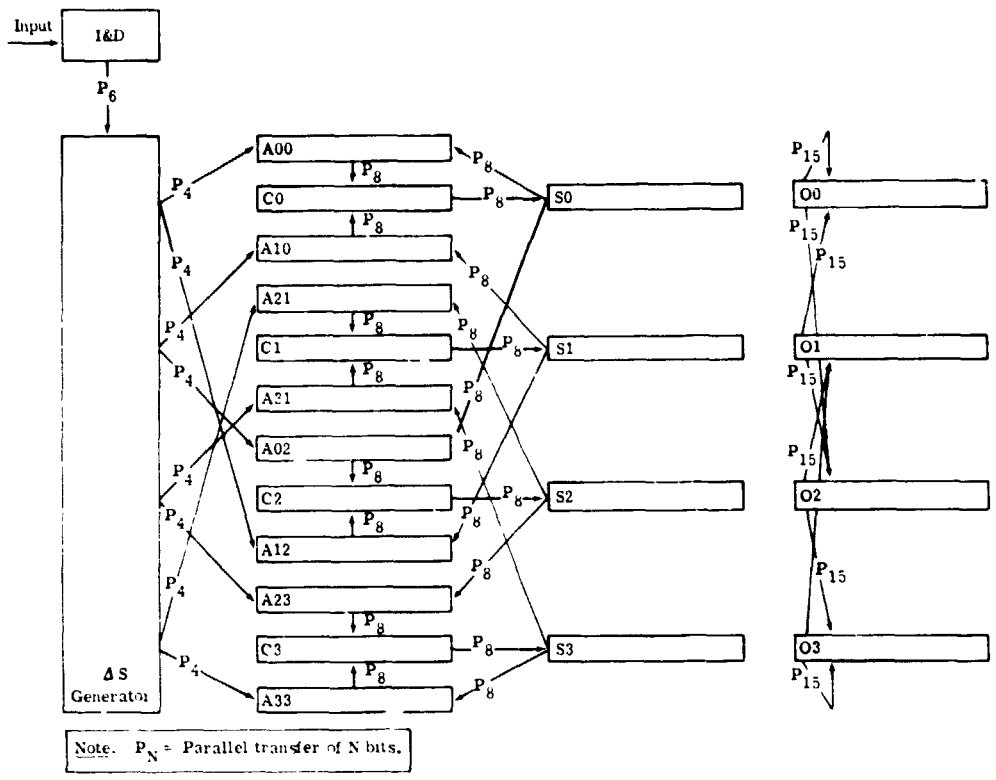


Figure IV-5 Viterbi Decoder

The parallel approach has been selected to take advantage of its speed, but involves using more hardware than a more serial system would use. The block diagram is presented as a single line flow diagram in which the type of transfer and the number of bits are indicated on the line [P(N) = parallel transfer of N bits]. The symbols on the blocks are:

1) I & D = Integrate and dump;
2) ΔS Generator = the score-increment generator;
3) A_{xy} = an adder receiving data from the S_x register and presenting its sum to the S_y register;
4) C_y = a comparator that generates the load command for the S_y Register;
5) S_y or S_x (same if x = y) - the score accumulator;
6) O_y = the output "tale" register.

The quantizer circuit contains an integrator that extracts the signal from the noise picked up during transmission and receiving, and an analog-to-digital converter that takes the analog level of the channel bit and converts it to a 3-bit binary number. This conversion is made in parallel on each channel bit (in this case 2) to form the outputs A = (M0, M1, M2) and B = (N0, N1, N2), and their complements A´ = (M0´, M1´, M2´) and B´ = (N0´, N1´, N2´). These outputs then are the inputs of the ΔS generator. The I & D is shown in the block diagram of Figures IV-6 thru IV-8.

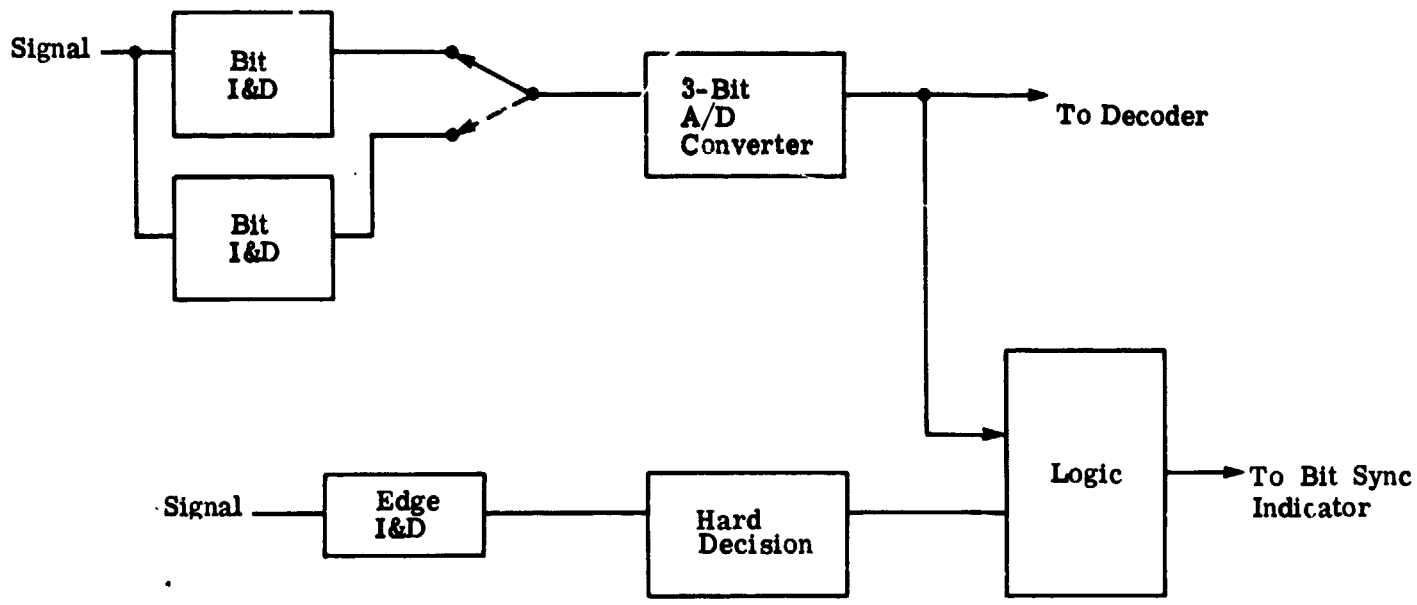


Figure IV-6 A/D Quantizer

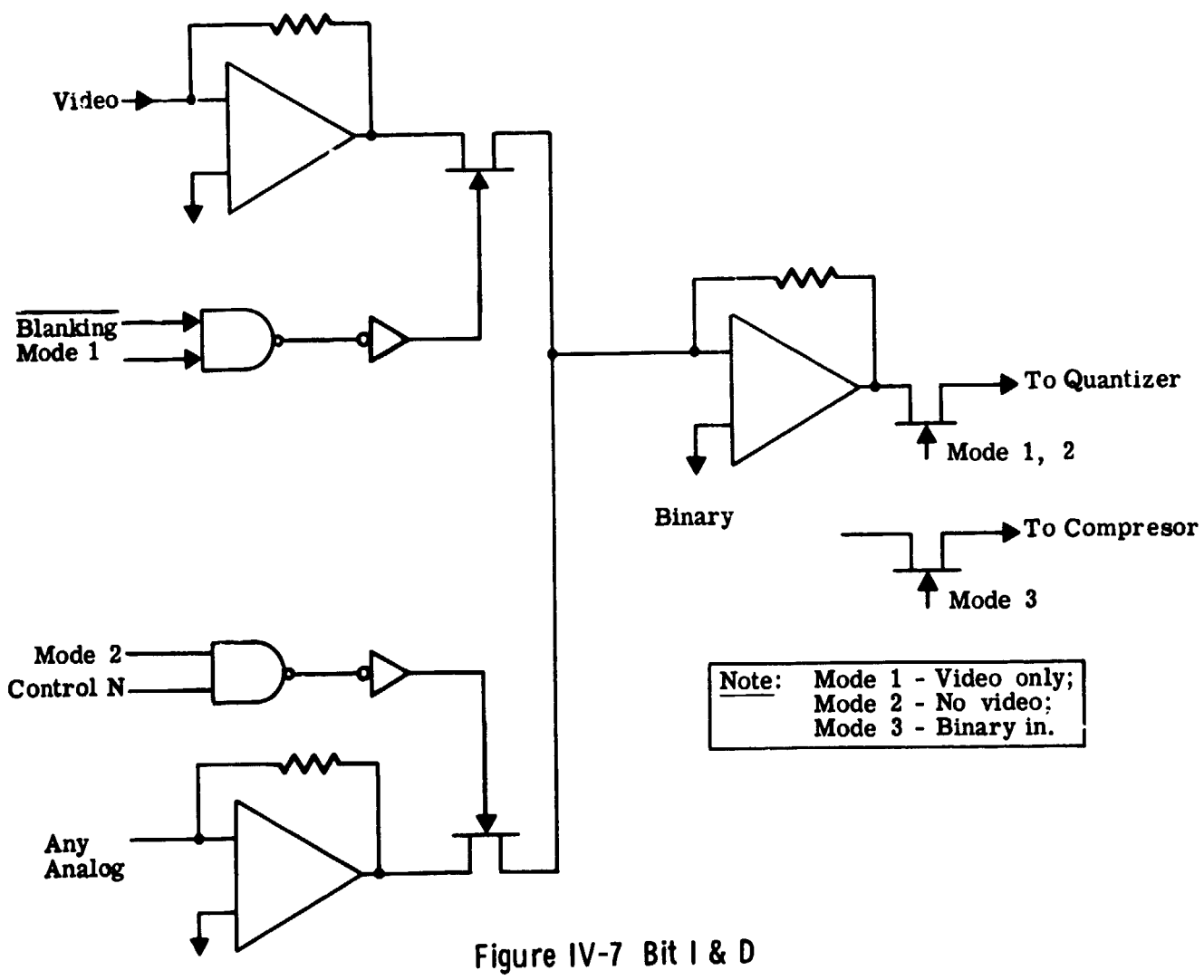


Figure IV-7 Bit I & D

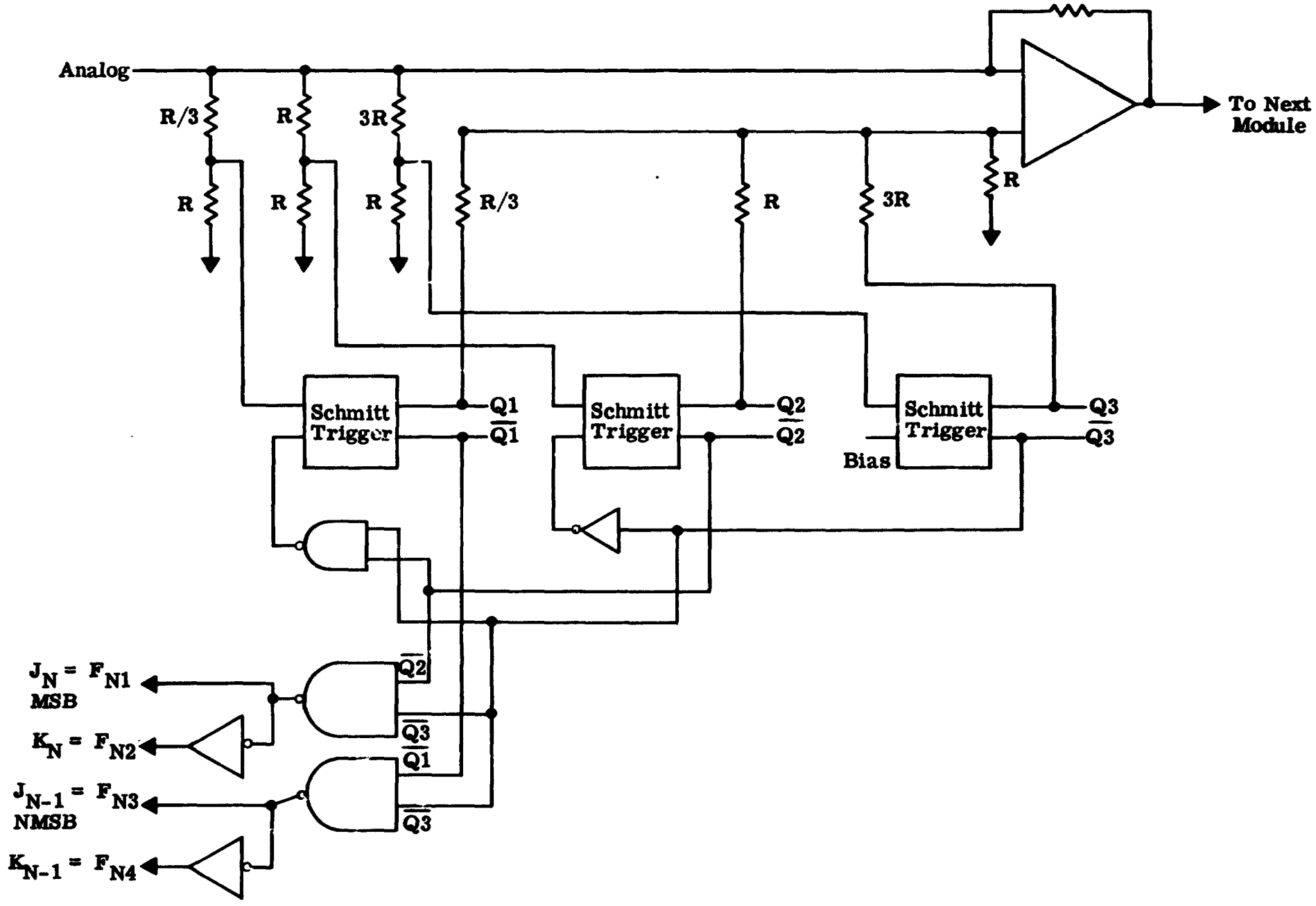


Figure IV-8 Converter Module

The ΔS generator receives the inputs A, B, A´, and B´ (actually their M and N components) from the I and D, and converts these to four 4-bit score increments. Each is then taken in pairs and added to yield A+B, A´+B´, and A´+B, and A+B´ (where + is SUM rather than logical OR). This operation is performed by four 6-input bit circuits with inputs (A,B), (A´,B´), (A´,B), and A, B´); and generate ADEL = (A0, A1, A2, A3), BDEL = (B0, B1, B2, B3), CDEL = (C0, C1, C2, C3), and DDEL = (D0, D1, D2, D3). This circuit is shown in Figure IV-9.

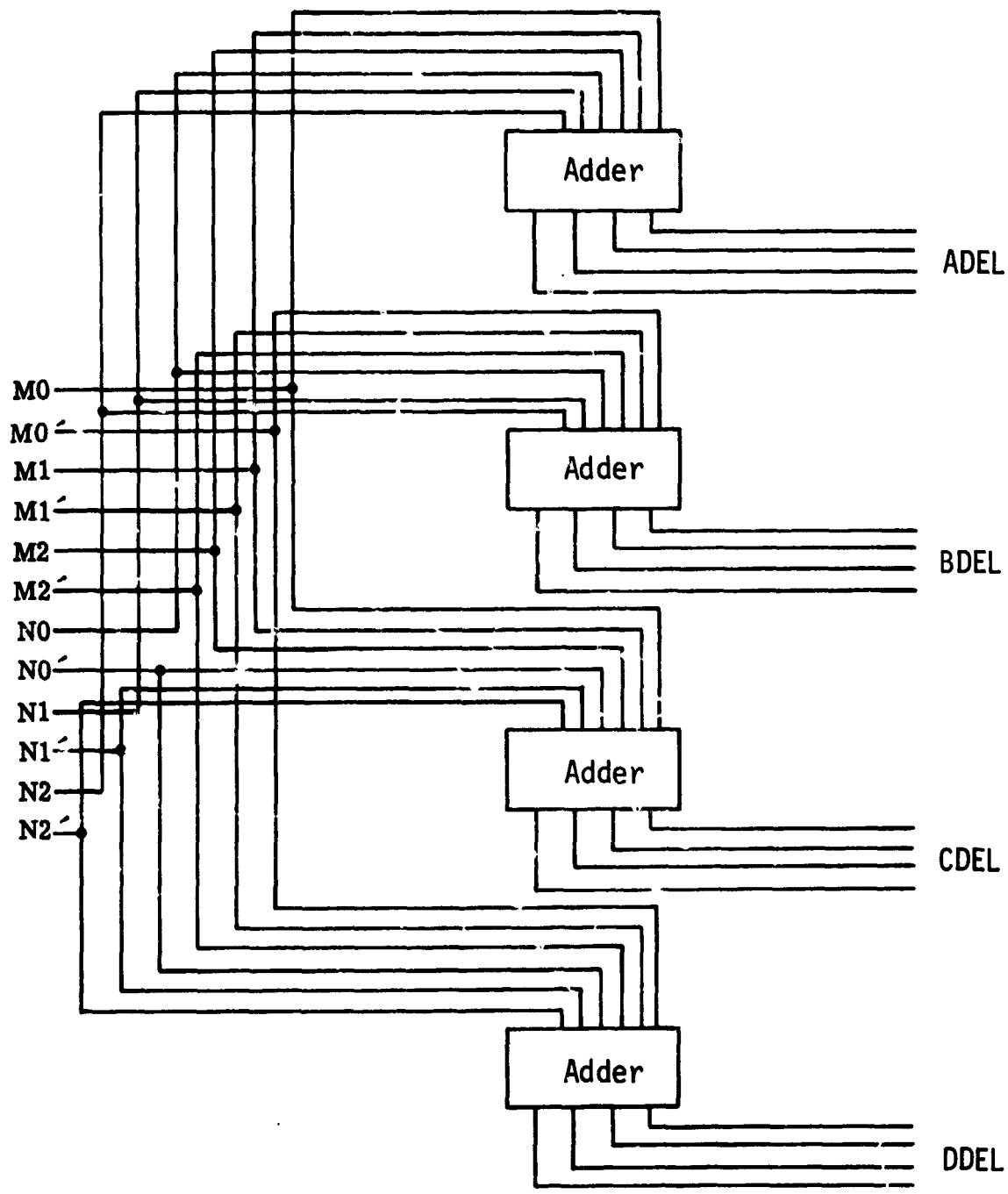


Figure IV-9 ΔS Generator

The adders Axy consist of two 4-bit adders totaling 8 bits. These adders receive their addends from the ΔS generator and the Sx register and present their sum to the Cy comparator, for selection, and to the selection logic of the Sx registers. The logic for this is shown in Figure IV-10.

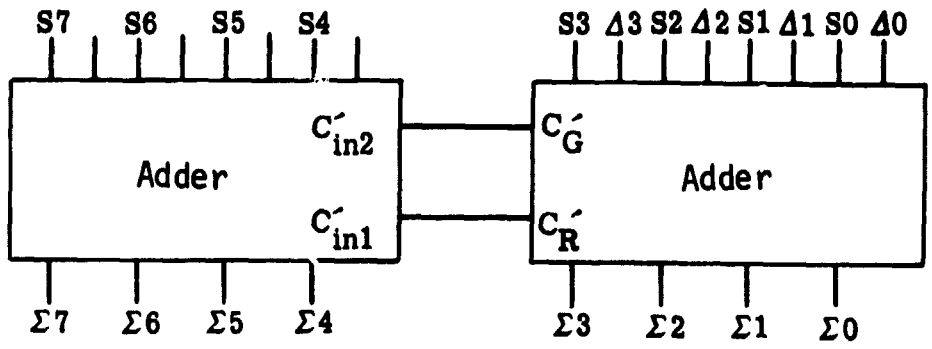


Figure IV-10 Viterbi Adder

The Cy comparators consist of 8-bit logical comparators operating in a parallel combinational mode. The comparator puts out a signal indicating whether the Axy or A(x´ + 1´) sum is the lesser of the two and steers this sum to the inputs of the Sy register. The output signal mnemonic is Txy = (T(x + 1)y, which is also used by the θ register for transfer control. Comparator logic is shown in Figure IV-11.

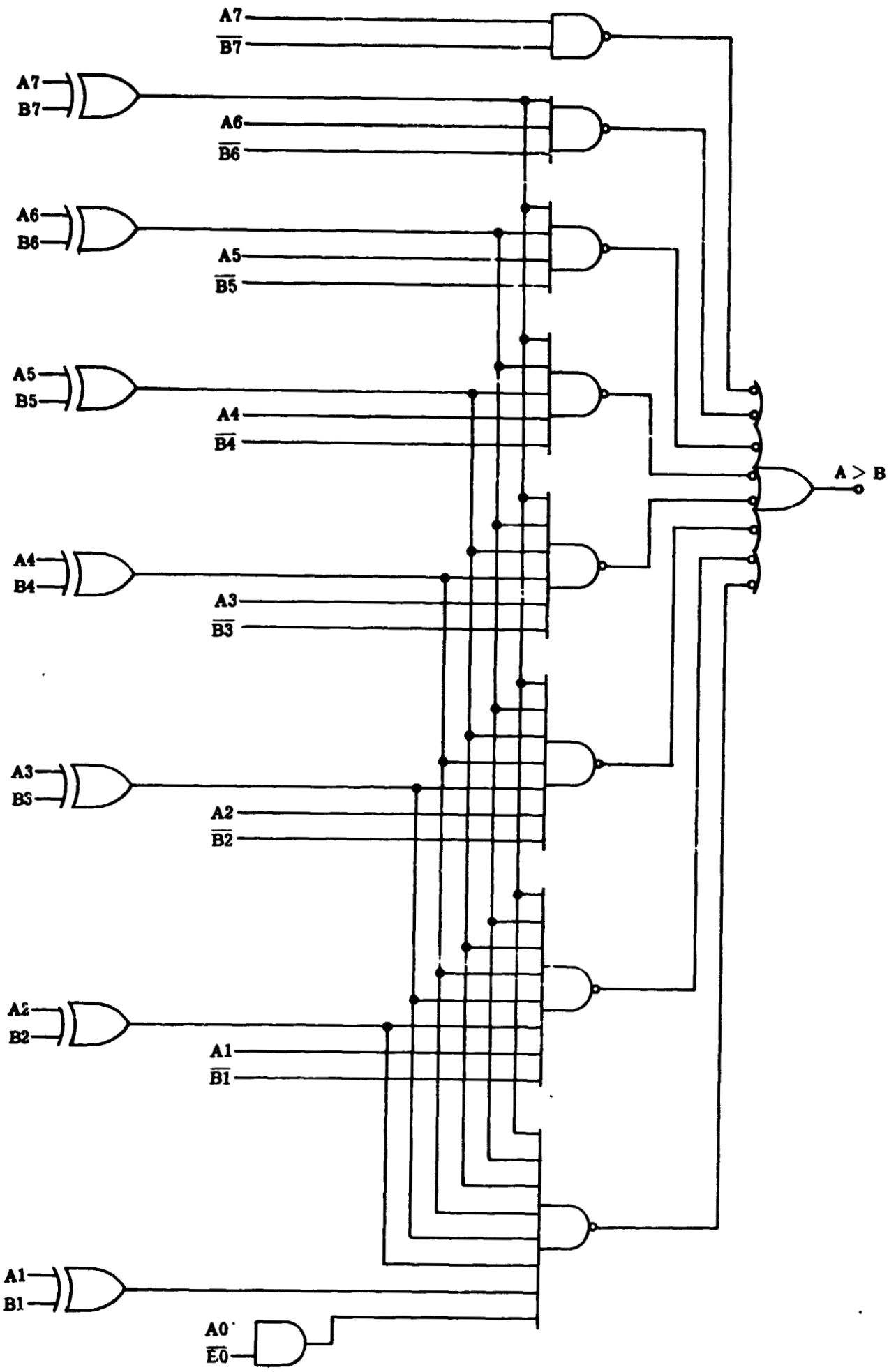


Figure IV-11 Comparator

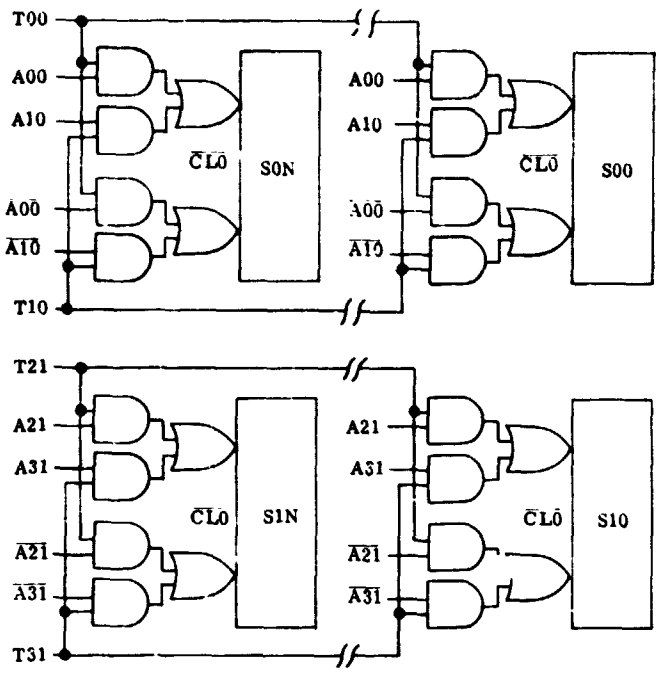


Figure IV-12 Score Register

The Sy or Sx registers are the score accumulators for the node scores. They consist of 8-bit flip-flop parallel-parallel registers. These registers receive their inputs from the Axy adders. The logic for this register is shown in Figure IV-12.

The output registers θy, shown in **Figure IV-13**, are 15-bit parallel-parallel flip-flop registers wired for a 1-bit shift transfer on command of Txy.

A timing diagram for the closed loop function is shown in **Figure IV-14**.

In addition to the k = 5, V = 2 mode, the high-speed Viterbi encoder-decoder will also be able to operate with k = 3 or 4 and with V = 3. These options require only a small increase in hardware over a single-mode system.

A parts count for the system is shown in Table II-1.

Table II-1 Viterbi Decoder Logic (k = 3, 4 or 5; V = 2 or 3)

Symbol	Function	No. of Dual In-Line Packages (k = 5 and V = 3)
ΔS	8 Adders	16
Sx	8 Bits x 2^{K-1} Registers = 8 x 16 = 128	128
Axyz	2 Adders x 2^{K} = 2 x 32	64
Cy	2^{K-1} (7x-OR, 1/2 2-Input, 1 4-Input, 6 8-Input, 1/3 And)	120
Cx	(5 x K) Bits x 2^{K-1} Registers = 400 Bits + 100	500
	Control	40
	Total	868

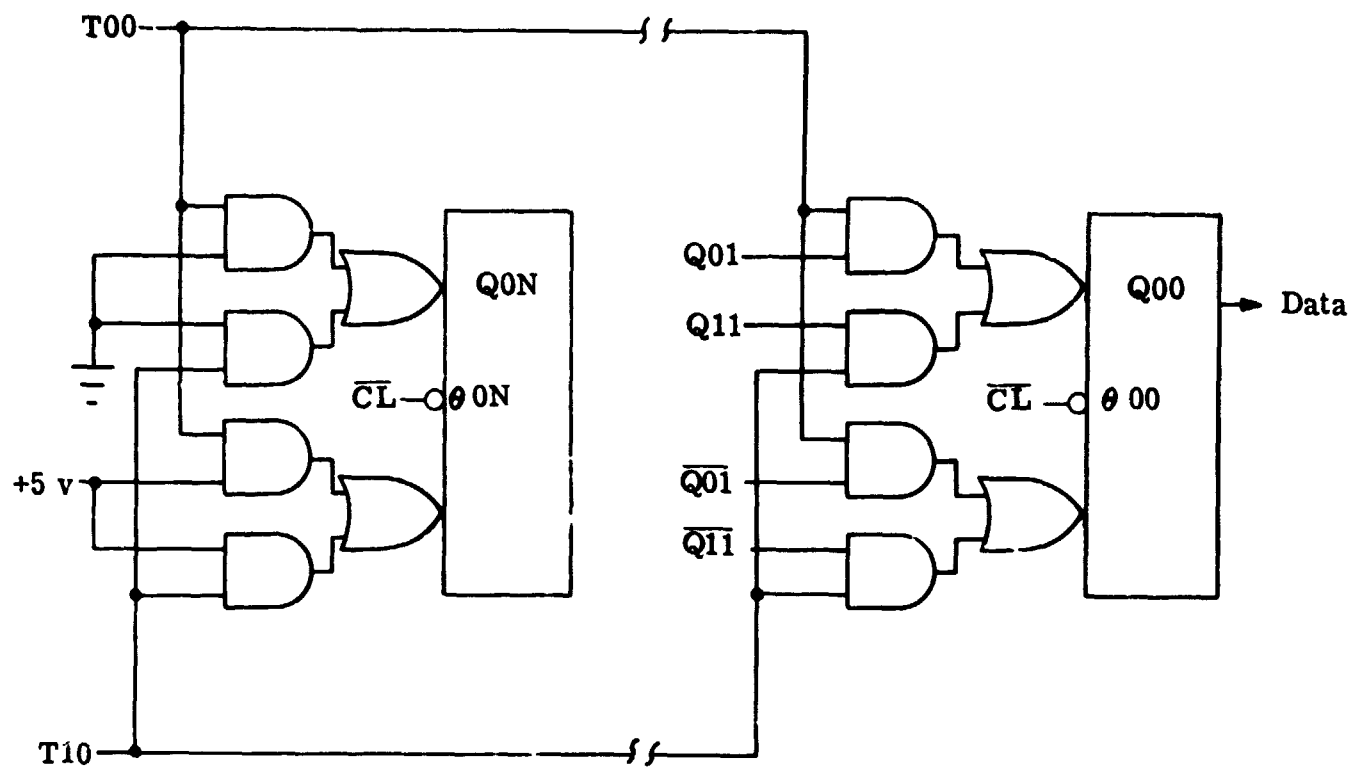


Figure IV-13 Output Register

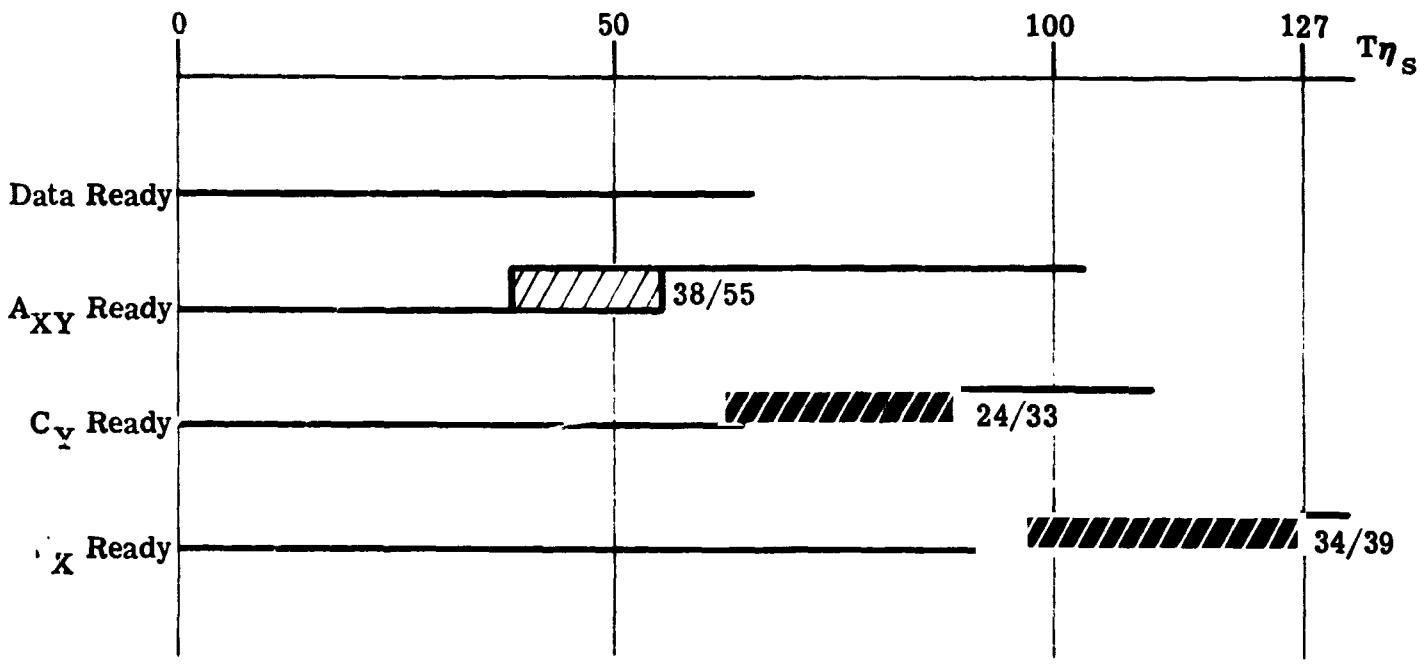


Figure IV-14 Viterbi Timing Diagram

C. SERIAL VITERBI DECODER

The Viterbi algorithms can be implemented for low-speed operation as a serial system, which has fewer components than a parallel system.

The decoder shown in Figure IV-15 is a K = 3 rate 1/3 decoder. The effect of increasing the constraint length K by one is to double the number of S registers and output registers. The output registers each have 5(K) stages. The hookup for the ΔS's also changes with a change in constraint length. Because of the serial nature of this decoder, its maximum operating frequency will decrease with an increase in K.

The serial Viterbi decoder receives 3 channel bits and quantizes each of these via an A/D converter to 3 bits. These 9 bits are then compared to the 8 possible hard-one and zero cases that could have occurred, and a delta score (ΔS) is generated. These ΔS's are then added to the previous nodal scores according to a pattern determined by the convolutional encoder connections. These additions are handled serially in pairs by the relation

$$S_y(t+1) = \min \left(S_x + S_z\right), \left(S_{(x+1)} + \overline{S_z}\right),$$

where x is the number of a score register S_x, and z is the hard binary number that the input bits were compared to. These pairs of sums are then compared by C_y, in parallel, and the lower is loaded in parallel into the appropriate S_y register. The output tale register is also transferred at the same time the A_{xyz} is transferred co S_y. The sequencing of this decoder will be done by a stage counter with $2^{(k-1)}$ stages, where k is the constraint length of the code, and by a substage sequencer to ensure time for the three operations of adding, comparing, and transferring. At the end of each cycle, the roles of the S_x and S_y registers are interchanged and the process is repeated.

The hardware requirements for a K = 7 rate 1/3 decoder are summarized in Table IV-2. The detailed logic of the serial decoder is the same as that for the parallel decoder.

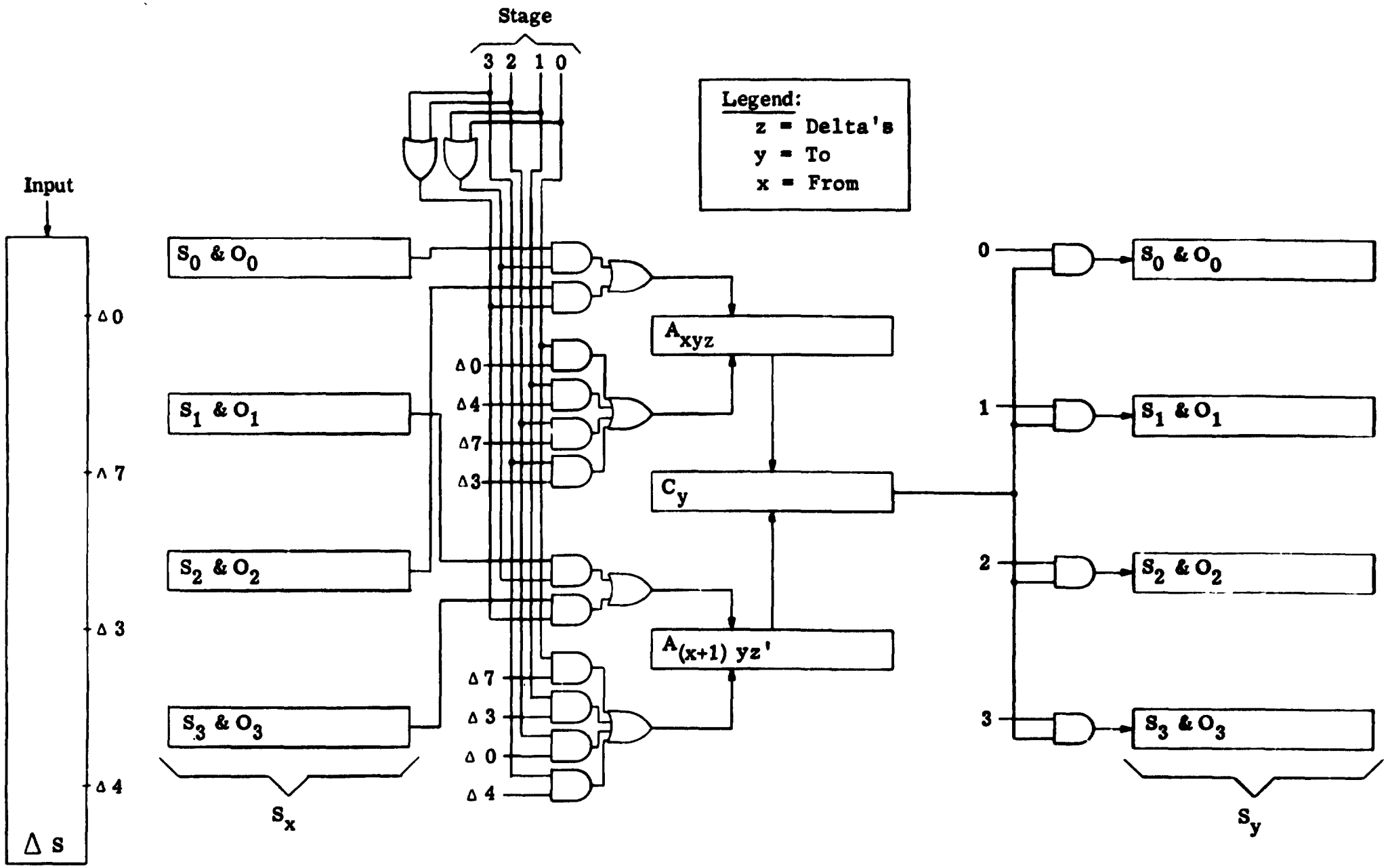


Figure IV-15 Serial Viterbi Decoder

Table IV-2 Hardware Requirements

HARDWARE	QUANTITY
ΔS Generator	25 ICs*
S_x and S_y	
2 x 10 bits each x 2^{k-1} = 2 x 10 x 64	20 ICs
1280 bits at 64 bits/IC	20 ICs
C_x and C_y	
2 x 45 bits x 5(k) = 3150 bits	
3150/64 bits/IC = 50	50 ICs
A_{xyz}	
2 x 3 adder x 2 sets =	12 ICs
2 x 25 packs steering x 2 sets	100 ICs
C_y	
3 adders x 2	6 ICs
Miscellaneous	50 ICs
Total	283

*Integrated circuits.

D. CORRELATION DECODER FOR BISIMPLEX CODES

In the beginning of this study, bisimplex codes were rejected for error correction because:

1) Large amounts of hardware were required;
2) A large bandwidth was required to obtain the same gain as that for the sequential codes;
3) Their performance is not as good as that of sequential codes.

The following describes a block decoder built by Martin Marietta on a company-funded research project.

A system block diagram is shown in Figure IV-16. As can be seen, the system must integrate and dump the input signal, digitize, operate arithmetically on the word registers, generate all control signals, secure and maintain bit and word synchronization, and select a word register and index for an output. (It is assumed that the input signal to the decoder comes from the detectors of a receiver that is holding carrier lock.)

The operating rate of such a decoder depends on the code used; e.g., a (15,5) bisimplex code with a 10-Mbps information rate requires a channel rate of 30 Mbps.

We have designed, built, and successfully tested a decoder using a (15,5) code with a 33.3-kbps information rate. The system works in real-time, and operations are performed at a 10-MHz clock rate.

In a bi-orthogonal or bisimplex system, three areas offer major obstacles to high-speed operation:

1) Integrating and dumping;
2) Digitizing and transferring the word registers;
3) Selecting the word register with the highest count.

The arithmetic operations involved are minor problems when compared with these.

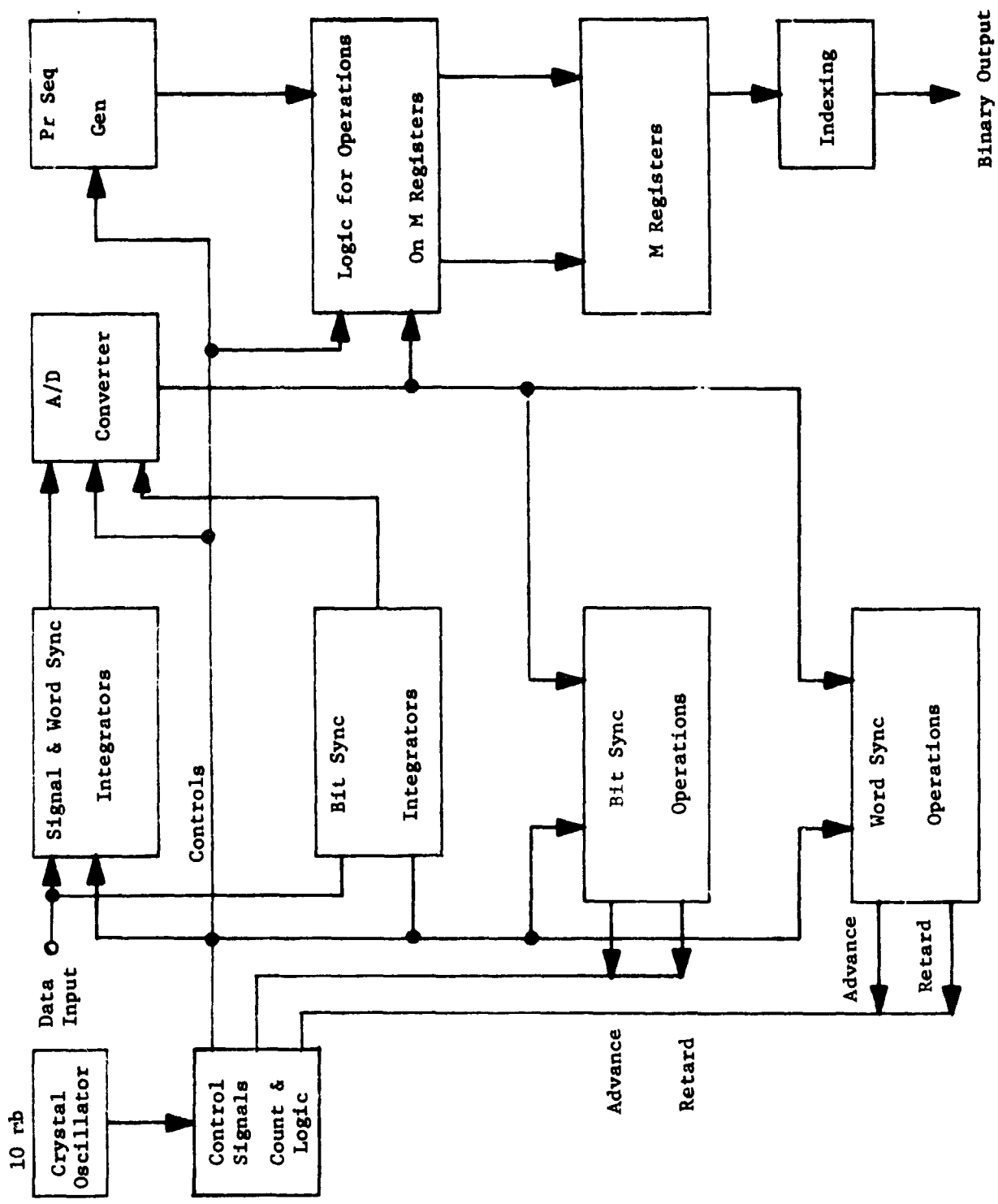


Figure IV-16 Correlation Decoder for Bisimplex Code

The following discussion deals with ways to maximize speed in each of the three major problem areas, using components presently available.

At least two word integrators must be used. Each must be sequenced to integrate a full bit time, hold the answer until it is digitized, and initialize the registers before receiving the next signal. Thus, the two integrators must work alternately. In addition, continuous bit-synchronization required another pair of integrators that will each integrate a small part of the trailing edge of one bit and a small part of the leading edge of the succeeding bit. All the integrator modules must have a bandwidth much greater than the channel rates (or a rise and fall time equal to a negligible part of a channel bit time). Any loss of bandwidth is a direct loss of signal.

Conventionally, an integrating and dumping module is arranged as shown in Figure IV-17. At high megahertz rates, however, such an arrangement is impractical because the isolating switches are too slow. In such a case, the circuit shown in Figure IV-18 can be used to eliminate the isolating switches. Furthermore, if the digitizer continuously reads the integrated signal, it is possible to eliminate the "hold" cycle of the integrators. In Figure IV-19, the Schmitt triggers are adjusted to respond at the quantizing thresholds, and are interconnected so that the only one turned on is the one corresponding to the quantum level of the input. At the precise end of a bit time integration, it is possible to store the digitized result.

This concept combines and minimizes the first two of the speed-limiting areas and will permit system operation up to a 50-Mbps channel rate -- 20 nsec for integration, 13 nsec for dumping, and between 1 and 2 nsec for digitizing and transferring into word registers. One to 16 digitizing levels may be used without difficulty, depending on requirements and on funds for hardware. Schmitt triggers are shown as the digitizing elements (tunnel diodes could also have been used) principally because hysteresis can be more closely and easily controlled.

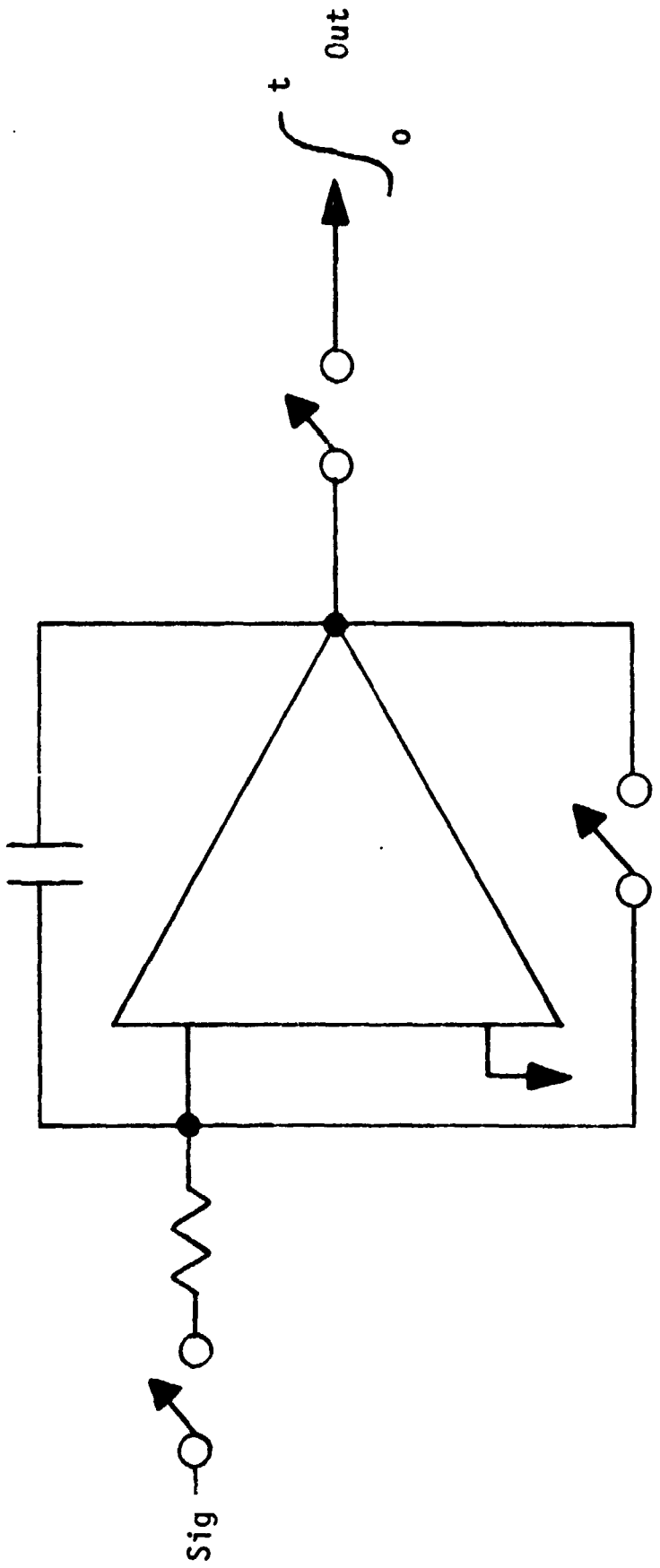
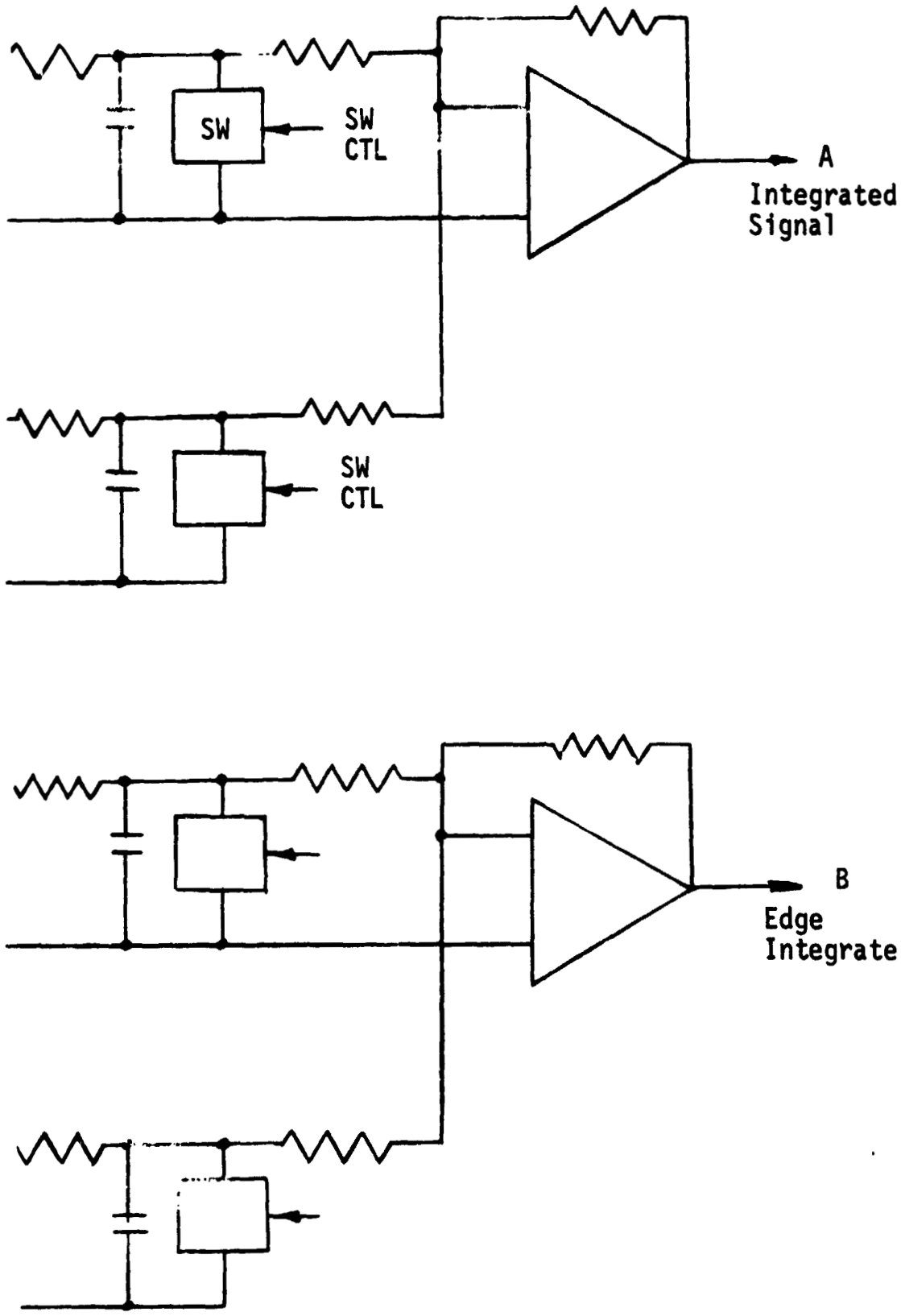
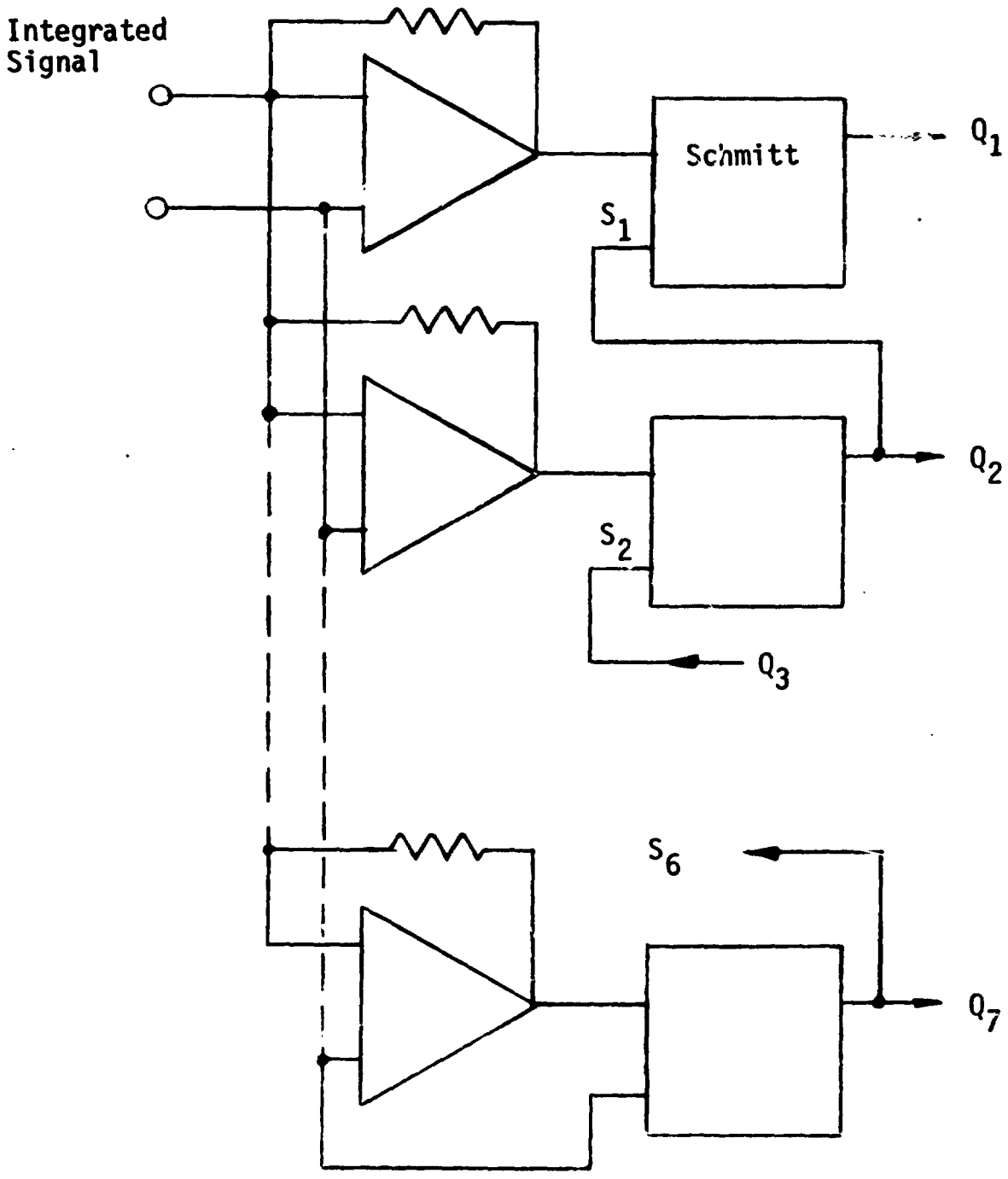


Figure IV-17 Feedback Integrater with Switches



·18 Integrater Array Using Shorting Switches



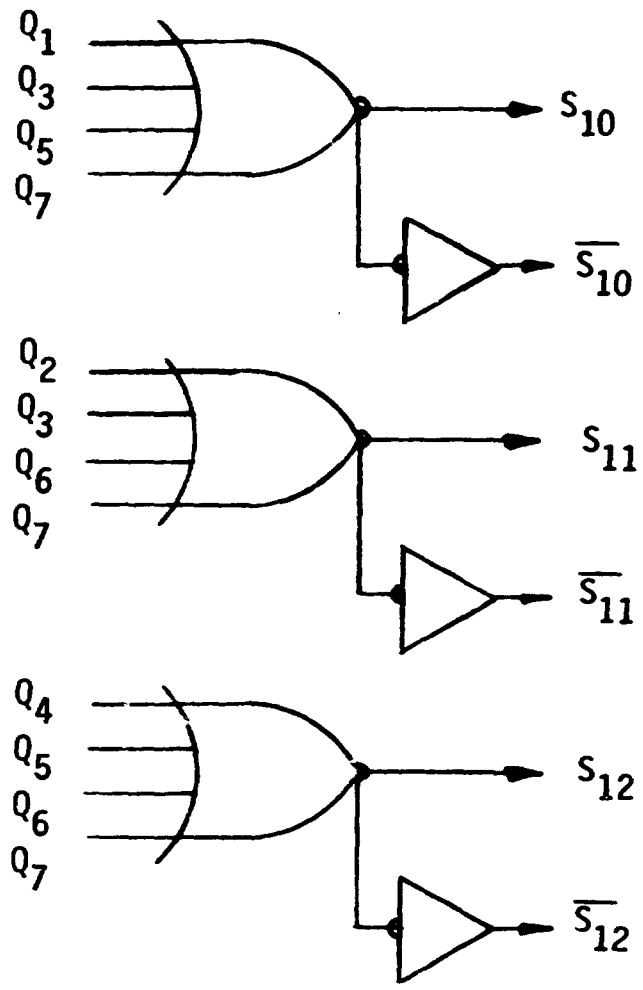


Figure IV-19 High Speed A/D Concerter

Word selection can be done as a serial operation in three ways:

1) By comparing each bit in each register successively with the same bit in all other registers and eliminating the smallest;
2) By turning it into a parallel operation (with four logic levels);
3) By handling serially within one bit time (20 nsec).

For both of the last two schemes, the operation must be a clock rate greater than 200 MHz. This is feasible, but has the major drawback of depending on one series of logic elements from one manufacturer.

A better solution would be to transfer the contents of the registers into a matching set of simple latching elements and to serially decode this set of latching registers (actually, only one would have to be transferred from each orthogonal pair). The decoding or word selection could now be done in approximately half of the word time, and the balance of the time could be used to achieve and maintain word synchronization. This method, too, would require less hardware than either of the first two decoding methods and would allow components to be obtained from two manufacturers.

Figure IV-20 shows the configuration of the first 4 bits of a typical word register and transfer register. The algorithm for selection is: if there is a "one" in the Nth bit of any register, all registers having "zero" in that position are cleared; after the least significant bit has been checked, the largest remaining register is indexed according to its proper code word.

No difficulty is expected with implementing word and bit synchronization in equations, although some compromises may be necessary in the time it takes to achieve bit synchronization. However, this will be a negligible factor, considering a 50-Mb channel rate.

A bisimplex decoder can be built to operate at up to a 50-Mb channel rate. The components are all presently on the market. Two manufacturers are available as sources, and both are reliable and have proven products. The signal can be digitized to up to 16 levels, and up to a (15,5) code can be used at a reasonable cost with an unduly complex hardware configuration. No major problems are contemplated using the suggested system.

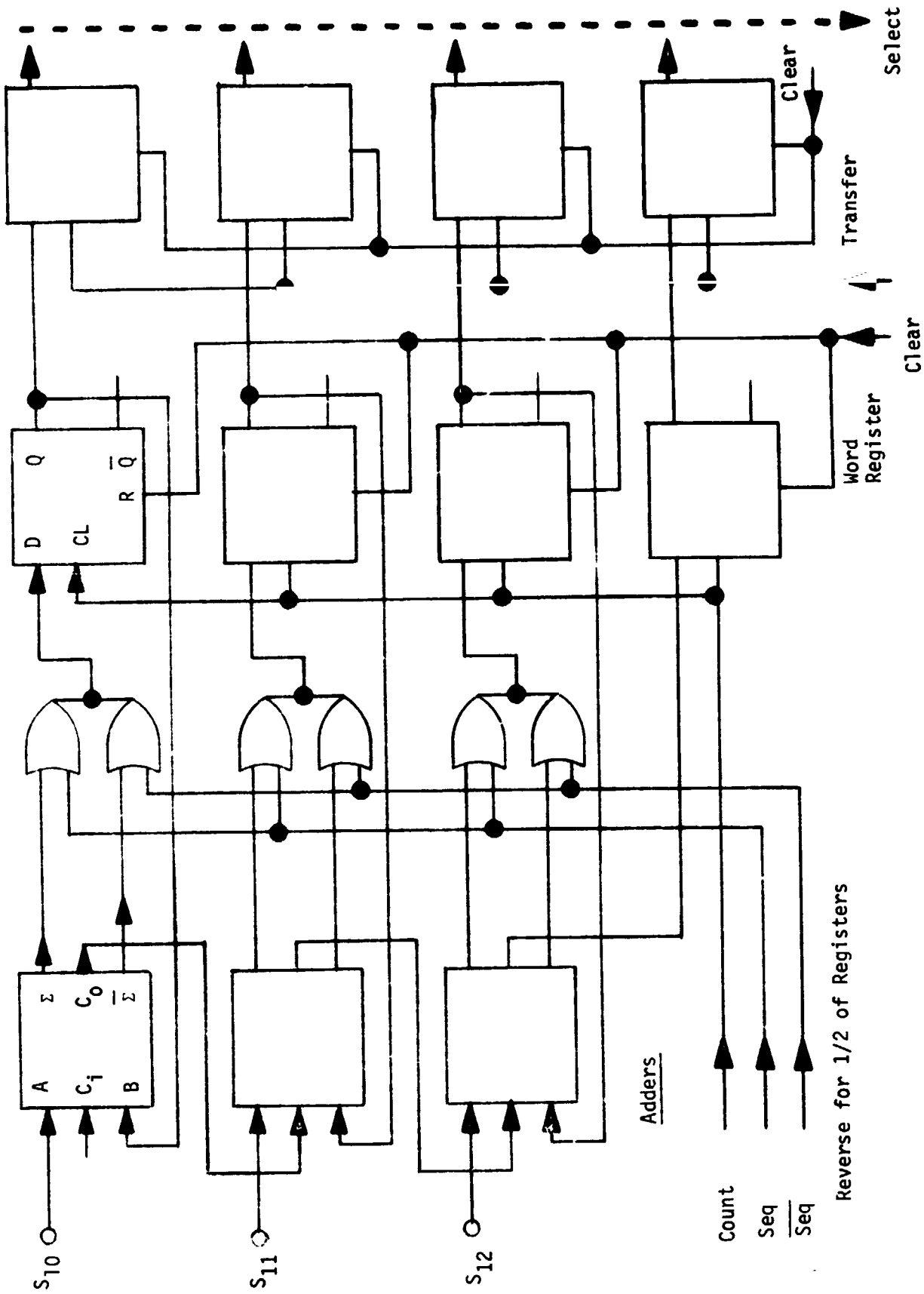


Figure IV-20 Word-Registration Logic for Correlation Decoder

E. COMPRESSION HARDWARE

1. Zero-Order Predictor

The zero-order predictor (ZOP) operates by transmitting only samples that differ from the previously transmitted sample by more than a tolerance (TOL).

The ZOP will be constructed of TTL micrologic elements. Maximum use will be made of medium-scale integration (MSI) and large-scale integration (LSI) wherever possible. Many of the operations must be done in parallel to permit operation at the maximum data rate. The basic elements are an input register, two holding registers, two comparators, and two adders.

The steps in the operation of the unit are:

1) Receive YN, the current data value;
2) Compare YN to YMIN, the minimum of YN since the last transmission. Compare YN to YMAX, the maximum since the last transmission;
3) If YN < YMIN or YN > YMAX, set YMIN = YN - TOL and YMAX = YN + TOL, and transmit YN.

The logic flow of these steps is shown in Figure IV-21, and the block diagram is shown in Figure IV-22.

2. Zero-Order Interpolator

The zero-order interpolator (ZOI) operates by transmitting only the average of the maximum and minimum values of a number of samples that do not differ from one another by more than a tolerance of 2T.

The ZOI will be also constructed of TTL micrologic elements, and maximum use will be made of MSI and LSI wherever possible. As with the ZOP, parallel logic will be used to ensure operation at the maximum data rate. The basic elements are an input register, an anverage generator, three comparators, two holding registers, a subtractor, and a buffer register.

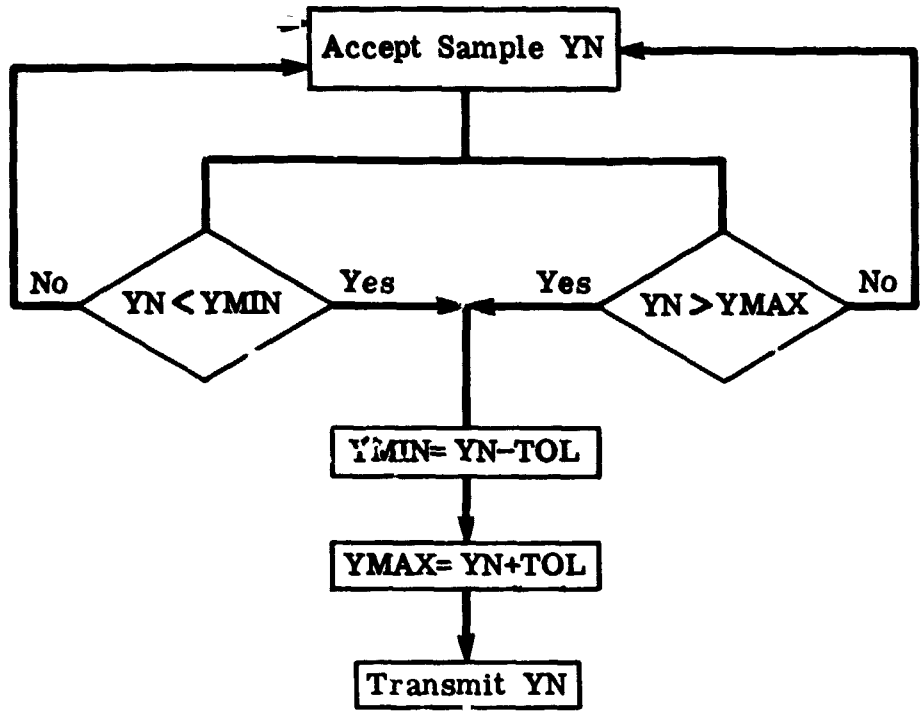


Figure IV-21 Zero-Order Predictor Flow Chart

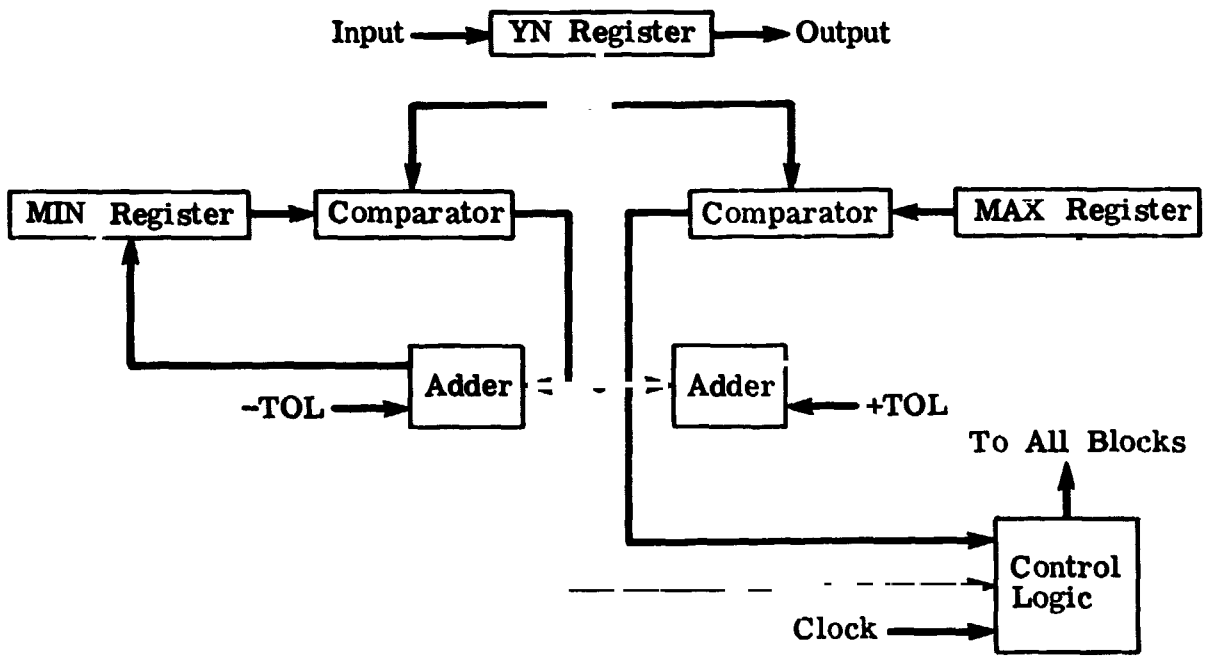


Figure IV-22 Zero-Order Predictor Block Diagram

The steps in the operation of the unit are:

1) Receive input bits YN;
2) Compare YN to YMAX, Compare YN to YMIN;
3) If YN > YMAX, set YMAX = YN, If YN < YMIN, set YMIN = YN;
4) Set AVG = $\frac{YMAX + YMIN}{2}$;
5) Subtract YMIN from YMAX = DIFF;
6) If DIFF > TOL, set AVG into BUFF;
7) Set YMIN = YMAX = YN.

The logic flow of these steps is shown in Figure IV-23, and each block of the ZOI is shown in maximum logic detail in Figures IV-24 thru IV-31.

The input register YN receives N bits in parallel from the A/D converters. These outputs are simultaneously presented to two comparators that compare YN to the previous maximum and minimum values of YN, designated YMAX and YMIN, respectively. If YN is found to be above YMAX or below YMIN, this new YN is made the new YMAX or YMIN value. If YN is within the YMAX and YMIN values, a new YN is accepted and these steps are repeated. When a new YMAX or YMIN is assigned, YMIN is subtracted from YMAX and this difference (DIFF) is compared to a tolerance (TOL). If DIFF is less than or equal to TOL, a new YN is accepted and the process is repeated, at which time a new average (AVG) is generated and stored. If DIFF is greater than TOL, the average (AVG) stored in the average register (AVGR) is transferred to the buffer register (BUFF), YMAX and YMIN are set to the value of YN, and AVG is recalculated as YN and stored in AVGR.

3. Transmitter Buffer

The ZOP and ZOI compression algorithms depend on a buffer to achieve efficient compression. The buffer must accept data at a maximum rate from active portions of the data, and it must have enough capacity to take advantage of periods of low activity. For TV, there is a period of zero activity during the vertical retrace time. This zero-activity period lasts 1/.5 line times, or close to 0.001 sec. During this time, about 7500 bits will be transmitted at the data rate that we will use for TV. The memory should have enough additional buffering capacity to absorb several TV lines at the maximum activity.

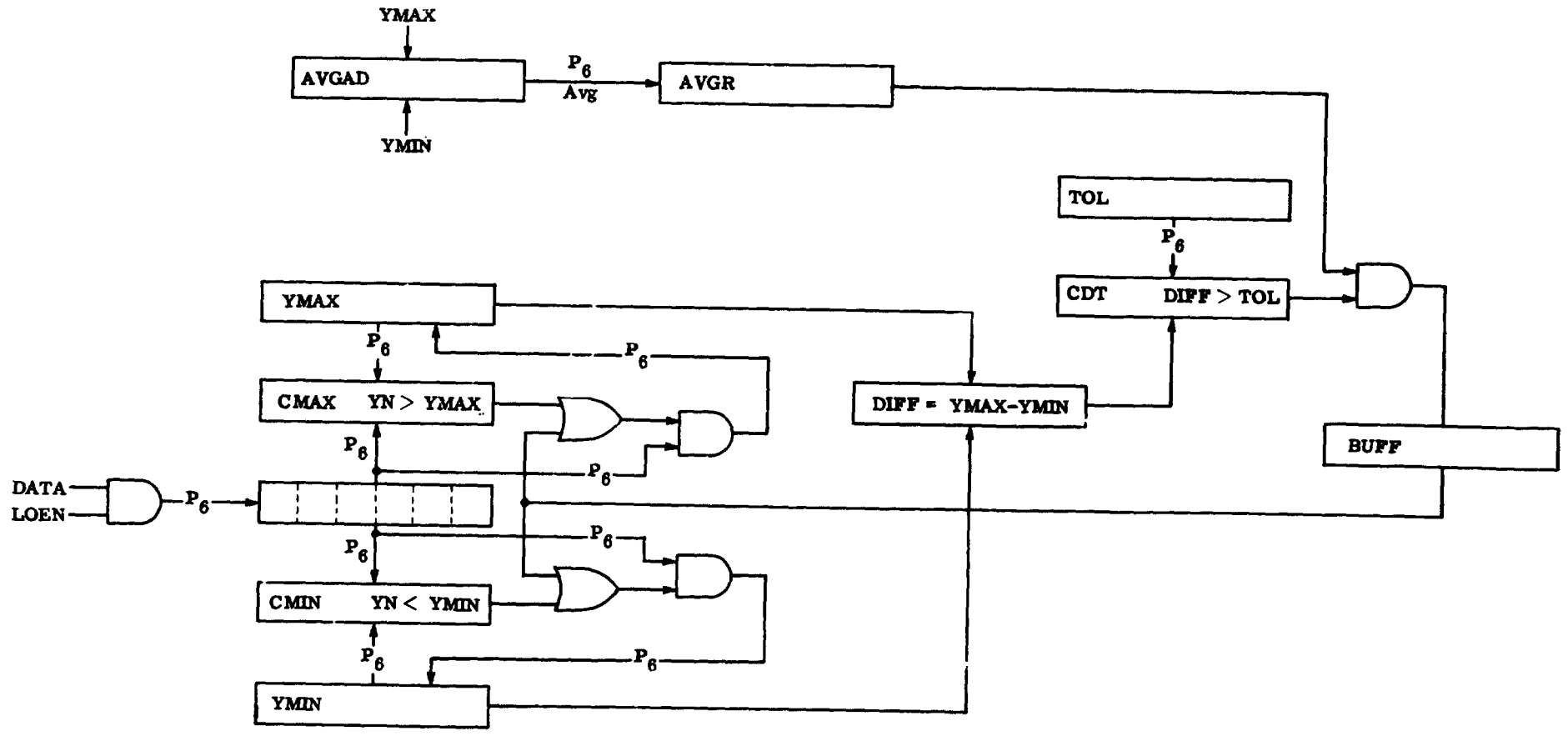


Figure IV-23 Zero-Order Interpolator Operation Flow Chart

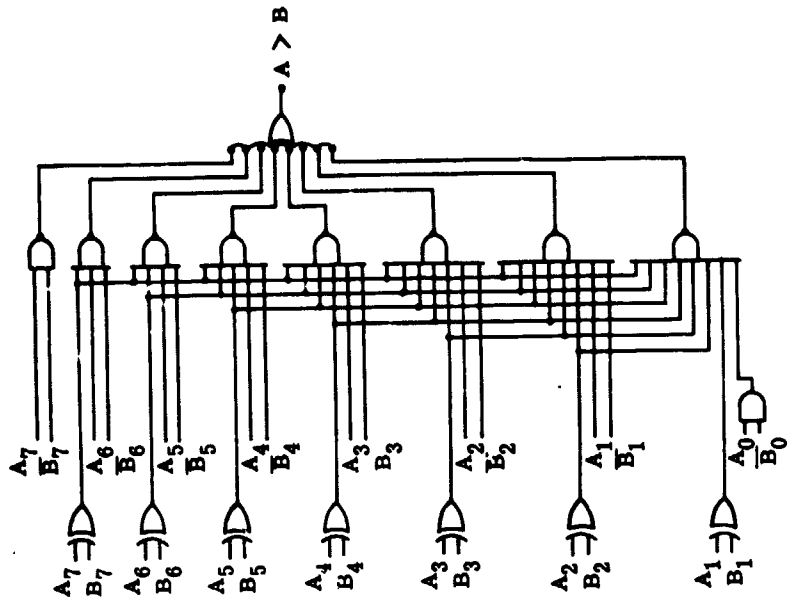


Figure IV-25 Comparator

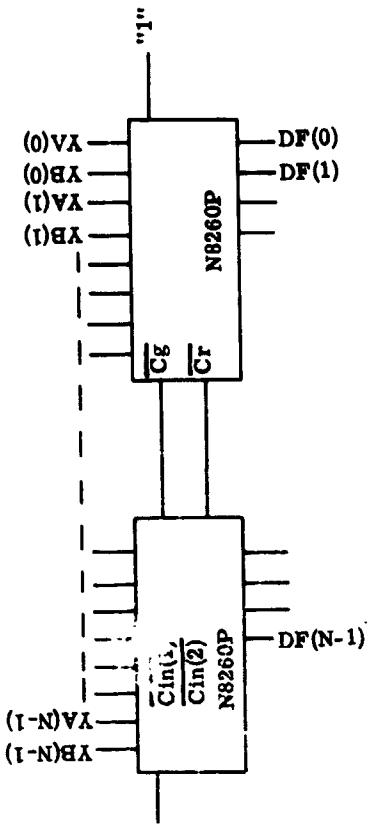


Figure IV-27 Difference Generator

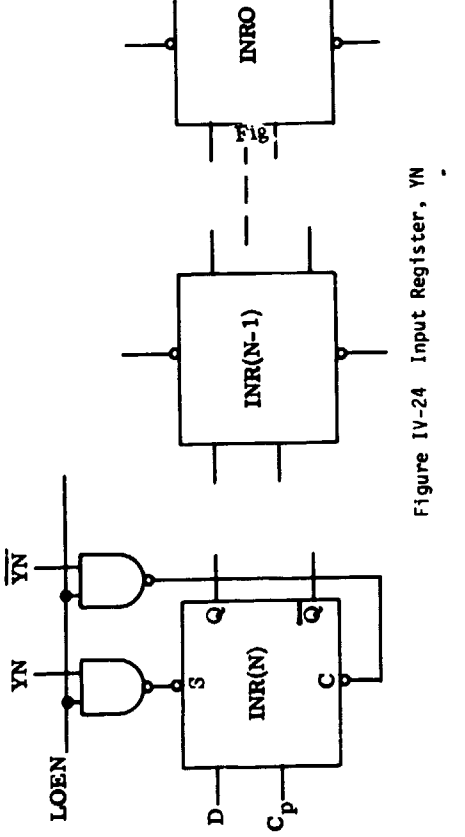


Figure IV-24 Input Register, YN

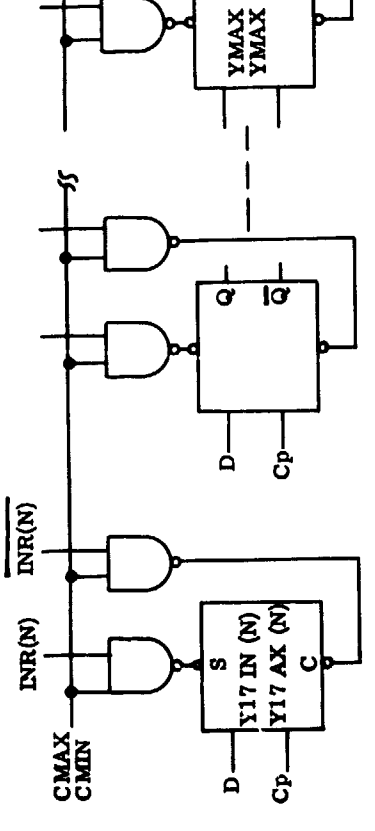


figure IV-26 Holding Registers for YMAX and YMIN

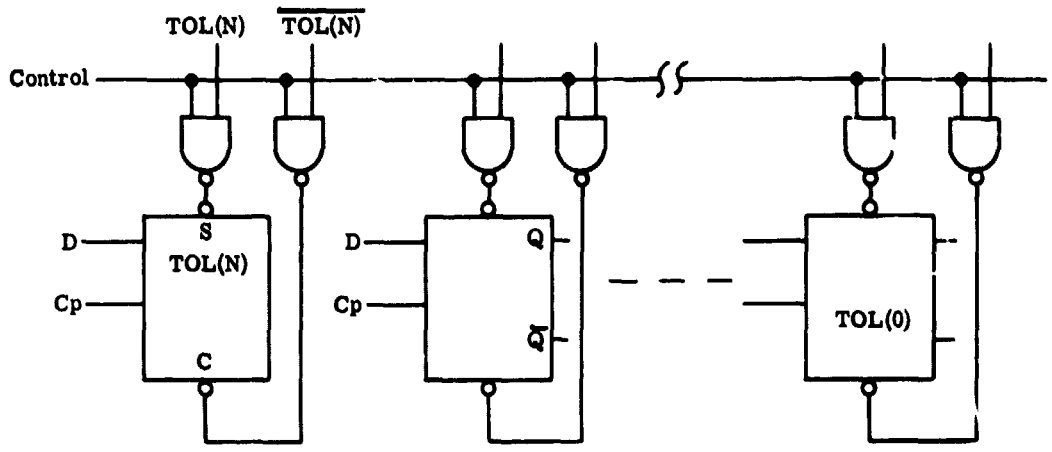


Figure IV-28 Tolerance Register

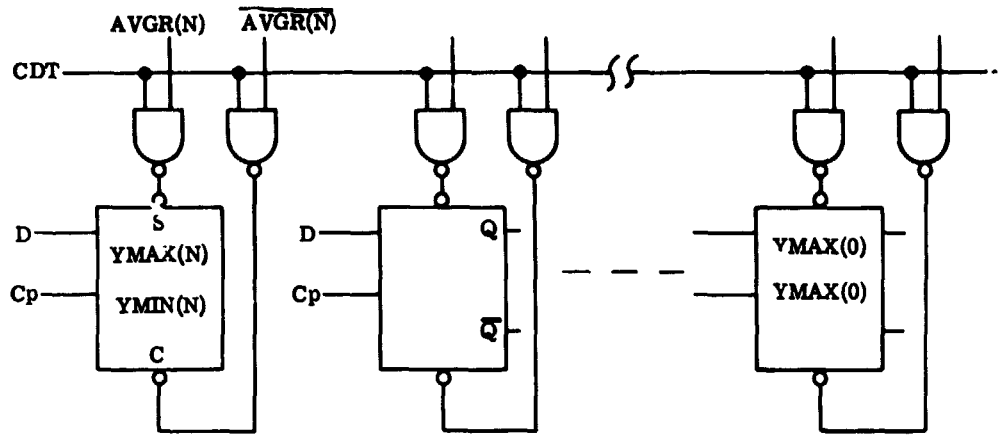


Figure IV-29 Buffer Register

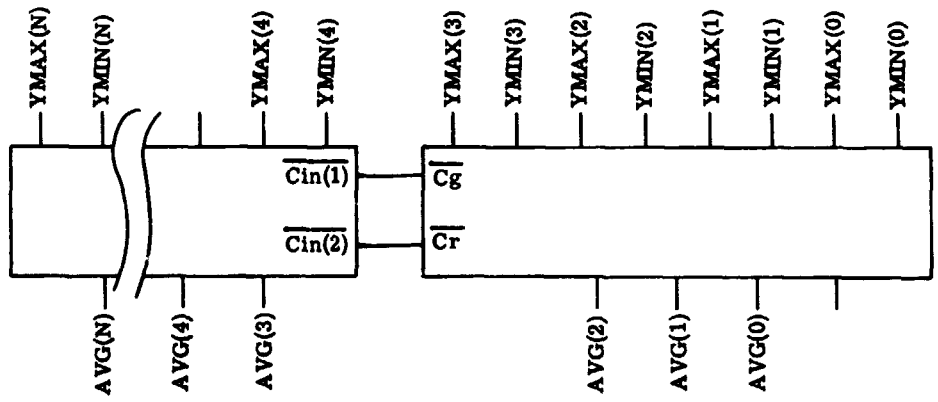


Figure IV-30 Average Generator

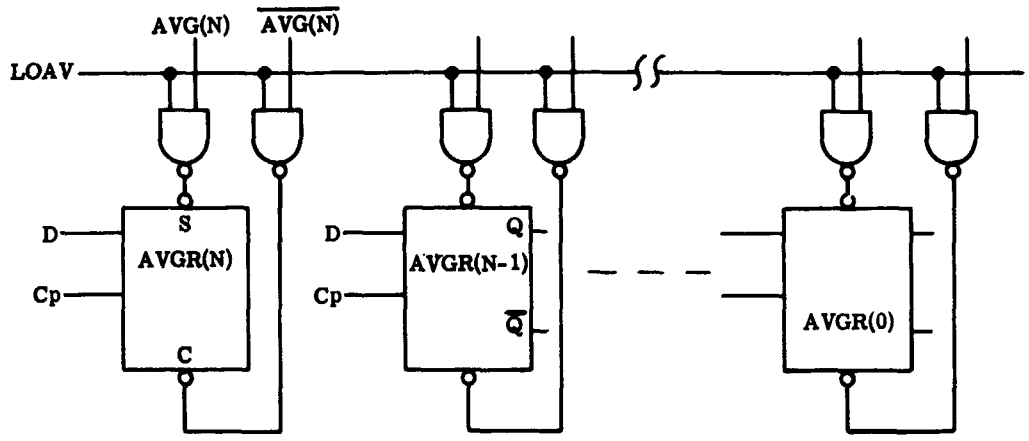


Figure IV-31 Average Register

For an actual spaceborne system, it might be economical to sacrifice some compression efficiency to reduce the size of the buffer; but, we believe it is important to be able to explore the range of parameters on both sides of the optimum values, and we will therefore use a 2048x45 memory as a buffer.

With a 1-μsec cycle time, the 45-bit word permits a maximum loading rate of five 9-bit TV pixel-words/μsec. The 9-bit word includes a 6-bit gray-level code and a 3-bit run-length code. The run-length code is of variable length, but when it is longer than 3-bits, the run length is at least five, and there is no problem with memory speed since no more than one word is transmitted for five pixels.

The 90,000-bit capacity is equivalent to 72% of a TV field and will be adequate to demonstrate the full possibilities of the compression algorithms.

4. Dithering

Horizontal dithering is the process of shifting the sample point within a sampling time according to a fixed cycling configuration. Its purpose is to minimize contouring and give the effect of a higher sampling rate. The second form of dithering we studied was intensity dithering.

Figures IV-32 thru IV-34 are function diagrams of the "dithering" section of the test set. Since all proposed dithering changes level on a field mark, a field counter is required. The sample period is divided into levels in a second shift register counter. Note that the clock to this counter should be at N (N = number of dither levels) times the desired sampling rate. A third counter is required for controlling the arithmetic in intensity dithering.

Control logic for the counters is single-level, horizontal dithering logic is two-level, and intensity dithering logic is shown as four-level (this could be reduced to two-level, but speed is of no real concern and it is debatable whether the increased cost of the components could be justified).

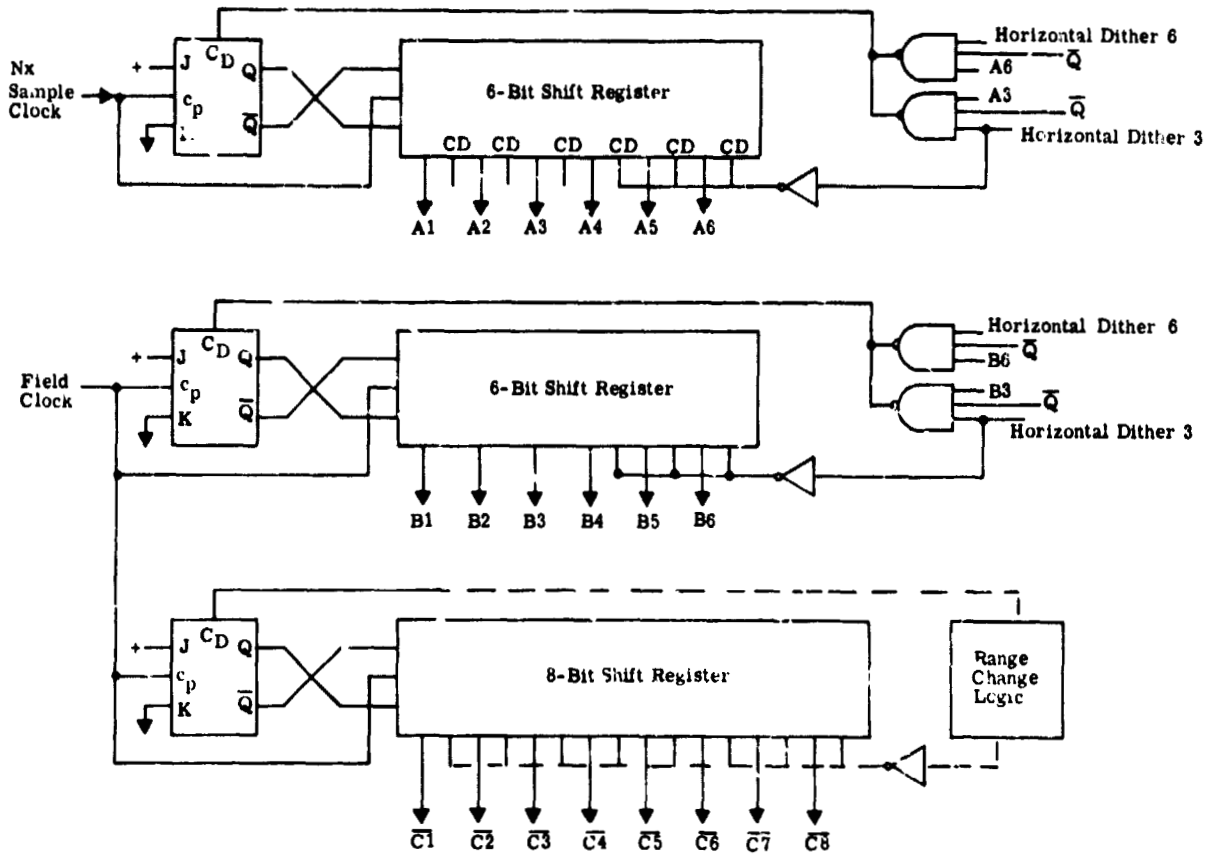


Figure IV-32 Horizontal and Intensity Dithering

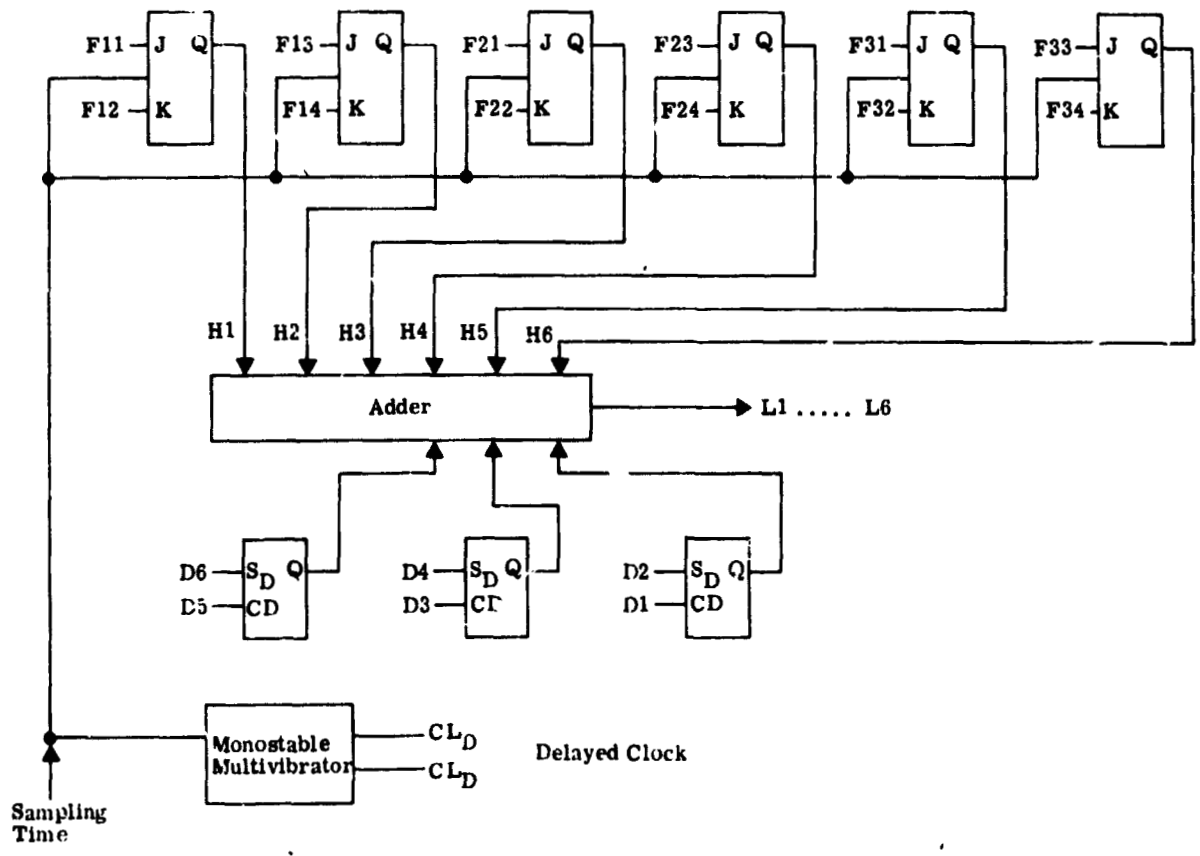


Figure IV-33 Intensity Dithering

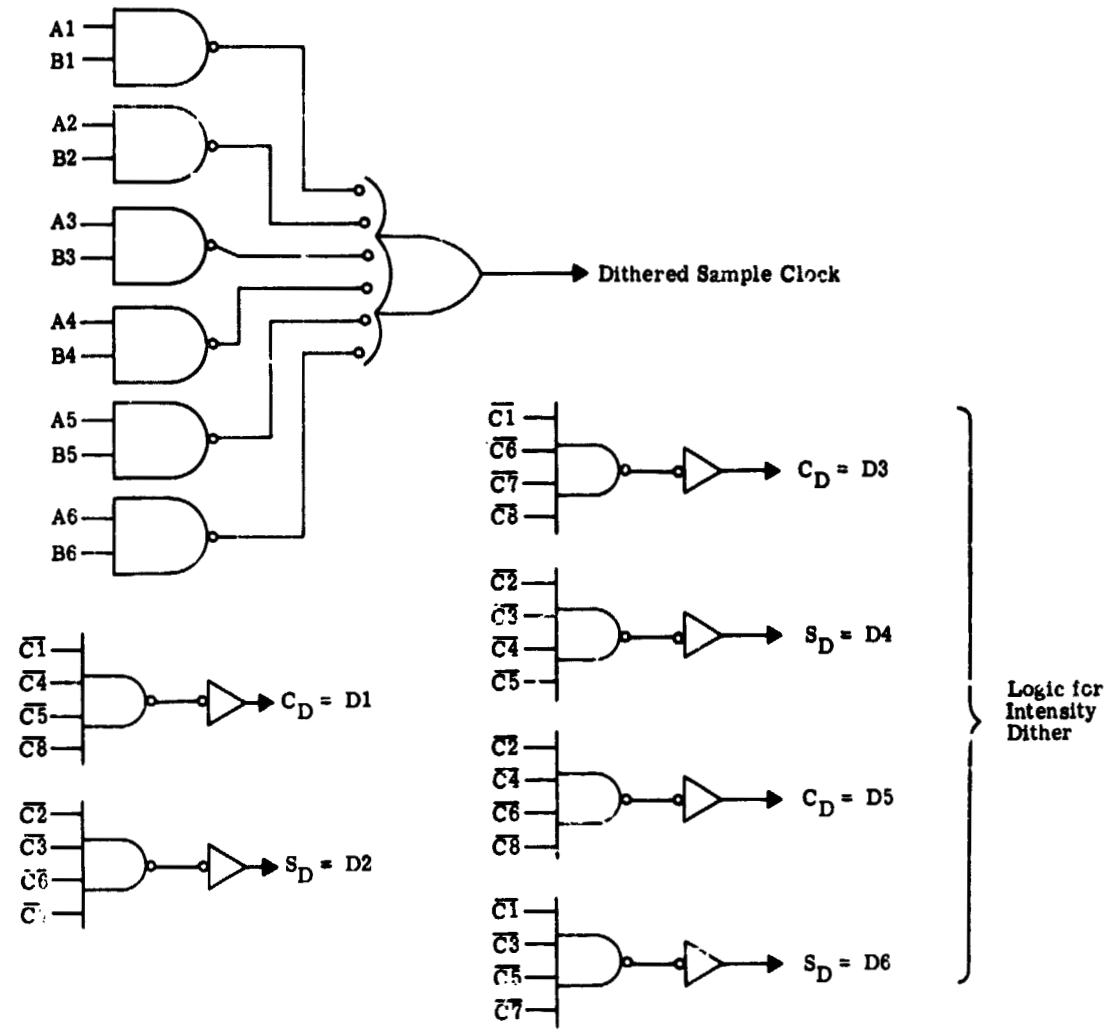


Figure IV-34 Dithering Logic

F. RECONSTRUCTION OF COMPRESSED DATA

Figure IV-35 shows the elements in the reconstruction section of the receiver. The bit stream from the Viterbi decoder is stored in the buffer at the constant rate of 7.5 Mbps, and it is removed at a variable rate to recreate the TV picture (or other analog data). Each word in the buffer indicates a gray level and the number of successive pixels that should be painted with this gray level.

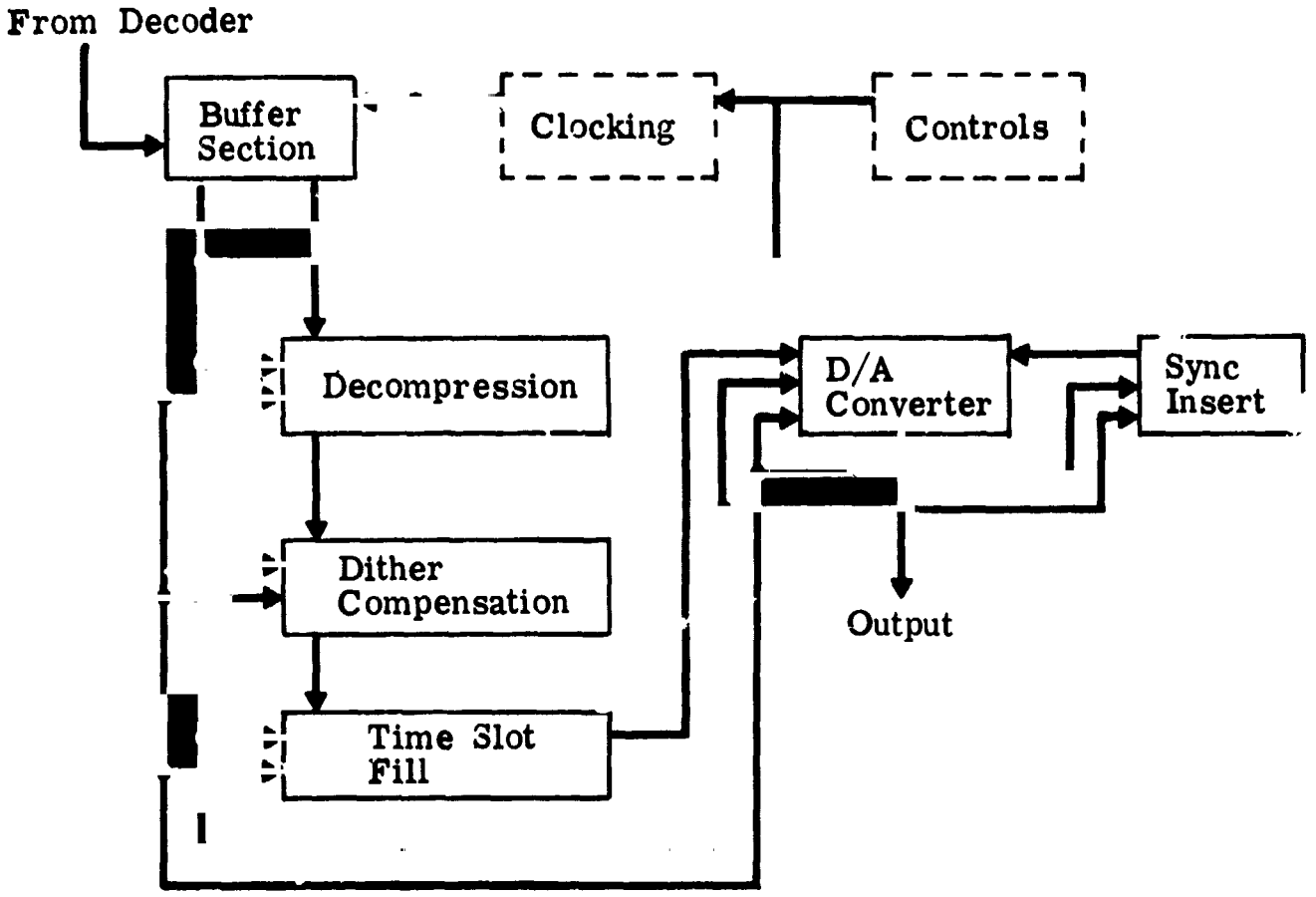


Figure IV-35 Reconstruction Section

The decompression module takes data from the buffer, recreates the indicated number of pixels at a fixed rate in pixels/μsec, and addresses the buffer to obtain more data when needed. Reconstruction is timed to duplicate the standard TV format, even for the line and field-retracing timer. Special code words are used to indicate end-of-line and end-of-frame so the decompression module can put the pixels in the right time slots after a channel error has upset the run-length coding.

These code words will be selected to prevent any confusion with the ordinary coding for gray level and run length. It is impossible to have more than 15 ones in succession, six for the gray level and nine for a maximum run length of 32. (This maximum is a design parameter.) The run-length code always ends in a zero. The end-of-line and end-of-frame codes will have a prefix consisting of 16 ones, and will thus be uniquely detectable.

The dither compensator removes the variable bias that was inserted at the transmitter to alleviate contouring. The synchronization insert module generates the line and field synchronization pulses that are needed for commercial TV receivers.

V. SYSTEMS DESCRIPTION AND HARDWARE

A. INTRODUCTION AND GROUND RULES

An all-digital spacestation-to-ground telecommunications system (less the RF link) is described herein.

The three data sources in the spacestation that output data to be processed into a single PCM data train are:

1) A TV camera with a commercial, black-and-white, interlaced-field format;
2) A 50.4-kbps PCM telemetry train;
3) Two 3-KHz analog voice channels to be converted to digital.

Before performing the main-stream multiplexing process (serial multiplexing of the four channels), a redundancy-removal-and-run-length-encoding operation is performed on the TV video data. The multiplexed data are then encoded for error detection and correction.

Performance ground rules for the system are:

1) TV picture quality shall be equal or better than Apollo TV quality for a 19-db rms/rms output S/N ratio;
2) BER for the 50.4-kbps PCM t lemetry data shall be 10^{-4} or better;
3) Voice channels shall have 90% word intelligibility and be equal to or better than Apollo voice at a S/N ratio equivalent to a 14-db rms/rms post-detection S/N ratio for the FM system;
4) The video bandwidth for the TV camera output is limited to 2.5 MHz and the RF bandwidth for the channel is limited to 12 MHz.

At the ground station, the RF channel is decoded and demultiplexed into the three data sources, and each is processed separately to recover the data.

Digital voice communications are converted to analog using a delta demodulator for each chennel. The 50.4-kbps PCM data are processed in the same manner as the data in the Apollo 51.2-kbps channel. The TV data stream is buffered, reconstructed digitally for processing in an A/D conversion, and synchronized to provide a composite analog TV waveform for transmission to commercial TV networks or stations.

The step-by-step system description that follows begins with the spacestation functions and components and ends with ground-station functions. Simplified system block diagrams are shown in Figures V-1 and V-2.

B. SPACEBORNE SUBSYSTEM

1. Data Sources

According to the ground rules listed in the previous section, data from four sources must be processed and multiplexed into a serial PCM data stream for transmission to Earth.

The two voice channels are individually preprocessed at analog to limit the band and to perform square-root-law syllabic compression. This is followed by a 1-bit delta modulation for each channel, sampled at a 25.2-kbps rate. This rate, which exceeds the minimum required estimate of 20 kbps, was chosen to simplify the multiplexing of the main stream, as will be shown later.

A third data source is assumed to be a 50.4-kbps PCM instrument data stream (science and engineering data), typically equivalent to the 51.2-kbps Apollo data channel. The 50.4-kbps rate was chosen to accommodate the main-stream multiplexing format, and legitimately falls within the 50 to 51.2-kbps limits for this source, as established in the ground rules for the system.

It is assumed that the 50.4-kbps source data format consists of 128 eight-bit words per frame.

The fourth, and most abundant, source of data is a TV camera operated at the standard commercially-broadcast, black-and-white TV frame, field, and line rates.

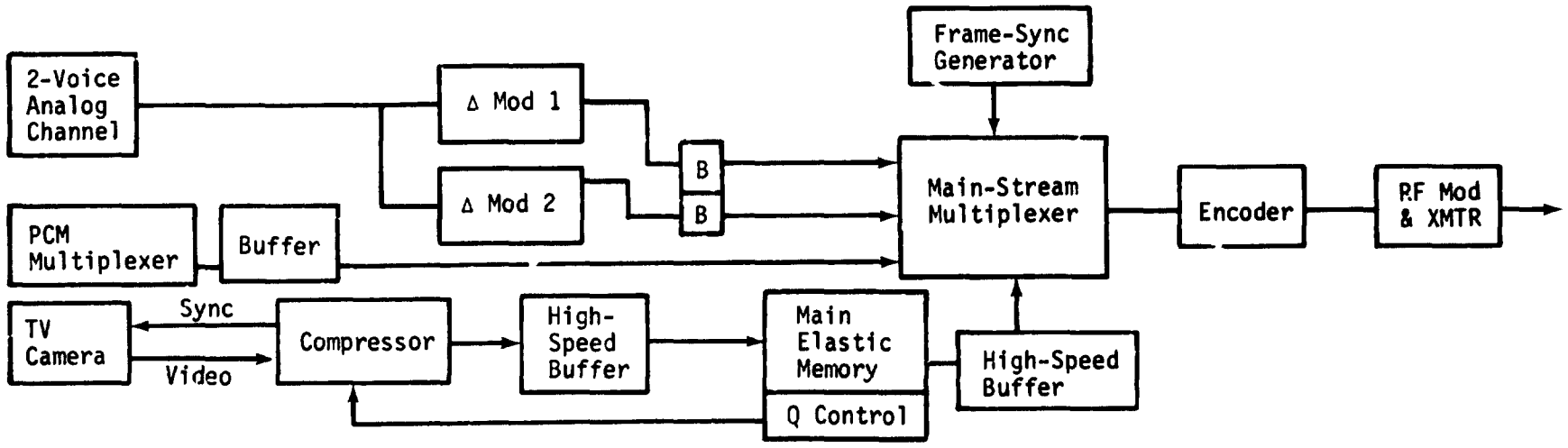


Figure V-1 Transmitting End Block Diagram

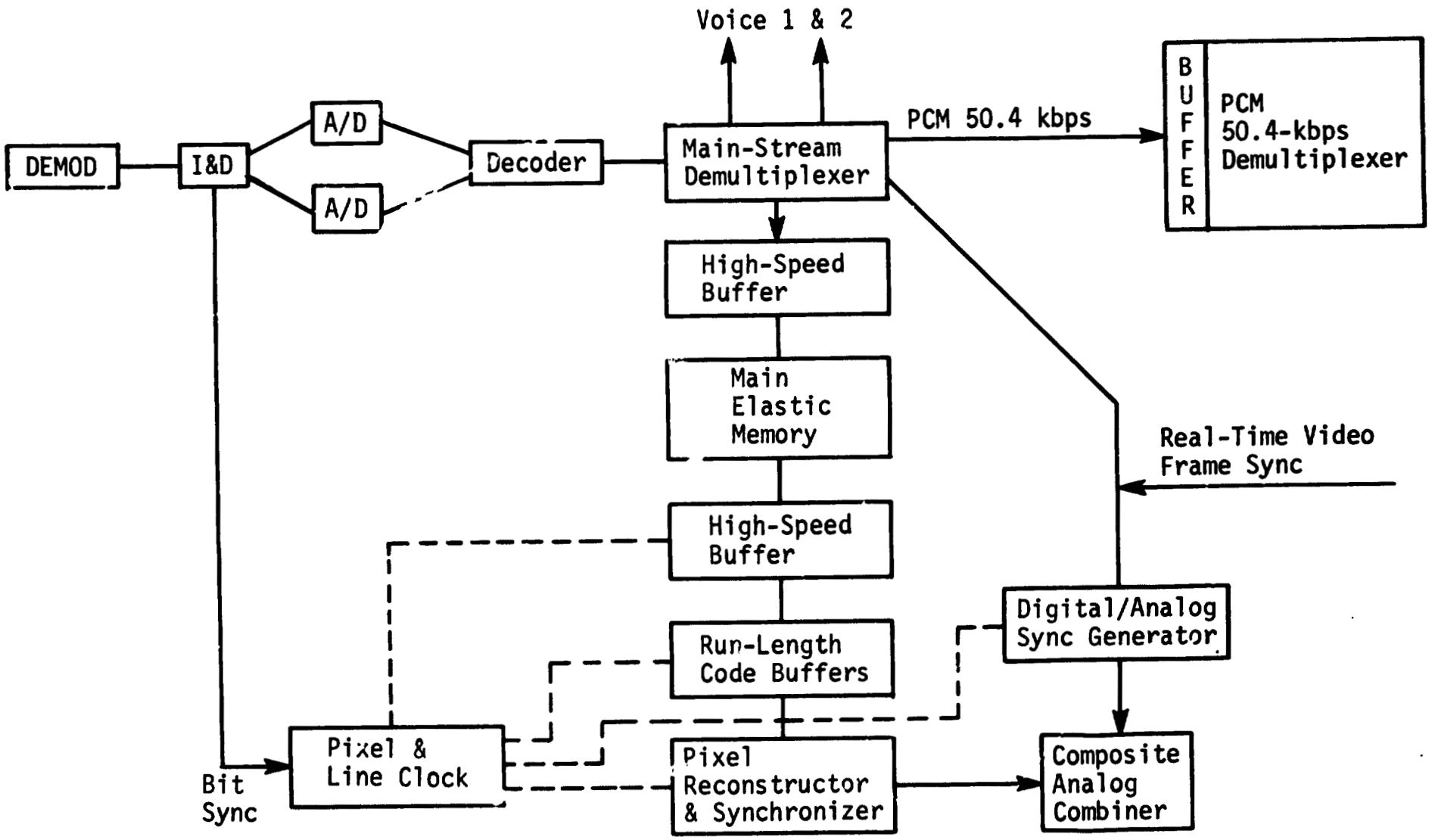


Figure V-2 Receiver End Block Diagram

Table V-1 lists some of the major characteristics of the standard system. The video bandwidth is constrained to be no greater than 2.5 MHz, which is, of course, somewhat less than the standard broadcast video bandwidth.

Table V-1 TV Standards*

525 lines/frame
60 fields/sec
2 fields/frame (interlaced)
13 to 21 inactive lines/field
249½ to 241½ active lines/field
*Camera scan blanking for line retracing and for vertical retracing can be generated internally and trimmed externally in the pixel sampling of the video (analog) since the line scan frequency and the pixel sampling frequency are synchronously related.

Camera-scanning synchronization is controlled from the space-station master clock, which supplies a 15,750-line synchronization rate, a 30-frame-per-sec synchronization, a 5.04-MHz pixel sample clock, and a 7.56-Mbps data rate clock, all of which are synchronous with the master clock, as indicated in Figure V-3.

Camera-scanning blanking for line retracing and for vertical retracing can be generated internally and trimmed externally in the pixel sampling of the video (analog) since the line-scanning frequency and the pixel-sampling frequency are synchronously related.

A fixed number of active lines per frame to be sampled and the number of active samples per line must be established to enable programing of the spaceborne data compressor and the ground-station reconstruction processes.

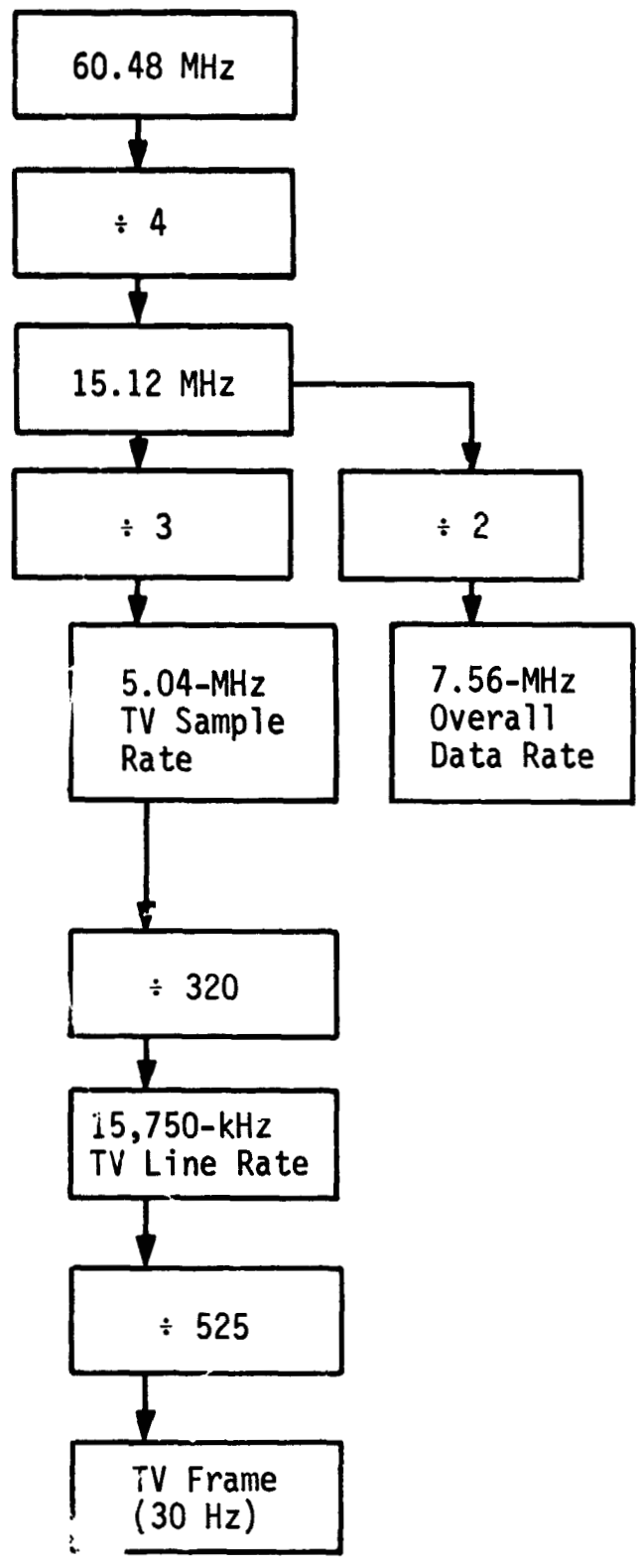


Figure V-3 Typical Station Clock Relationships

Of the 262½ lines per field, from 13 to 21 are inactive due to the interfield blanking (vertical blanking) pulse. In this report, 491 lines were assumed active per frame, or 245½ per field. Four lines in each field, were assumed blanked before the vertical retracing, three were assumed blanked during the vertical retracing, and 10 were assumed blanked following the vertical retracing.

2. TV Sampling and A/D Conversion

Two-hundred-seventy active samples per line were assumed from a total of 320 sample periods per line at a sample rate of 5.04M samples/sec.

Six-bit A/D conversion at this rate is accomplished by converting two bits in parallel for each module and rippling through three modules to obtain a six-bit conversion. This approach has previously been satisfactorily demonstrated at Martin Marietta, and is advantageous in that the A/D conversion is essentially continuous and the binary value can be read into a holding register whenever desired.

3. Redundancy Removal

The next step in the processing where dithering is not used is to remove redundant data using a zero-order interpolator (ZOI).

The zero-order interpolator (ZOI) transmits only the average of the maximum and minimum values of a number of samples that do not differ from one another by more than a tolerance of 2T.

The ZOI will be constructed of TTL micrologic elements. Maximum use will be made of MSI and LSI wherever possible. As with the ZOP, parallel logic ensures operation at the maximum data rate. The basic elements are an input register, an average generator, three comparators, two holding registers, a subtractor, and a buffer register.

The steps in the operation of the unit are:

1) Receive input bits YN;
2) Compare YN to MAX, compare YN to YMIN;
3) If YN > YMAX, set YMAX = YN, if YN < YMIN, set YMIN = YN;

4) Set AVG = $\frac{YMAX + YMIN}{2}$;

5) Subtract YMIN from YMAX = DIFF;

6) If DIFF > TOL, set AVG into BUFF;

7) Set YMIN = YMAX = YN.

The logical flow of these steps is shown in Figure V-4.

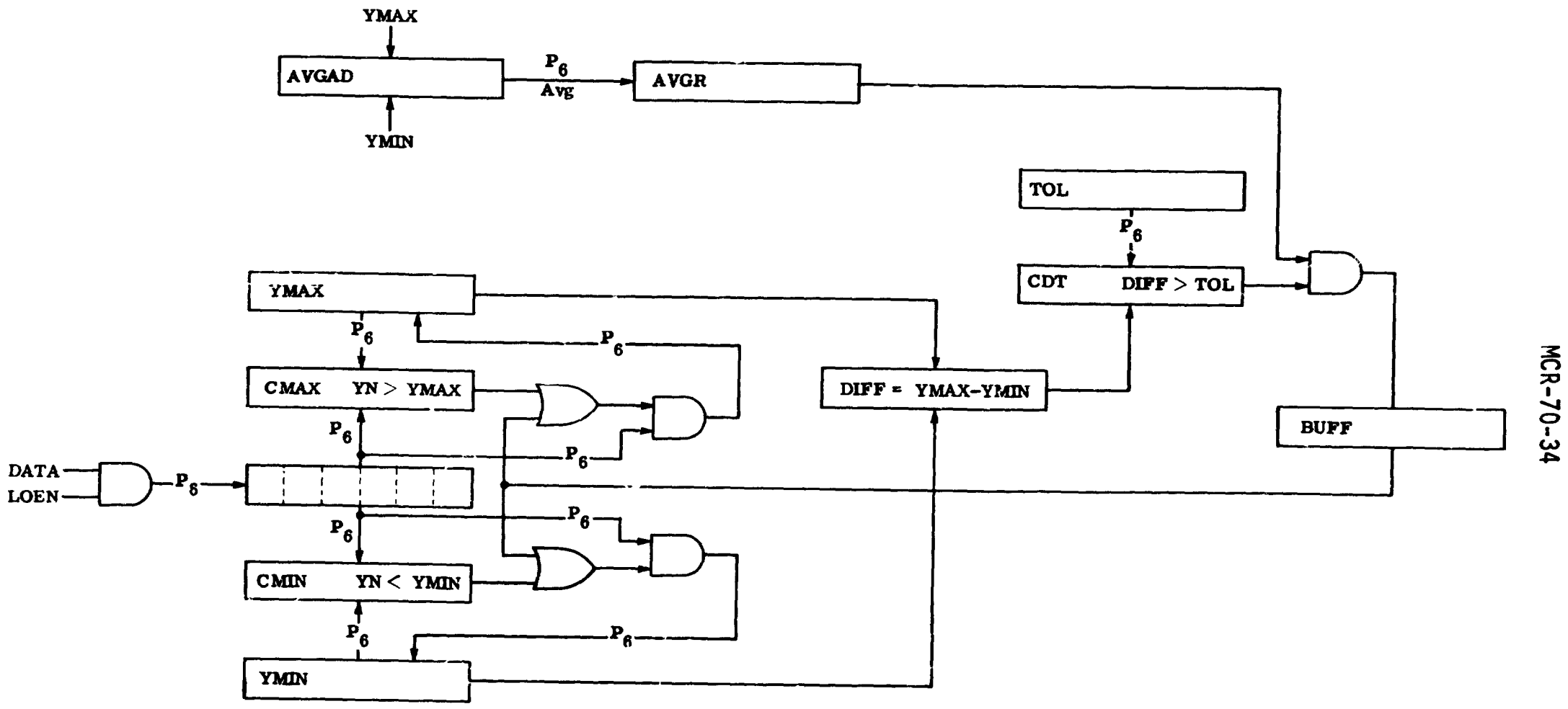
The ZOI hardware operates in conjunction with a main-elastic-memory Q control and a run-length encoder. Main memory Q is controlled by controlling the minimum and maximum run length and by controlling tolerance limits on the allowable amplitude spread for samples in any run length. Run length is defined as the number of sequential samples which meet the amplitude-tolerance value established for the ZOI algorithm, or, in a forced-sample mode, the number of quantized samples represented by a single-variable length code.

The run-length code chosen for illustration is based on an average run length of four samples. In using this code, the average sample-intensity level (represented by six bits) is followed by two bits designating run lengths of four or less, followed by a one for each overflow of the 2-bit run-length designator. A "0" comma bit then terminates the code for a given run length.

The codes used for a run length of three and for a run length of 20 (the maximum allowed) are shown in Table V-2, along with the codes used to indicate real-time end of frame synchronization, buffered end-of-line, buffered end-of-field, and buffered end-of-frame.

Table V-2 Codes Used for Compressed Data

RUN LENGTH	CODE
1	000
2	010
3	100
4	110
5	0010
8	1110
16	111110
19	1011110
20	1111110
End of Line	00111110
End of Field	01111110
End of Frame	10111110



MCR-70-34

Figure V-4 Zero-Order Interpolator Block Diagram

4. Main Elastic Memory (MEM)

Since output from the ZOI and the run-length encoder is sporadic and variable in word length (run length code) as well as in speed (45-Mbps maximum), buffering is required ahead of the main-memory loading registers. A typical arrangement is to use six 6-bit high-speed buffer memories to handle ZOI output, and six 7-bit buffer memories for the run-length code (RLC). These are loaded and unloaded sequentially into one or the other of two 45-bit main-memory loading registers. The number of bits transferred from the RLC buffers will vary from three to seven, depending on the run length. When a 45-bit buffer is fully loaded, its contents are transferred to the main elastic memory (MEM). Buffered end-of-line, field, or frame codes are also inserted into the MEM loading register(s) at the appropriate time.

Dual high-speed 45-bit output registers are used to unload the MEM a word at a time. A serial-by-bit output mode is used to supply TV data to the main-stream multiplexer from the high-speed (MEM) unloading registers (7.56-Mbps peak rate).

When one register is being unloaded serially, the other is being parallel-loaded from the MEM at a maximum rate of one parallel transfer per 5.95 μsec.

The MEM storage capacity is typically 45 x 2048 = 92,160 bits, or approximately 184 TV lines, assuming even distribution per line (92,160 ÷ 122,850 X 245.5 ≅ 184), where 122,850 is the average number of bits per field and 245.5 is the number of lines per field.

5. Main-Stream Digital Data Multiplexing and Formats

The main-stream data multiplexer must serially interleave data from each of the four data sources described, plus a main-stream frame-synchronization signal from a frame-synchronization generator.

The main-stream data format selected as typical is shown in Figure V-5. The frame-synchronization identification word (28 bits) is followed by an 8-bit word for Voice 1, an 8-bit word for Voice 2, two 8-bit words of PCM instrumentation (50.4-kbps data), and finally, by 2048 bits of variable-word-length TV data. Thus, the main-stream frame length is 2400 bits. The frame rate is 3150 frames/sec, which is 1/5 the TV line rate.

22 BITS	6 BITS	8 BITS	8 BITS	16 BITS	2340 BITS
Main Stream Data Sync	PCM Sub-Frame Count	Voice 1 Mod	Voice 2 Mod	2 Words of 50.4-kbps PCM	This is all TV. The first 45 bits of this portion is real time video frame sync. It occurs once each IOS of these frames; otherwise it is part of the compressed TV data.

----------2400 Bits/Frame----------

Frame Rate: 3150 Frames/sec

Data Rates:

3150 x 16 =	50.4 kbps
3150 x 8 bits =	25.2 kbps Voice 1
3150 x 8 =	25.2 kbps Voice 2
3150 x (45 x 52) =	7371.0 kbps TV Data Rate
3150 x 28 =	88.2 kbps Main Stream Sync & Count
Total:	7.560 Mbps Overall Data Rate

For TV, the video line rate, pixel sample rate, and video frame rate are all synchronous with the transmit bit rate and the main-stream data frame rate.

Figure V-5 Main-Stream Data Format

Since the TV frame-scanning rate is constant, but the number of bits per frame (after compression) is a variable, the buffered end-of-picture frame synchronization is nonsynchronous with the main-stream data frame. However, the real-time end-of-picture frame TV synchronization is synchronous with the main-stream frame because the real-time end-of-picture synchronization is inserted in the main-stream format as the first 45-bit word of the TV data once every 105 main-stream frames in order to give a video frame rate of 30 frames/sec. This real-time frame synchronization (it bypasses the TV data MEM) is used to synchronize the reconstructed analog video composite wave frame at the ground station.

The 50.4-kbps PCM subframe is synchronous with the main data stream since it is assumed that there are 128 eight-bit words per 50.4-kbps data frame. Thus, for every 64 main-stream frames, one complete 128-bit, 50.4-kbps data frame is received.

The two voice channels do not have synchronized words (other than bit synchronization) since the digital data are single-bit delta modulated.

The main-stream synchronization and ID word is composed of 22 synchronized bits followed by six bits to identify one of 64 subframes of 50.4-kbps PCM.

6. Channel Encoding

Convolutional Encoder - Convolutional codes are generally conceded to be better than block codes for most channels.* They are easy to implement, are less complex, consume less power, require less weight, and offer flexibility from the standpoint of decoding, particularly at high data rates. A binary convolutional encoder is shown in Figure V-6.

---

*I. M. Jacobs: "Sequential Decoding for Efficient Communication from Deep Space." *IEEE Transactions on Communication Technology*, Vol COM-15, No. 4 August 1967, pp 492-501.

D. R. Lumb: "Test and Preliminary Flight Results on the Sequential Decoding of Convolutional Encoding Data from Pioneer IX." *Conference Records*, IEEE International Conference on Communications, Boulder, Colorado, 1969, pp 39-1 to 39-8.

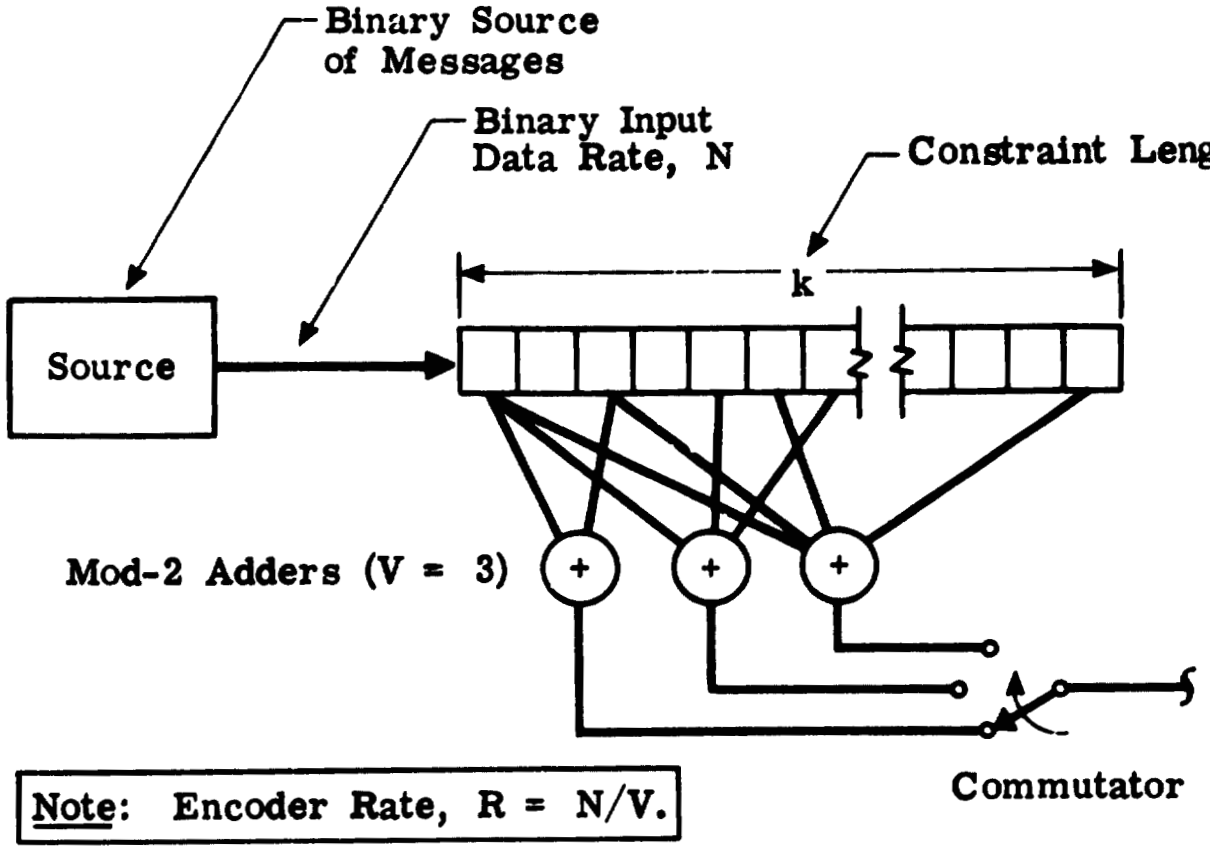


Figure V-6 Convolutional Encoder with Parameters

The stream of input symbols, such as picture elements or telemetry data, enters a K-bit shift register N bits at a time from the left. The V Mod-2 adders are each connected to a large number of stages of the shift register. (The particular connections can be specified by a set of special binary coefficients.) When a connection is made between the j^{th} Mod-2 adder and the i^{th} stage of the K register, the coefficient is 1; otherwise it is 0 for an open connection. The coefficients for the three Mod-2 adders shown in Figure V-7 are 1010000...000 for the first Mod-2 adder, and 10000101...000 and 1010010...001 for the last two adders. The rate of the encoder and decoder, R, is specified by N/V, and for the convolutional codes used for this project, N = 1. The Mod-2 adders are sampled in turn by the switch for each N-bit shift to form the encoded output message. The message vectors M are preceded by a register filled with zeros. The output sequence is denoted as X, where $X = \left(x_1, x_2, \ldots x_{t/\tau}\right)$.

For the system described here, a rate ½ convolutional encoder having a constraint length of five has been selected. A typical set of connections for the encoder is shown in Figure V-7.

The resulting code characteristics and expected performance are discussed in Subsection 1 of Section A, Chapter II.

Output from the encoder is parallel (two bits at a time) to the quadrature phase modulator.

Communications Link - The communications link is undefined as to frequency band; however, it is assumed that when quadrature phase modulation of a carrier is employed, it conforms to the RF bandwidth allocation of 12 MHz.

For a ½ rate system, the convolutional-encoder output rate is twice the input rate (2 x 7.56), or 15.12 Mbps. The channel symbol rate for quadphase is, therefore, 7.56×10^6 symbols/sec, or 15.12 Mbps.

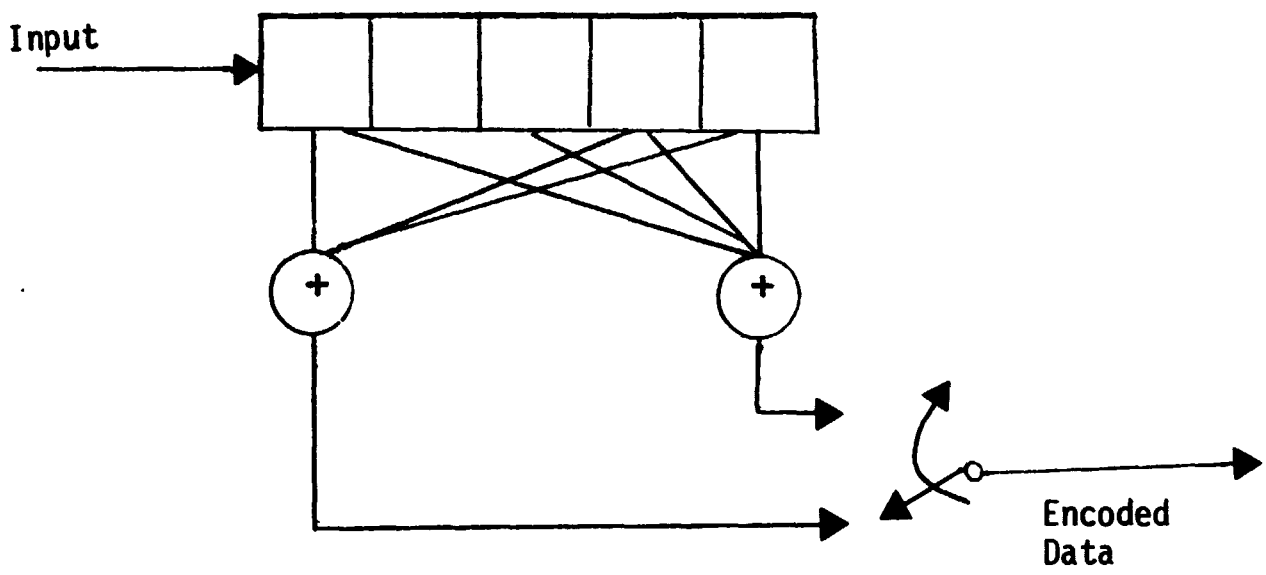


Figure V-7 Connections for Encoder

C. GROUND STATION EQUIPMENT

1. Signal Detection and Quantizing

For quadraphase demodulation, two channel bits are represented by a single symbol; therefore, in the detection process, simultaneous outputs from two phase detectors operating in quadrature provide the signals to be integrated, dumped, sampled, quantized to three bits, and fed to a high-speed Viterbi decoder. Due to the high symbol rate (>7.5 Mbps), two integrating and dumping circuits are required ahead of each A/D converter. These circuits operate alternately in an integrating mode, and a continuous sum of their outputs is supplied to the high-speed A/D (which is identical to that used in TV A/D conversion), as shown in Figure V-8. Symbol synchronization is derived from a VCO locked to either of the incoming bit streams. Each stream is integrated over a symbol period that last for two bits.

2. High-Speed Viterbi Decoder

The Viterbi decoder matches the spacestation encoder constraint length of five, and the rate of ½, but instead of making a hard decision on each pair of bits received, the analog representation of each is quantized to three bits before decoding. Output from the decoder is delayed about five constrain lengths from the input symbol rate, and appears serially, a bit at a time, for demultiplexing.

The following presents a brief functional description of a decoder and methods of selecting good codes.

The block diagram of Figure V-9 shows the Viterbi algorithm implemented as parallel hardware systems for K = 3, and V = 2. K = 3 and V = 2 are chosen for simplicity of drawing, and can be expanded by parallel addition of adders, comparators, and registers.

The parallel approach has been selected to take advantage of its speed, but necessitates more hardware than a more serial system would use. The block diagram is presented as a single-line flow diagram in which type of transfer and number of bits are indicated on the line [P(N) = parallel transfer of N bits]. The symbols on the blocks are:

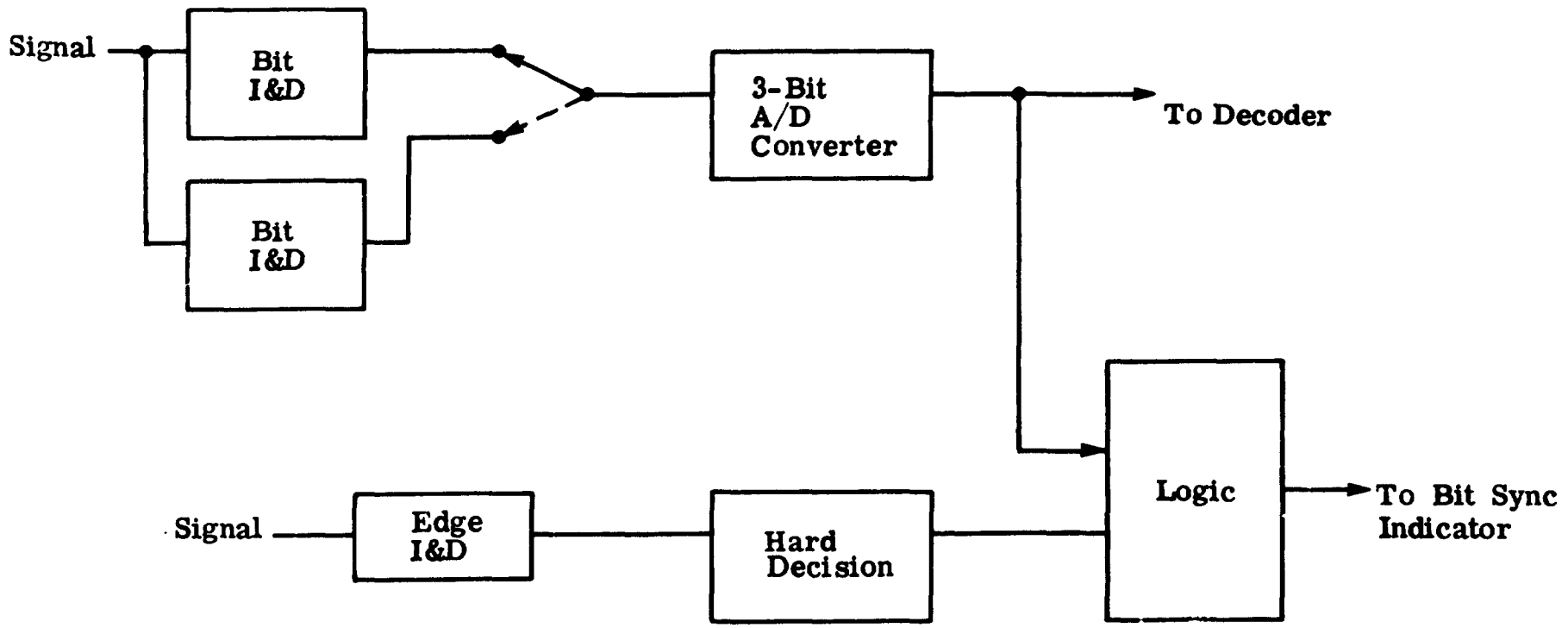


Figure V-8 Ground Side Input

1) I&D = integrating and dumping;
2) ΔS generator = the score increment generator;
3) A_{xy} = an adder receiving data from the S_x register and presenting its sum to the S_y register;
4) C_y = a comparator that generates the load command for the S_y register;
5) S_y or S_x (same if x = y) = the score accumulator;
6) O_y = the output "tale" register.

The I&D circuit contains an integrator to extract the signal from the noise picked up during transmission and receiving, and an analog-to-digital converter that takes the analog level of the channel symbol and converts it to a 3-bit binary number.

The ΔS generator receives six bits (three bits for each of two channel symbols) from the I&D and converts these to four 4-bit score increments. The first operation is to take the two 3-bit numbers (A and B) and subtract each from 7_8, which in effect creates A, B, A´, and B´. These are then taken in pairs and added to yield A + B, A´ + B´, A´ + B, and A´ + B´ (where the + operator designates a SUM, not a logical OR). This operation is implemented by four 6-input-bit circuits, which are presented as AB, AB, AB, and AB, and are used to generate ΣD, ΣC, ΣB, and ΣA. This circuit arrangement is shown in Figure V-10.

There are two 4-bit A_{xy} adders which total eight bits. These adders receive their addends from the ΔS generator and the S_x register, and present their sum to the C_y comparator for selection, and to the selection logic for the S_x registers. The logic for this is shown in Figure V-11.

The C_y comparators are 8-bit logical comparators that operate in a parallel, combined mode. The comparator puts out a signal indicating whether the A_{xy} or $A_{(x+1)}$ sum is the lesser of the two, and steers this sum to the inputs of the S_y register. The output signal mnemonic is $T_{xy} = T_{(x+1)y}$, which is also used by the θ register for transferring control. Comparator logic is shown in Figure V-12.

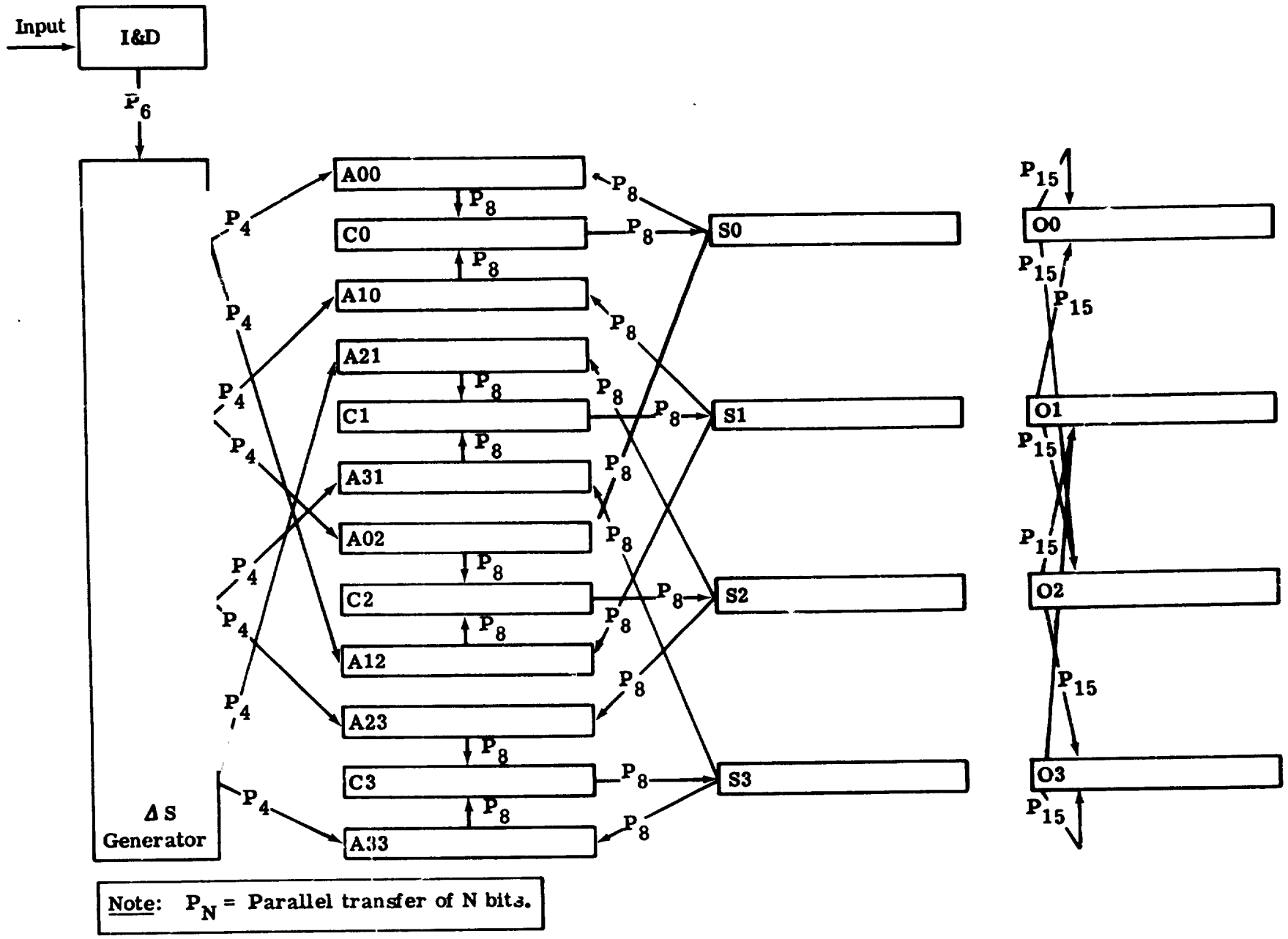


Figure V-9 Parallel Viterbi Decoder

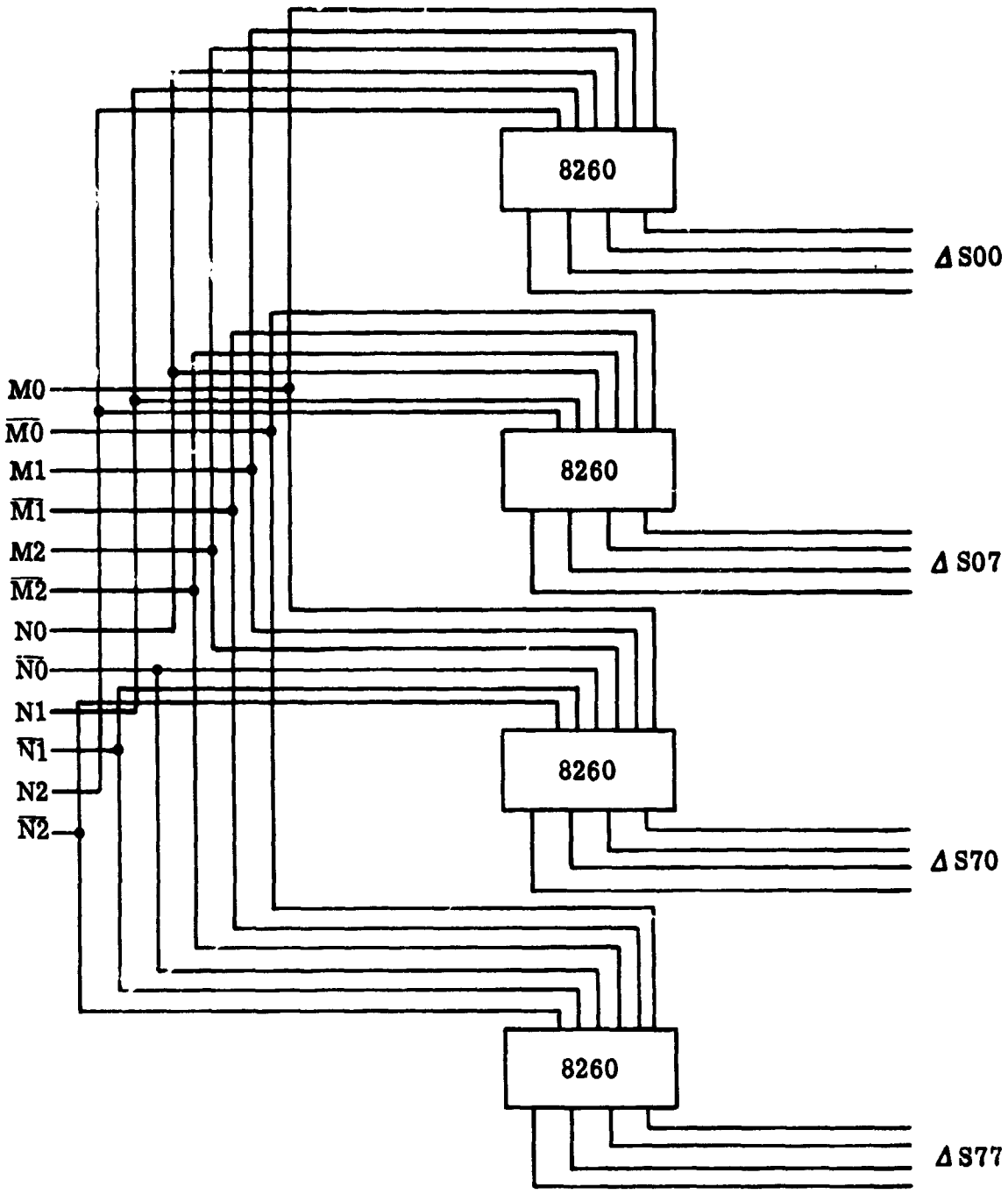


Figure V-10 ΔS Generator

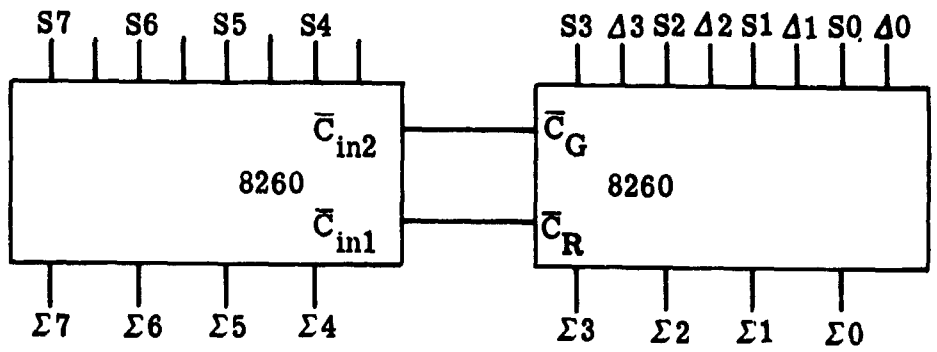


Figure V-11 Viterbi Adder

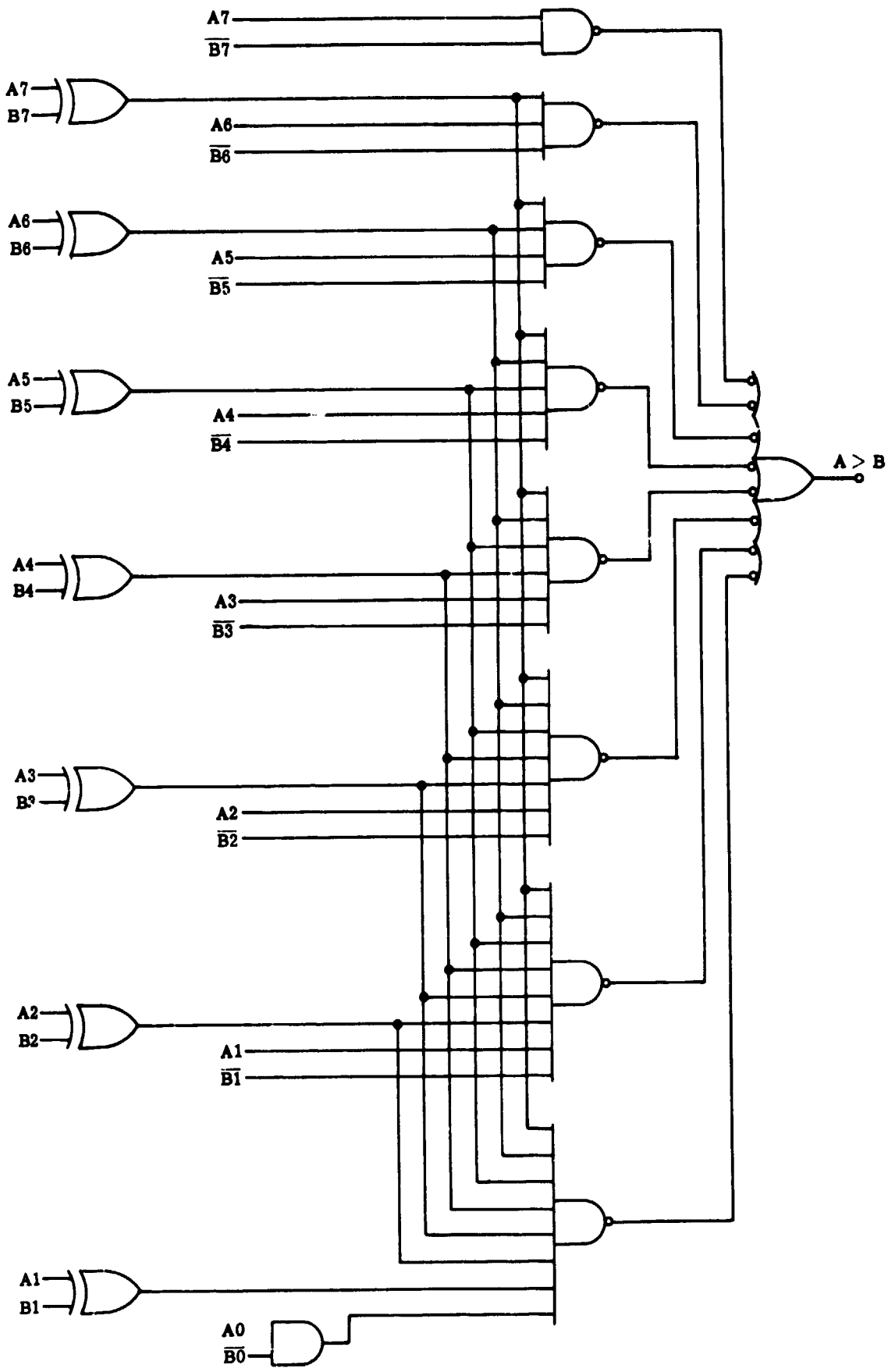


Figure V-12 Comparator

3. Main-Stream Digital Demultiplexer

The main-stream digital demultiplexer (after locking on to the main frame synchronizer) separates the main stream of data into three separate channels for further processing.

Only a frame-synchronization code is used for main-stream synchronization. Standard 3-mode synchronizer acquisition and lock-on procedures are required (search, test, and lock). A subframe synchronization code for the 50.4-kbps data is contained in the first three 8-bit words of the 50.4-kbps frame, which is subsynchronous with the main frame (1:64). The compressed TV telemetry frame length varies and is nonsynchronous with the main-stream frame. A real-time end-of-TV-picture frame, plus delayed end-of-picture frame, field, and line synchronization code words form the basis for TV reconstruction and synchronization.

Voice channel bit synchronization occurs automatically with the main-stream frame synchronization and, of course, for a bit Δ-modulated signal, no other synchronization signal is required.

4. TV Reconstruction and Synchronization

The compressed TV data are reconstructed in reverse sequence to that of the compressors in the space station. Since the uncompressed TV frame rate, TV line rate, and pixel rate are synchronous with the main-stream bit rate, the analog synchronization train can be generated in synchronism with the real-time frame synchronization, giving a constant delay (except for Doppler) between the space-station scanning waveform and the reconstructed analog synchronization waveform.

Proper reconstruction of pixel data then is a matter of decoding the brightness level and run length, as well as matching up the delayed end-of-line, field, and frame with the reconstructed analog synchronization waveform.

The serial TV output from the main-stream multiplexer alternately fills two 45-bit high-speed buffers, which are alternately dumped (45-bit-word parallel) into a main elastic memory (MEM) at a ground station. Output from the memory is a 45-bit word that is bit-paralleled alternately into two high-speed unloading buffers. They are alternately unloaded sequentially into six 13-bit pixel-value and run-length code registers, which are emptied sporadically at the pixel rate during the reconstruction of each line. Recall that the run-length code word has a variable length and,

therefore, that the number of bits transferred to each 13-bit holding register varies with run length; consequently, the length of the transferred word length varies from nine to 13 bits. Recognition of the end-of-run length code, as well as that of the special end-of-line, end-of-frame, and end-of-field codes, must take place during continued search of the MEM unloading registers to enable a logical transfer of the correct number of bits to each 13-bit holding register.

Initial TV synchronization is obtained (following main-stream format frame synchronization) by recognizing the real-time TV picture frame synchronization and by synchronizing the analog synchronization waveform generator to this. (This synchronization process is exactly equivalent to synchronizing any subcommutated channel.)

The buffered-run-length data are allowed to fill the MEM, and are then clocked out at a 45-Mbps rate until a buffered end-of-frame is recognized. (Buffer Q control can prevent emptying this MEM if an end of frame synchronization is missed.)

The MEM output is halted at an end-of-frame synchronization condition until the next real-time end-of-frame synchronization time is used to start the pixel reconstruction process.

5. Ground Rules in TV Sync and Reconstruction when Errors Occur in the Data

Coding errors in run length and in the delayed end-of-line, end-of-field, and end-of-frame must be handled logically to maintain synchronization or to regain synchronization with as little loss of data as possible. This logic follows:

1) Errors in the real-time TV picture synchronization - Errors in the real-time TV picture synchronization can be handled like those in any subcommutated PCM channel. A search, check, and lock mode will be used when a search for the synchronization pattern occurs every frame (the first 45-bit TV word) until it is found and when this search is repeated every 105 frames. When the error rate in the synchronization code is low enough, a lock-on mode is enabled. One can usually set a threshold for errors in the synchronization code before starting a new search;

2) Errors inside a line (run length) - A line may be short or long with regard to the number of pixels. If the line is short, revert the mid-grey level until reaching the end-of-line time, and then begin a new line. If the line is long, ignore the extra pixels and begin a new line. If the next end-of-line code cannot be found in time, then begin a new line at the mid-grey level and fill each line until the next end-of-line code is recognized. The line will now be resynchronized but will be offset vertically.

   Depending upon the type of run-length errors, one may be either in synchronization or one or more lines behind the output synchronization waveform at the end of the field. If the end-of-field (or end-of-frame) code is not found (if one is behind a line), discard data during blanking at the end of the field or frame by going to the next end-of-line (or end-of-field) code when reaching the end of the frame.

3) False end-of-line code - If a false end-of-line code occurs, handle this the same as a short line, except that at the end of the field this system will be a line behind, and one must discard a line during the between-field blanking period by going to the next end-of-line (or end-of-field or end-of-frame) code.

4) Missed end-of-line code - If an end-of-line synchronization code is missed, terminate at an end-of-line pixel count and discard the remainder. Begin the next line at the value set by the first run-length code which follows the next end-of-line synchronization code unless the next end-of-line synchronization code cannot be pulled from the buffer in time. In this case, begin the next line using the mid-grey scale, and terminate at the correct pixel count. Continue filling the screen until the next end-of-line code is recognized.

   Under this condition (missed end-of-line code), the system may be either in synchronization or be one line high on the screen. When the system recognizes the end of a field or frame and the system is ahead, stop unloading from the buffer and fill the screen until reaching the time to begin the next field or frame, as the case may be.

5) Missed end of field code - If an end-of-field (frame) code is missed, terminate the line at the correct pixel count and go to the next recognizable end-of-line code to begin the next field.

This will result in the system being one or more lines ahead in the next field. Follow the same procedure at the end-of-field code as described under missed end-of-line code where the system is ahead one or more lines.

If Buffer Q gets out of tolerance, resynchronization must be accomplished.

APPENDIX A

AN ADDITION TO FORTRAN TO SIMPLIFY PROGRAMING OF ALGORITHMS OR SIMULATION OF HARDWARE

During programing of both Viterbi and Fano algorithms it is apparent that Fortran, which is basically a computational language, is not well suited to this work. These programs use mainly logic instructions such as *and*, *or*, and *exclusive or*, and shifts both logical and circular as the bulk of the program. In order to do these operations in Fortran, it is necessary to supply subroutines to perform some or all of these instructions on most of the machines. The result is that the programing itself is difficult, hard to follow, and execution is slow. In order to speed up the execution, it is necessary for the programer to either recode, in a form of machine language, or use unorthodox techniques to avoid the use of the logical set and shifts. These techniques also must be varied for each application, and thus they are not easily used by the engineer who does not constantly do this work.

It appears that addition of several instructions to the normal Fortran set could enhance the suitability of Fortran for this work. First, it must be recognized that for maximum gain in the use of these additions, the capability would have to be coded into the Fortran complier. Also usable, but less effective, would be the coding of some of the instructions as subroutines in the machine language of the particular computer to be used. Least effective of all, but at least making the program much easier to understand and to code, would be to program these instructions as Fortran subroutines.

The greatest single difficulty encountered in use of Fortran is the inability to easily simulate a register or shift register of n bits long, and have any freedom of complication if the number of bits in the register does not match the number of bits in the computer word. An "instruction similar to the Fortran *DIMENSION* instruction. For example, *SIZE* could indicate any variable that is used to simulate a hardware register, and the number of bits it would contain would also be specified. It would then be up to the complier to set up the mechanism so that when this varible is used it is to be of n bits long, as opposed to a fixed computer word. The needed additional instructions to make use of this capability would be: (1) e.g., *OR VAR1 (26,29), VAR2 (21,24)* whose use would result in bits 26 through 29 of the variable VAR1 to be bit by bit inclusively OR'd into bits 21-24 of VAR2; (2) *AND.....)* same as above except the logical and; (3) *EXOR.....)* same as above except exclusive or; (4) *MOVE VAR1 (2,4), VAR2 (7,9)* which would simply move bits from one variable to another; (5) *SHIFTL VAR4 (-8)* which would shift logical (with automatic zero fill of vacated positions 8 bits to the left, where negative members would indicate left and positive to the right, and *SHIFTC VAR4 (22)* which would treat the register as circular (those bits falling from the right side of the register would be fed into the left end).

APPENDIX B

SMALL COMPUTER USE FOR DECODING

In the course of programing the simulation of the Viterbi algorithm on various computers, an obvious question arose as to whether specialized hardware is necessary as opposed to having the computer simulation actually act as a decoder. Although it is not practical on the Apollo USB, it could be of significance in an application when an on-board computer is not being fully used and could perform the decoding function in a background or time sharing mode.

The present state-of-the-art of small computer execution speed is controlled by the access-time-to-memory, with these memories being not significantly slower than those of large computers. Therefore, the execution speed is mainly affected by differences in the suitability of the machines instruction set for performing the algorithm. In the future, electronic memories (without magnetic data retention) may make actual decoding at higher speeds a reality. (See Figure B-1)

The main loop of the Viterbi Algorithm can be programed on a computer with a powerful instruction set in 65 instructions, this loop being executed 2^{K-1} times for each information bit. Some adjustment of this loop length is possible if core requirements are relatively unrestricted, as when tables for scanning are more expanded, the instructions involving lookup from the tables can be reduced in numbers.

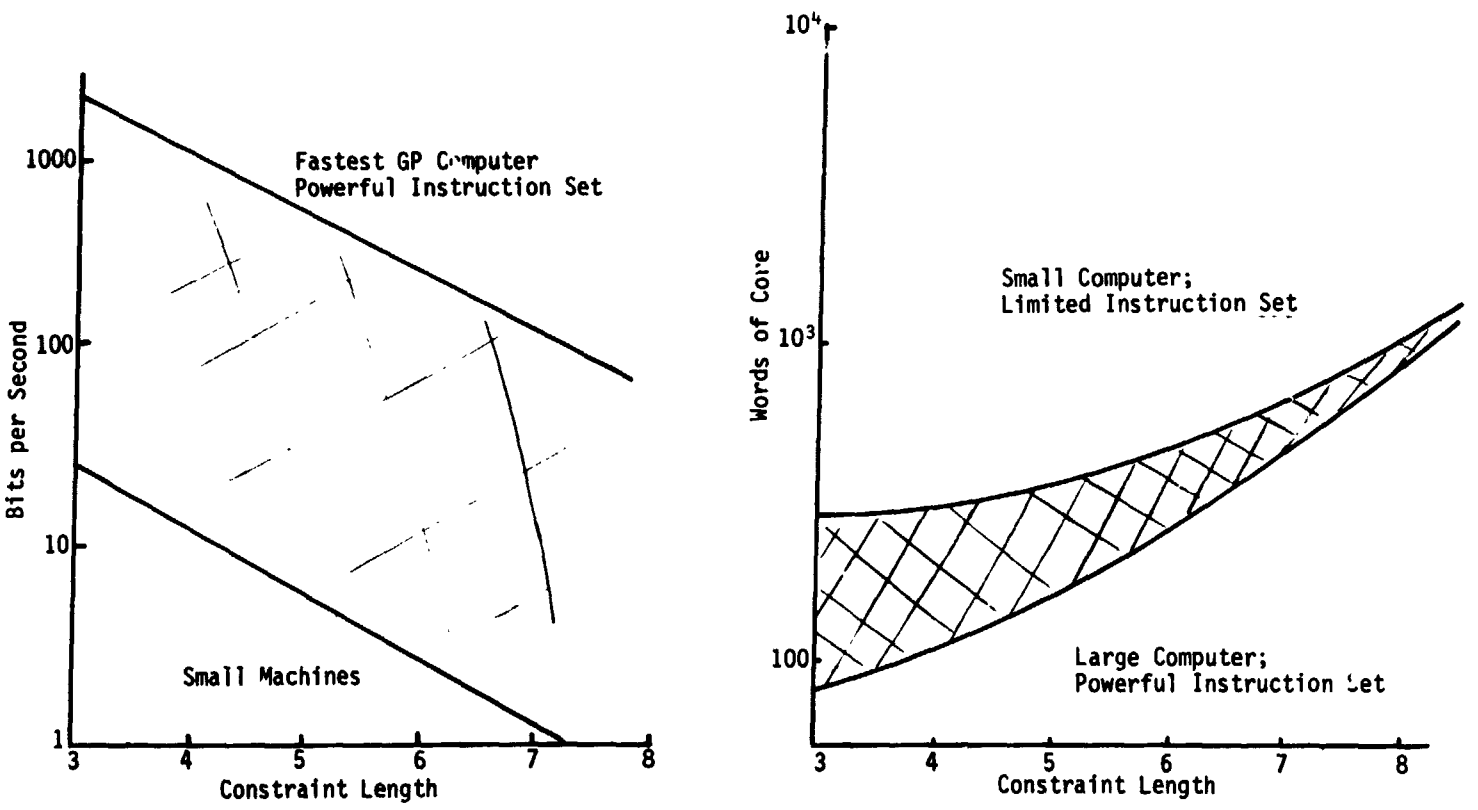


Figure B-1 Core Requirements and Speed on Small Computers

APPENDIX C

THE VITERBI ALGORITHM

The Viterbi algorithm is a maximum likelihood decoder that acts on the principle that a shift register can proceed from one state to only two others; the first is a result of a 1-bit shifted into one end of the shift register, and the second is a result of a 0-bit. Should the state appear to change in any other way, it is obvious that incorrect information has been received by the decoder. This algorithm then uses previous and subsequent information to the received error bit(s) to correct the error.

To illustrate, Figure C-1 is a 3-bit or constraint length of 3-shift register; and for state diagram purposes, the "state" of the register is the first two bits, 1 and 2, of the register. Bit 3 has no effect on the contents of the register because each time a new bit is shifted into position 1 of the register, bit 3 is shifted out. The Mealy model state diagram of Figure C-2 shows the legal moves from state-to-state (the double digits represent the state; the single digits, the bit shifted in; the arrow, the direction of legal movement). Figure C-3 shows the type of encoder used for constraint length 3, rate 1/2. As pairs of bits are received from the encoder (a pair for each state or each input bit), they represent moves from state-to-state on the diagram. When an illegal move is made on the diagram, a score is kept as to how unlikely the legality of the move was to reach each of the four states. After a few moves, some of the paths around the diagram would result in high scores for some paths since many illegal moves would be required. To eliminate the unlikely paths, a decision is made as to the most likely entry to each state, and its corresponding path record is held; the other is rejected.

To implement the Viterbi algorithm, it is necessary to define a method of scoring each path, a method of retaining the path record for each state, and a comparison for rejecting paths that do not score well.

This can be more readily accomplished by rearranging the state diagram to a more usable form. Figure C-4 shows a redrawn state diagram with the states vertical and all moves going to the right from A to B. By continuing C, D, etc, as many consecutive moves as needed can be drawn, but the pattern is the same as for the first move. The encoded output that would result with each move can be derived from Figure C-3 and can be listed as in Table C-1. Then the encoded output can be transferred to Figure C-4 as the double digits along the path lines. The single digits show what bit would have been shifted into the state A to produce the move to B. Scoring is accomplished by counting the bit difference

between the double digits in Figure C-4 and the two digits of the received message for each step. The two scores for each state are then added to the scores from the previous step, then compared and the path with the smaller score is taken as the most likely path. The previous path record is taken, shifted on position, and the corresponding 1 or 0 bit (single digits in Figure C-4) inserted in the vacated end position.

Figure C-5 shows a completed diagram of 11 moves proceeding from the left to the right with only the chosen path to each state drawn in, and the resulting score at that state and at that move. The accompanying table shows the data message, the encoded message, and the received message. Move No. 3 simulated an error in the received message. However, when the data get far enough downstream, the message has been corrected by elimination of the incorrect path.

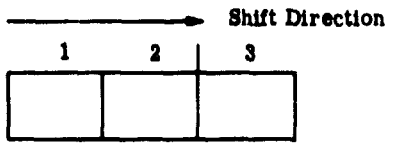


Figure C-1 Constraint Length, Three-Shift Register

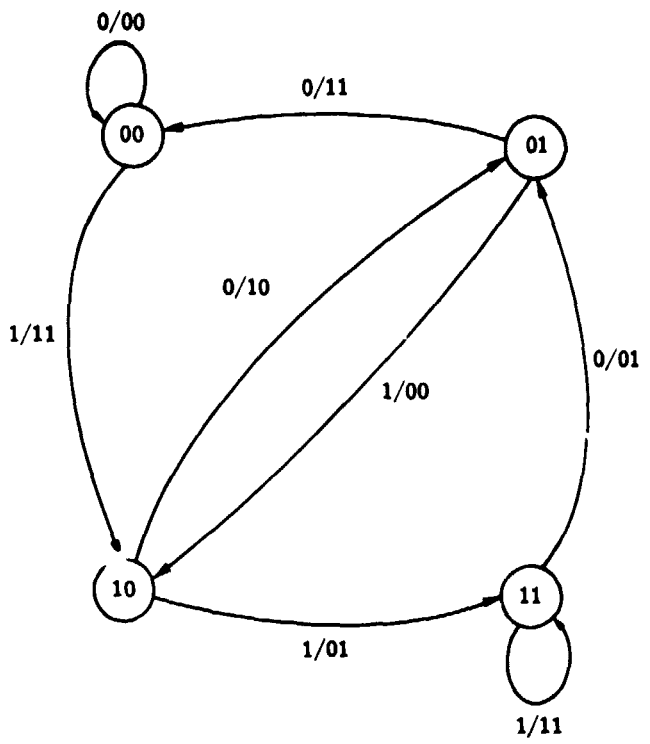


Figure C-2 State Diagram

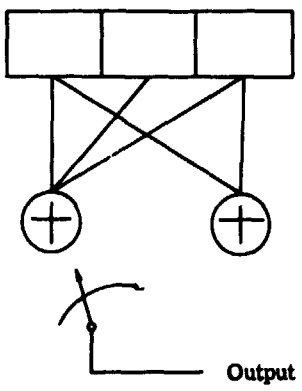


Figure C-3 3-Bit Encoder

Table C-1 Encoded Output

Register Contents	Encoded Output
State	
00 0	00
00 1	11
01 0	10
01 1	01
10 0	11
10 1	00
11 0	01
11 1	10

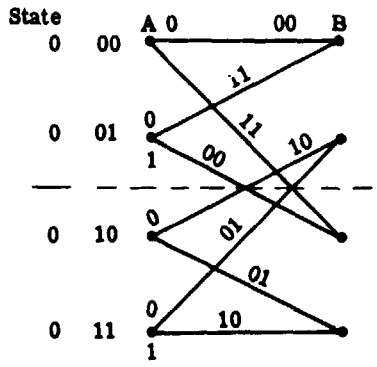


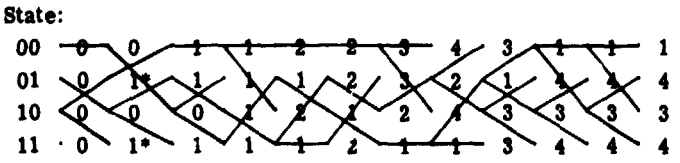
Figure C-4 Redrawn State Diagram

Move	Input to Encoder	Shift Register Contents	Encoder Output (Transmitted)	Received Bits
1	0	000	00	00
2	1	100	11	11
3	1	110	01	00
4	0	011	01	01
5	1	101	00	00
6	1	110	01	01
7	1	111	10	10
8	0	011	01	01
9	0	001	11	11
10	0	000	00	00
11	0	000	00	00

Received Bits ----- 00 11 00 01

Move: Initial Condition 1 2 3 4 5 6 7 8 9 10 11

State:



Resulting Output Bit: 0 1 1 0 1 1 1 0 0 † †

*Note that if both paths score the same, the upper was arbitrarily chosen.

†These output bits cannot be identified until more moves; the output lags the input by a constraint length or more

Figure C-5 Decoded Message