Eindhoven University of Technology

MASTER

Implementation of a soft-decision Viterbi codec

Nouwens, W.J.W.M.

Award date:
1985

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

4/919
TECHNISCHE HOGESCHOOL
EINDHOVEN
STUDIEBIBLIOTHEEK
ELEKTROTECHNIEK

AFDELING DER ELEKTROTECHNIEK
TECHNISCHE HOGESCHOOL EINDHOVEN
VAKGROEP TELECOMMUNICATIE EC

IMPLEMENTATION OF A SOFT-DECISION VITERBI CODEC

door W.J.W.M. NOUWENS

Verslag van het afstudeerwerk
uitgevoerd van februari tot december 1984
Afstudeerhoogleraar: prof.dr. J.C. Arnbak
Begeleiders: ir. A.P. Verlijsdonk/ S.H. Mneney M.A.Sc.

De afdeling der elektrotechniek van de Technische Hogeschool Eindhoven aanvaardt geen verantwoordelijkheid voor de inhoud van stage- en afstudeerverslagen.

CONTENTS

SUMMARY

This report describes the implementation of a soft-decision Viterbi decoder. The effects of the threshold spacing and the number of quatization levels (Q) on the bit erreor probability are investigated. An upper bound on the bit error probability is derived and computed as function of the threshold spacing and E_s/N_o for Q=4 and Q=8.For the threshold spacing, two cases are considered: The threshold spacing normalized to the noise-power density (α) and the threshold spacing normalized to the signal energy per code symbol (β). The degradation of the bit error probability performance is not significant even if the the threshold spacing differ considerably from the optimum values. Therefore a soft decision Viterbi decoder with 8 quatization levels is chosen. In view of the desired high decoding rate, a parallel realization of the decoder is chosen. The theoretical minimum decoding data rate is at least 4 Mbits/s. The bit error rate performance of the decoder is measured with a polar baseband channel. The measured and computed results agree well. The bit error rate of the channel without coding is also measured. The code gain of the decoder is 3.4 dB for Q=8 and 3.1 dB for Q=4.

LIST OF NOTAIONS

ACS	Add-compare-select
AGC	Automatic gain control
AWGN	Additive white Gaussian noise
	The threshold normalized to the noise power density
BER	Bit error rate
BM	Branch metric
	The threshold normalized to the signal energy per code symbol
D_f	Free or minimum distance of a code
E_s	Signal energy per information bit
E_b	Signal energy per code bit
FEC	Forward error correction
G/T	Figure of merit
K	Constraint lenght of a code
LSTTL	Low power Schottky TTL
N	Noise power
N_0	Noise power density
P	Bit error probability
Q	Number of quantization levels
QPSK	Quadriphase phase shift keying
R	Code rate
S	Signal power per code symbol
SCPC	Single carrier per channel
SM	State metric
T	Threshold spacing

1. INTRODUCTION

The Telecommunication Division of the Eindhoven University of Technology is carrying out a research project (EC-18) . entittled "Satellite communication for rural zones", in which a information distribution system by satellite has been proposed [1]. Many African countries have a infrastructure which is only developed in the urban areas. While over 75% of the population lives in the rural areas where the main occupation is farming. A information distribution network is needed for education, weather forecast, news and information on market prices. Since most African countries have a vast size and low density of population, the cost of a terrestial communication network would be high. Therefore the proposed information distribution system uses a existing satellite link. The system offers four services on a time-shared basis, namely

a) a still picture plus speech service
b) a teletext plus speech service
c) a teletext service only
d) a scribophone service

The system configuration is illustrated in fig. 1.

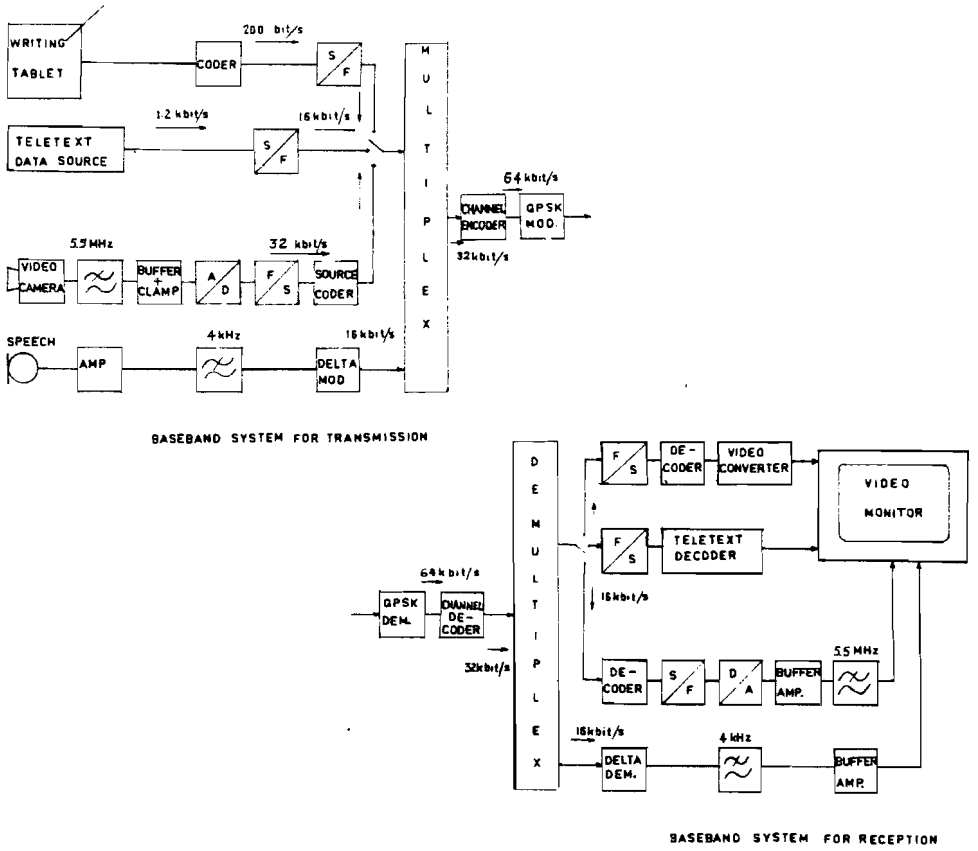


fig. 1 Block diagram of information distribution system

The proposed digital satellite link is composed of a standard-B INTELSAT earth station [1], a dedicated SCPC channel and a low- cost non-standard receiving station. A SCPC channel is a narrowband channel with its own carrier. The SCPC channel has a bandwidth of 38 kHz, the 64 kbit/s data stream being modulated on the carrier with a QPSK-modulator. The non-standard receiving station has a figure of merit G/T=16.4 dB/K, compared to reception by a standard-B earth station which has a figure of merit G/T=37.7 dB/K. A INTELSAT link with bit error rate (BER) of 10^{-5} is based on transmission and reception by standard-B earth stations. Therefore the EIRP of the QPSK carrier radiated from the satellite should be increased by about 15 dB, to reach the non-standard earth station. A forward error correting (FEC) coding scheme is used to reduce the extra EIRP required by about 3 dB. The remaining 12 dB should be gained by increasing the carrier power radiated from the satellite.
This report describes an implementation of the forward error correcting scheme. Convolutional coding and soft-decision Viterbi decoding have been chosen, because convolutional coding can decode continously without the complex synchronisation mechanism required when using block coding. The adopted code is a non-systematic convolutioal code [2] with constraint length K=3. This code and soft-decision Viterbi decoding gives a reasonable code gain (in the order of 3 dB) with a modest decoder complexity. Furthermore, the effects ot the threshold spacing and the number of quantization lecels on soft-decision decoding are investigated.

2. REVIEW OF CONVOLUTIONAL CODING AND VITERBI DECODING

2.1. Convolutional coding

Convolutional codes are a subclass of the linear codes and they outperform block codes of the same complexity. This class of codes is called "convolutional" codes because the output sequence $\underline{v}$ can be expressed as the convolution of the input data sequence $\underline{u}$ and a generator sequence $\underline{g}$ [3]. The convolutional encoder is a linear finite state machine, composed of a K-stage shift register and n modulo-2 adders, each forming one code symbol. The data will be shifted into the register b bits at a time ($b<n$). The length of the register is called the constraint length K of the code. The rate R of the code is $R=b/n$. The encoder used in this project has constraint length $K=3$ and rate $R=1/2$ (fig. 2).

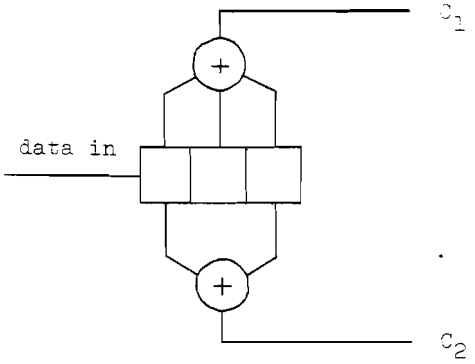


fig. 2 a convolutional encoder

The code word x depends on the new data bit and the previous 2 data bits. The permutations of the last bits of the shift register are the state of the encoder, so there are 2^{K-1} states. The transitions from one state to another are called branches. All the branches together form the possible decoder paths and these are represented in a trellis diagram (fig. 3). The branches representing a input "0" are shown as solid lines and the branches representing a input "1" are shown as dashed lines. Each branch in this diagram is accompanied with its code word (c_1 c_2).

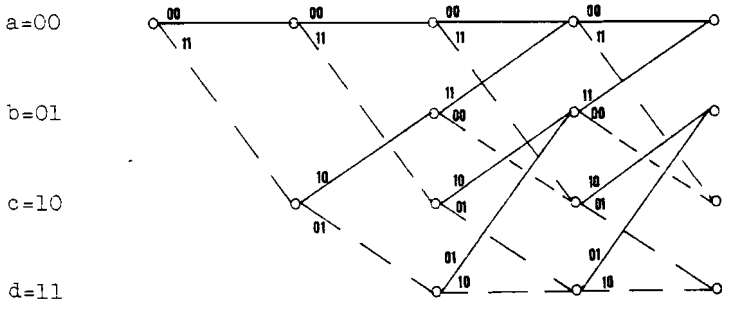


fig. 3 Trellis diagram (K=3,R=1/2)

2.2. Viterbi algorithm

The Viterbi algorithm is a maximum-likelihood decoding algorithm for convolutional codes. A straightforward maximum-likelihood decoding scheme compares the received sequence $\underline{y}$ with all possible sent code sequences $\underline{x}$ and chooses the most likely sequence. The decision criterion is the log likelihood function or metric

$$\ln \prod_{i=1}^{B} p(y_i \mid x_i^m) = \sum_{i=1}^{B} \ln p(y_i \mid x_i^m) \qquad (1)$$

with B : the message length.
y_i : a received symbol.
x_i^m : the code symbol of the m^{th} message sequence.

A message sequence with length B has 2^B possible code sequences. The message length B can easily be several hundred bits, so the complexity of the decoder becomes enormous. The Viterbi algorithm uses the properties of the convolutional codes and therefore the complexity of the decoder depends on the constraint length K, instead of on the message length B.

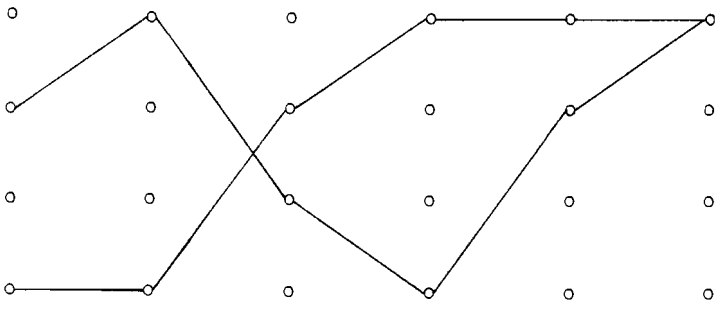


fig.4 Merging paths in trellis diagram

When two paths merge in the ith node (fig. 4), the most likely path is chosen and the other is discarded. The Viterbi algorithm makes, for every received code symbol

and for each of the states, a decision which paths will survive. A memory of length B is needed to store the survivors. The state metric SM_i is the log likelihood function of path ending at state S_i (i=a,b,c,d) and the branch metric BM_{nm} is the log likelihood function of a code branch v_j ,where n,m are the corresponding code bits of branch v_j .The new state metrics are computed as the sum of the previous state metrics and the branch metrics (table 1).

$$
\begin{aligned}
SM_{00} &= \max[\ SM_{00} + BM_{00}\ ,\ SM_{01} + BM_{11}\ ] \\
SM_{01} &= \max[\ SM_{10} + BM_{10}\ ,\ SM_{11} + BM_{01}\ ] \\
SM_{10} &= \max[\ SM_{00} + BM_{11}\ ,\ SM_{01} + BM_{00}\ ] \\
SM &= \max[\ SM_{10} + BM_{01}\ ,\ SM_{11} + BM_{10}\ ]
\end{aligned}
$$

table 1 State metric computation

The path memory length d can be reduced because the survivor path will merge with high probability after a certain length of about 4 or 5 constraint lengths [4]. By resticting the memory path length d to be 5 times the constraint length (d=5K), there will be a neglible degradation.

2.3. Distance properties of convolutional codes

The distance properties of convolutional codes are important for the analysis of the error performance of the codes. The error probability for linear codes can be bounded in terms of the weight of all the code vectors. The weight of the linear codes corresponds to the set of Hamming distances from any one code vector to all others. The Hamming distance is the number of different code bits between two code vectors. Linear codes have another interesting property. The set of Hamming distances from a given code vector to all other code vectors is the same for all code vectors. Thus we can consider one code vector and have the results for all the code vectors [5].

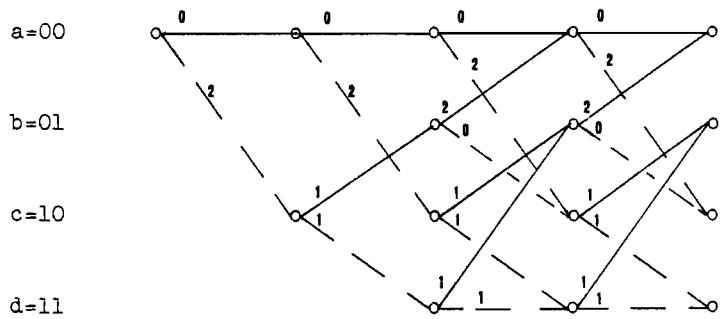


fig. 5 Trellis diagram labelled with distances from all-zero path

We assume an all-zero code vector to be sent and look at all the paths that merge with the all-zero path for the first time at an arbitrary node j. There is one path with distance 5 which diverges 3 branches back and there are two paths with distance 6 which diverge 4 and 5 branches back.It is possible in this manner to calculate the distances of all diverging paths. The generating function can be derived from the state diagram (fig. 6). We label the branches with D^0, D and D^2, where the exponent corresponds with the distance of the branch and D is a formal parameter. The state diagram is split open at node a=00 because this is the close loop which corresponds to the all-zero path.

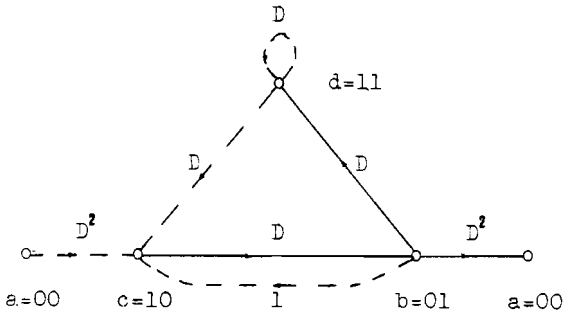


fig. 6 State diagram labelled with distances from all-zero path

The generating function T(D) can be computed by summing all possible path through the state diagram [6].

$$T(D) = \frac{D^5}{1 - 2D}$$

$$= D^5 + 2D^6 + 4D^7 + \ldots + 2^K D^{K+5} + \ldots \qquad (2)$$

This expression indicates that there are 2^K paths at a distance K+5 from the all-zero path. We can generalize the generating function by adding the length of the diverging path with formal parameter L and the number of input "1" in the diverging path with formal parameter N (fig. 7).

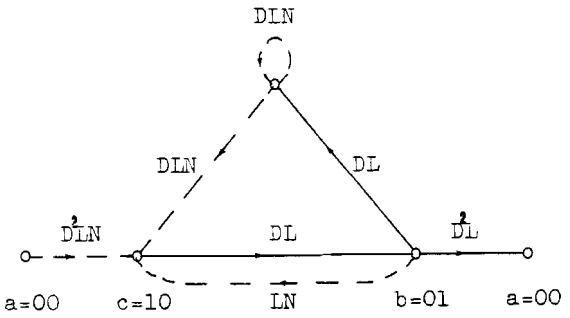


fig. 7 State diagram labelled with D, L and N

The generating function T(D,L,N) can be computed according to eq. 2. The exponents of the formal parameters D, L and N are the distance from the all-zero path, the length of the diverging path and the number of input "1" in the diverging path.

$$T(D,L,N) = \frac{D^5 L^3 N}{1 - DL(1+L)N}$$

$$= D^5 L^3 N + D^6 L^4 (1+L) N^2 + \ldots$$

$$+ D^{K+5} L^{K+3} (1+L)^K N^{K+1} + \ldots \qquad (3)$$

3. SOFT-DECISION VITERBI DECODING

The satellite communication channel considered here is composed of a QPSK-modulator, a satellite RF-link and a QPSK-demodulator.The additive white Gaussian noise (AWGN) channel is a good model for this satellite channel. Hard decision quantization using one threshold is the optimum for non-coded data sent over an AWGN channel [7]. However, hard decision causes a loss of about 2 dB for coded data sent over an AWGN channel. Soft-decision quantization using more than one threshold, can be applied to reduce the loss.

3.1. Communication channel model

Many real communication channels can be modelled into an AWGN channel and therefore the AWGN channel is the basis for the soft-decision channel model. The received baseband signal before quantization can be represented by:

$$r = +\sqrt{S} + n \qquad (4)$$

where S is the received signal power for acode symbol and n is the Gaussian noise voltage with variance N /2 (N: the noise power and N_o the one-sided noise power density). The demodulated signal is quantized into a Q-level digital word y_i (i=0,1,..,Q-1), using a A/D converter with equally spaced thresholds. This digital word is the input of the Viterbi decoder. The transmitted signals are binary symbols. So the over-all transmission channel can be represented as a binary input Q-ary output symmetric memoryless channel (fig. 7).

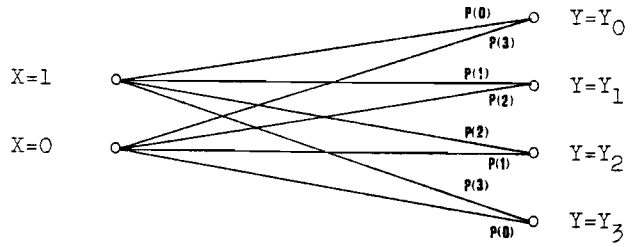


fig. 7 Binary input Q-ary output symmetric memoryless channel (Q=4).

The transition probability P(i), i.e. the probability that output symbol y is received when input symbol x=1 is transmitted, can be calculated by using. 5 if S/N and the soft-decision thresholds {b } are given.

$$P(i) = Pr(y=y_i \mid x=1)$$

$$= Pr(y=y_{Q-1-i} \mid x=0)$$

$$= \frac{1}{\sqrt{N\pi}} \int_{b_i}^{b_{i+1}} \exp\{ -(r-\sqrt{S})/N \}\, dr \qquad (5)$$

with $i=0,1,\ldots,Q-1,Q$

$b = -\infty$

$b = \infty$

$b = (i-Q/2)T$

T :the threshold spacing

3.2. The metric for an AWGN channel

The metric or log likelihood function is the decision criterion in the Viterbi decoding process. Therefore we will derive in this section the metric for an AWGN channel [6]. We use the notation that x_{jk} is the kth code symbol of the jth branch and r_{jk} is the corresponding received signal. Each binary symbol x_{jk} , which we take here for convience to be +1, is sent with signal power S . Thus the conditional probability density (or likelihood) function of r_{jk} given x_{jk} is

$$p(r_{jk} \mid x_{jk}) = \frac{\exp\{ -(r_{jk} - \sqrt{S}\, x_{jk})/N \}}{\sqrt{\pi N}} \qquad (6)$$

The likelihood function for the jth branch of a particular code path x^m is

$$p(r_j \mid x_j) = \prod_{k=1}^{n} p(r_{jk} \mid x_{jk}) \qquad (7)$$

since each symbol is assumed to be affected independently by the band-limited white Gaussian noise. Where n is the total number of code symbols per branch. Then the metric or log likelihood function for the jth branch is

$$\ln p(r_j \mid x_j^m) = \sum_{k=1}^{n} \ln[p(r_{jk} \mid x_{jk}^m)$$

$$= -\frac{1}{N} \sum_{k=1}^{n} [r_{jk} - \sqrt{S}\, x_{jk}]^2 - \frac{1}{2} \ln \pi N$$

$$= \frac{2\sqrt{S}}{N} \sum_{k=1}^{n} r_{jk} x_{jk}^{m} - \frac{S}{N} \sum_{k=1}^{n} [x_{jk}^{m}]^2 -$$

$$\frac{1}{N} \sum_{k=1}^{n} r_{jk}^2 - \frac{1}{2} \ln \pi N$$

$$= C \sum_{k=1}^{n} r_{jk} x_{jk} - D \qquad (8)$$

where C and D are independent of m. The branch metric is the sum of the symbol metrics and the symbol metric is the product of the received signal r_{jk} and the transmitted symbol x_{jk}. Similarly, the metric for any path is the sum of the metrics of each of its branches.

3.3. The upper bound of the bit-error probability

The bit error probability P_B, the expected number of errors normalized to the total number of errors in a received sequence, is used to analyze the performance of the Viterbi decoder [6]. The all-zero sequence considered to be sent. The bit error probability can be bounded by:

$$P_B < \sum_{k=d}^{\infty} c_k P_k$$

where k : the distance from the correct path (see fig. 5)
d_f : the minimum or free distance of the code
c_k : the number of errors caused by paths with distance k from the correct path
P_k : the probability that such a path is chosen

3.3.1. The evaluation of P_k

A Viterbi decoder chooses the path with the highest metric. Therefore an incorrect path is chosen if this path has a higher metric than the correct path. The metric of a path sent over an AWGN channel is

$$\sum_{j=1}^{m} \sum_{k=1}^{n} x_{jk} r_{jk} \qquad (10)$$

where x_{jk} is the kth code symbol of the jth branch (x=+1), r_{jk} is the corresponding received symbol and m is the length of the path. Consider the case of an all-zero sequence being sent, the correct path consists of x =-1 symbols and an incorrect path consists of code symbols x=+1. An incorect path is chosen if:

$$\sum_{j=1}^{m}\sum_{k=1}^{n} x'_{jk} r_{jk} \geq \sum_{j=1}^{m}\sum_{k=1}^{n} x_{jk} r_{jk}$$

$$\sum_{j=1}^{m}\sum_{k=1}^{n} (x'_{jk} - x_{jk}) r_{jk} \geq 0$$

$$\sum_{j=1}^{m}\sum_{k=1}^{n} 2\, r_{jk} \geq 0$$

$$\sum_{j=1}^{m}\sum_{k=1}^{n} r_{jk} \geq 0 \qquad (11)$$

The probability P_k is the probability that an incorrect path is chosen which has a distance k from the correct path.

$$P = Pr\left(\sum_{m=1}^{k} r_m \geq 0 \right) \qquad (12)$$

However the received signal r is an analog signal, but will be quantized with an A/D converter using equally spaced thresholds (fig.8).

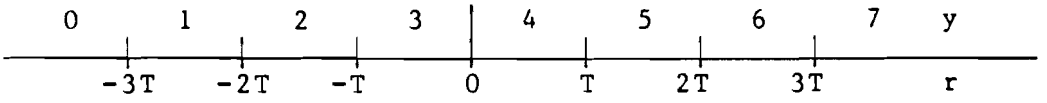


fig. 8 Quantization scheme (Q=8).

This changes the calculation of the probability P_K. The zero-level of the signal r correspond to y=(Q-1)/2. So the probability P_K for a soft decision decoder is:

$$P_K = Pr\left(\sum_{m=1}^{k} y \geq \frac{(Q-1)k}{2} \right)$$

$$= \sum_{n=0}^{(Q-1)k} Pr\left(\sum_{m=1}^{k} y = \frac{(Q-1)k+n}{2} \right)$$

$$= \sum_{n=1}^{(Q-1)k} q(n) + \frac{1}{2} q(0) \qquad (13)$$

with: $$q(n) = Pr\left(\sum_{m=1}^{k} y = \frac{(Q-1)k + n}{2} \right) \qquad (14)$$

The probability of the inequality (see eq. 13) is the sum of the probabilities of all the events which satisfy the inequality and therefore the index n is introduced. If the metrics of the correct and incorrect path are the same (this is the case for n=0), a decision has to be made about which path is chosen. An arbitrary choice is made and there is the probability 1/2 that the incorrect path is chosen. Therefore the term q(0) has a coefficient 1/2. The terms q(n) can be calculated by eq. 5 where the integers L_i are the number of symbols $y_m = i$ ($i=0,1,\ldots Q-1$) and P(i) is the probability that $y_m = i$ is received when x=1 was transmitted. The set $\{L_i\}$, i.e. the permutations of L_i which satisfy the equations 16 and 17, corresponds with all the possible paths of distance k from the correct path.

$$q(n) = \sum_{\{L_i\}} k! \prod_{i=0}^{Q-1} \frac{P(Q-1-i)^{L_i}}{L_i!} \qquad (15)$$

where $\{L_i\}$ satisfies:

$$\sum_{i=0}^{Q-1} L_i = k \qquad (16)$$

$$\sum_{i=0}^{Q-1} i*L_i = \frac{(Q-1)k + n}{2} \qquad (17)$$

3.3.2. The evaluation of c_K

The number of errors in the output symbols of a path with distance k is bounded by c_K. The generating function T(D,L,N) contains all the information over the used convolutional code. The number of branches of the path with distance k is not of interest for derivation of c_K. Therefore we use the generating function T(D,N).

$$T(D,N) = T(D,L,N)\Big|_{L=1}$$

$$= \frac{D^5 N}{1 - 2DN}$$

$$= \sum_{d=5}^{\infty} 2^{d-5} D^d N^{d-4} \qquad (18)$$

The coefficient c_K is the number of path with distance k multiplied by the number of "1" in the received data sequence. This can be expressed in terms of the generating function T(D,N) as follows:

$$\frac{\delta T(D,N)}{\delta N} = \sum_{d=5}^{\infty} (d-4)\, 2^{d-5} D^d\, N^{d-3}$$

$$= \sum_{d=5}^{\infty} c_K\, D^d\, N^{d-3} \qquad (19)$$

3.4. The analysis of the soft-decision channel

In the previous sections we derived a numerical algorithm for the computation of the upper bound of the bit error probability. The bit error probability is bounded by an infinite weighted summation of P_K. It is impossible to compute an infinite summation and therefore we have to truncate at k=M. We choose M=32, this is the number of code symbols corresponding to the memory path length used in the decoder

$$P_B < \sum_{k=5}^{M} c_K\, P_K = \sum_{k=5}^{32} (k-4)\, 2^{k-5} P_K \qquad (20)$$

$$P_K = \sum_{n=0}^{(Q-1)k} q(n) + \frac{1}{2} q(0) \qquad (14)$$

$$q(n) = \sum_{\{L_i\}} k! \prod_{i=0}^{Q-1} \frac{P(Q-1-i)^{L_i}}{L_i!} \qquad (15)$$

where $\{L_i\}$ satisfies:

$$\sum_{i=0}^{Q-1} L_i = k \qquad (16)$$

$$\sum_{i=0}^{Q-1} i * L_i = \frac{(Q-1)k + n}{2} \qquad (17)$$

The bound on the bit error probability is computed (programs in appendix A) to gain insight into the influences of fading on the channel and the threshold spacing on the soft decision Viterbi decoding process [8]. Two theoretical cases are analized. The first case (a), the noise power is kept constant and the threshold spacing is normalized to the noise power density N_0. This case correspond to a receiver with a high noise temperature and without an AGC. The second case (b), the signal power is kept constant and the threshold spacin is normalized to the signal energy per code symbol. This case correspond to a receiver with an ideal AGC. Both cases are computed for Q=4 and Q=8.

a) with parameters E_b/N_o and
   α: the threshold spacing normalized to the noise power density
   for Q=4 $T = 2\alpha\sqrt{N_o/2}$
   for Q=8 $T = \alpha\sqrt{N_o/2}$

b) with parameters E_b/N_o and
   β: the threshold spacing normalized to the signal energy per code symbol
   $T = \beta\sqrt{E_s}$

3.4.1. Optimum threshold spacing

The optimum threshold spacing of α and β is determined by the computation of the bit error probability curves as function of α and β. The optimum values of the threshold spacing normalized to the noise power density is $\alpha=0.5$ for both Q=4 (fig. 9) and Q=8 (fig. 12). The optimum value of the threshold spacing normalized to the signal energy per code symbol is $\beta=0.5$ for Q=4 (fig. 10) and $\beta=0.25$ for Q=8 (fig. 13). In all cases, the variations of the threshold spacing from the optimum values give no significant degradation of the bit error probability performance. The theoretical code gain of the decoder is 3.2 dB for Q=4 (fig. 11) and 3.6 dB for Q=8 (fig. 14). Therefore a soft decision Viterbi decoder with 8 quantization levels is chosen.

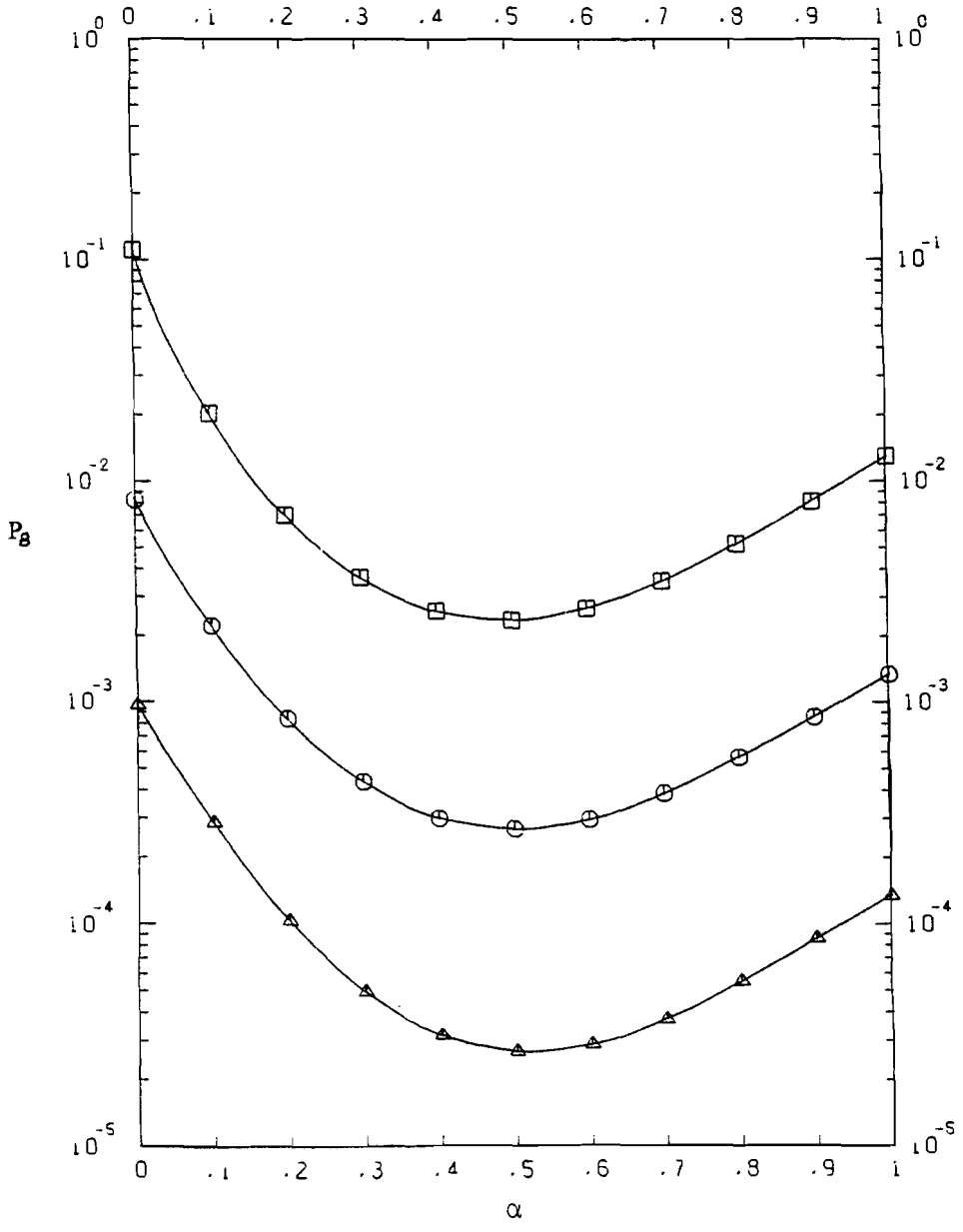


fig. 9 The bit error probability curves as function of the threshold spacing normalized to the noise power density with E_b / N_o as parameter (Q=4).

E_b / N_o = 6 dB △
E_b / N_o = 5 dB ○
E_b / N_o = 4 dB □

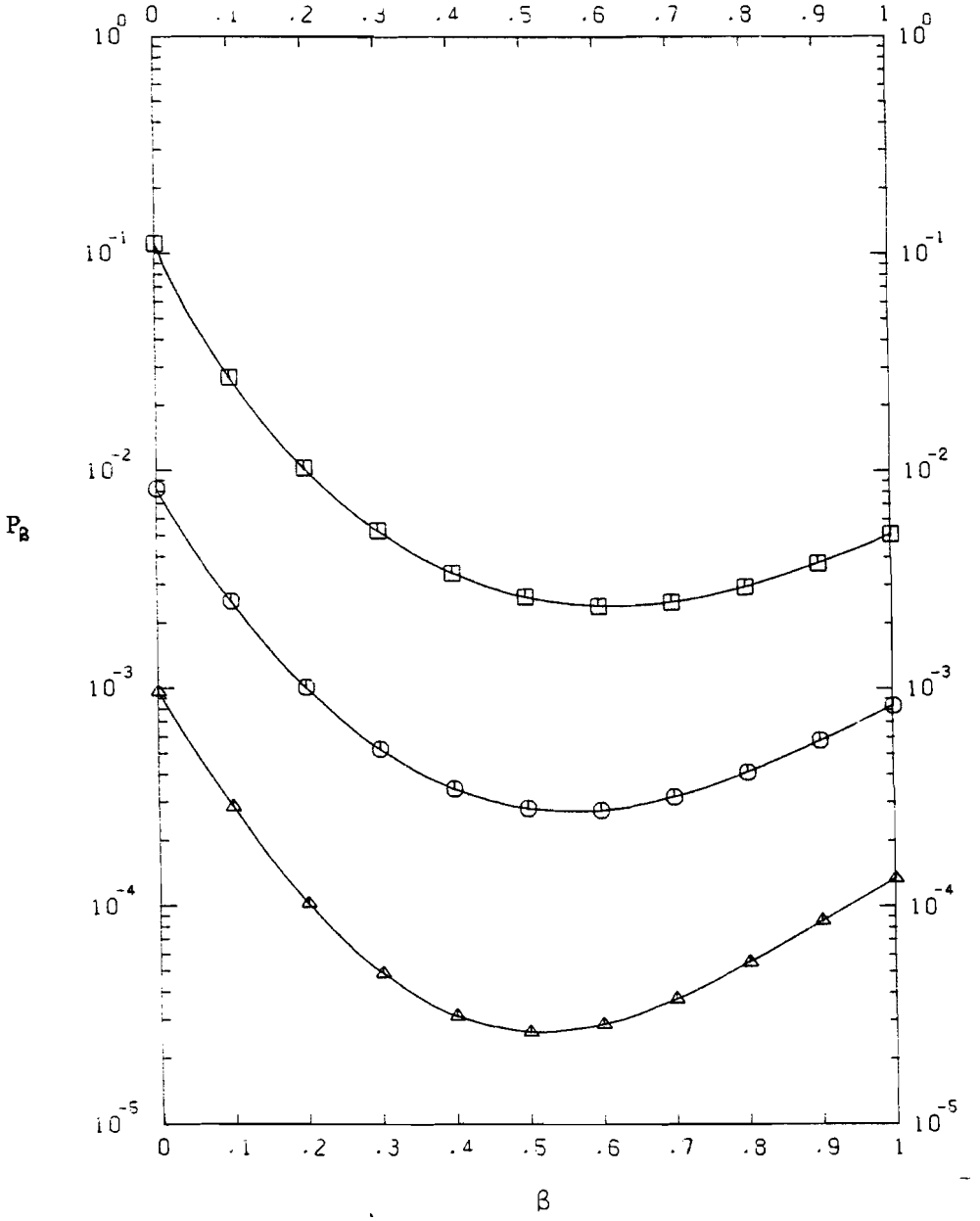


fig.10 The bit error probability curves as function of the threshold spacing normalized to the signal energy per code symbol with E_b / N_o as parametr (Q=4).

E_b / N_o = 6 dB △
E_b / N_o = 5 dB ○
E_b / N_o = 4 dB □

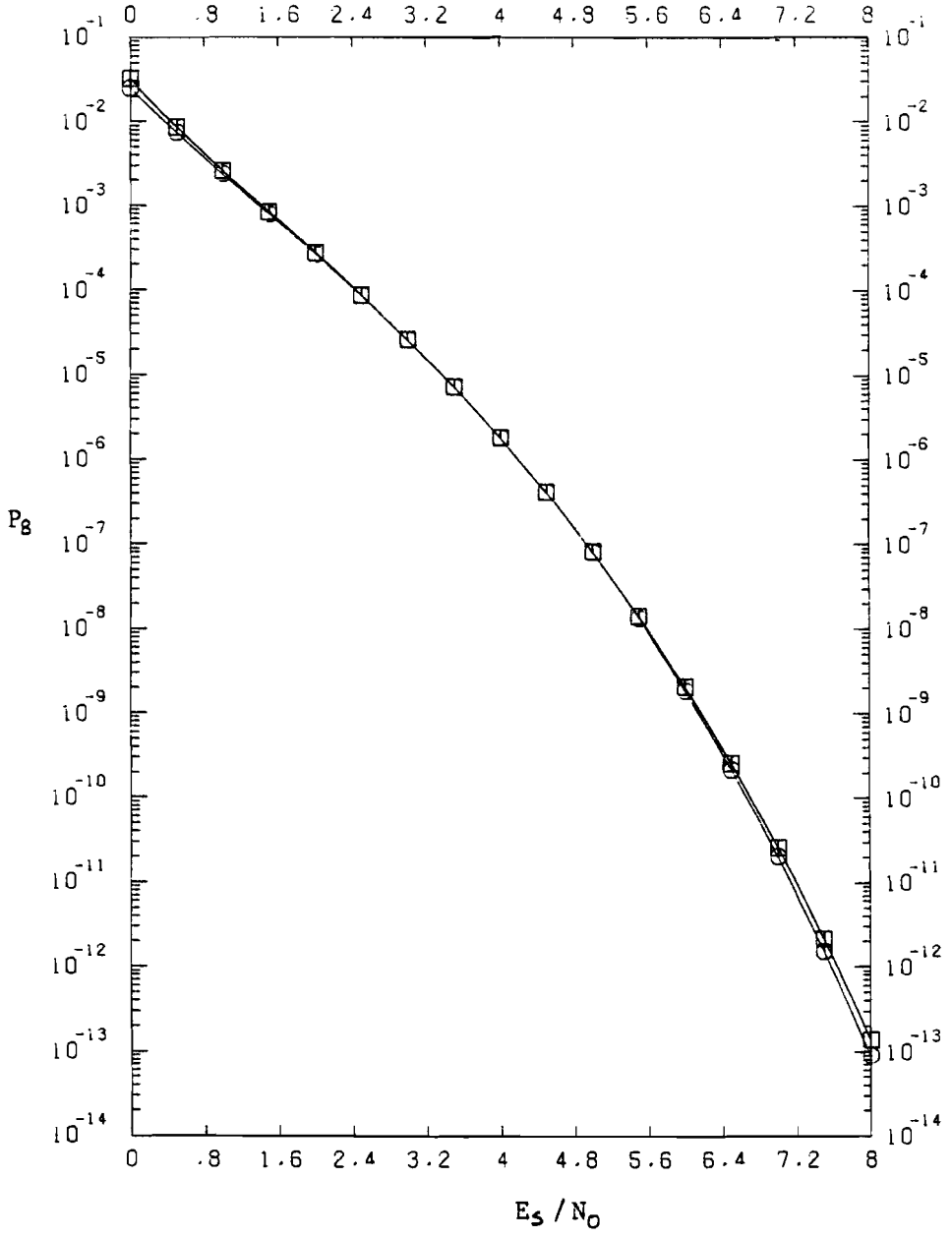


fig.11 The bit error probability curves as function of E_S/N_0 with parameters the threshold spacing normalized to the noise power density (α) and the threshold spacing normalized to the signal energy per code symbol (β).

$\alpha = 0.5$ ○
$\beta = 0.5$ □

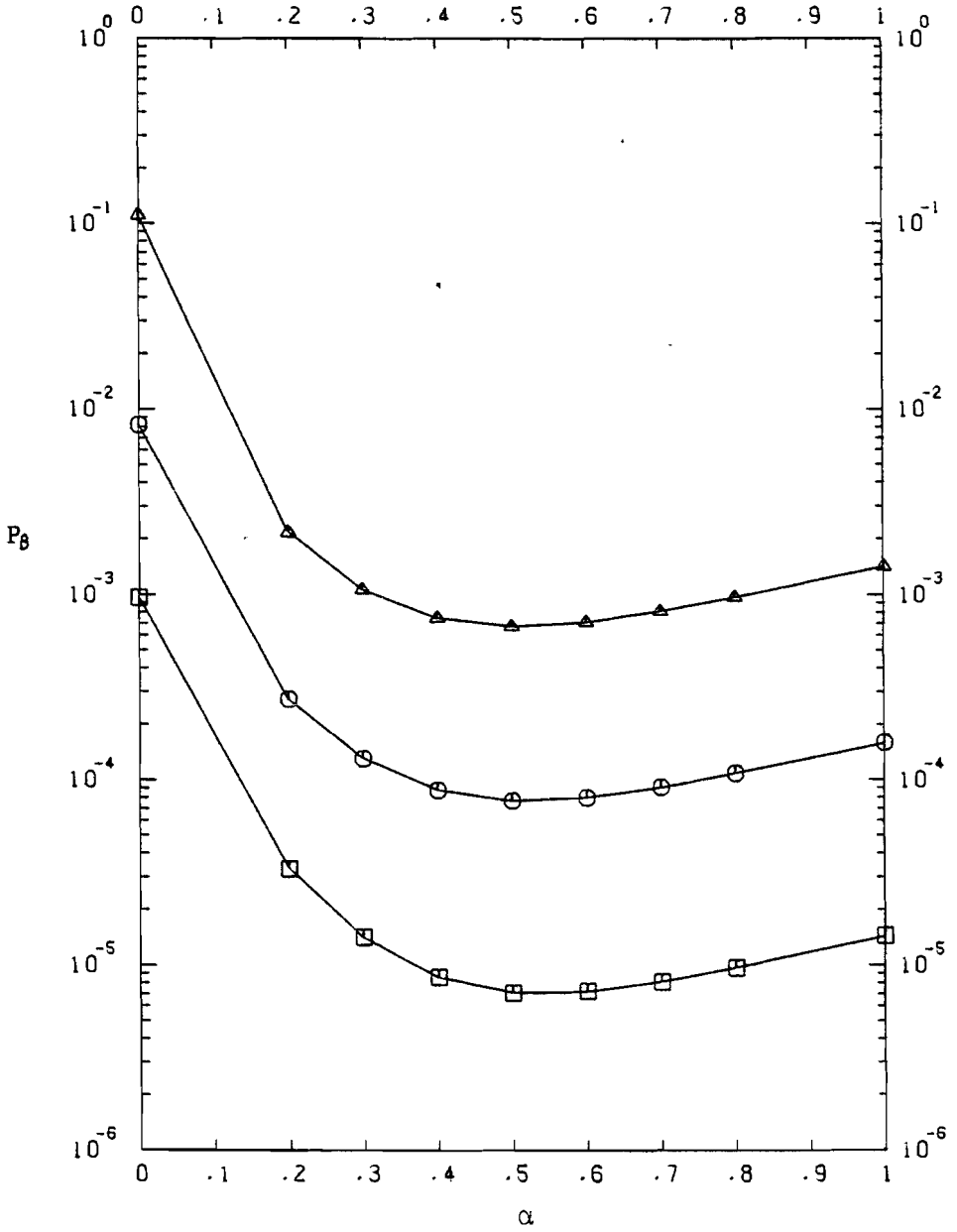


fig.12 The bit error probability curves as function of the threshold spacing normalized to the noise power density (α) with parameter E_b/N_o (Q=8)

E_b/N_o = 6 dB □
E_b/N_o = 5 dB ○
E_b/N_o = 4 dB △

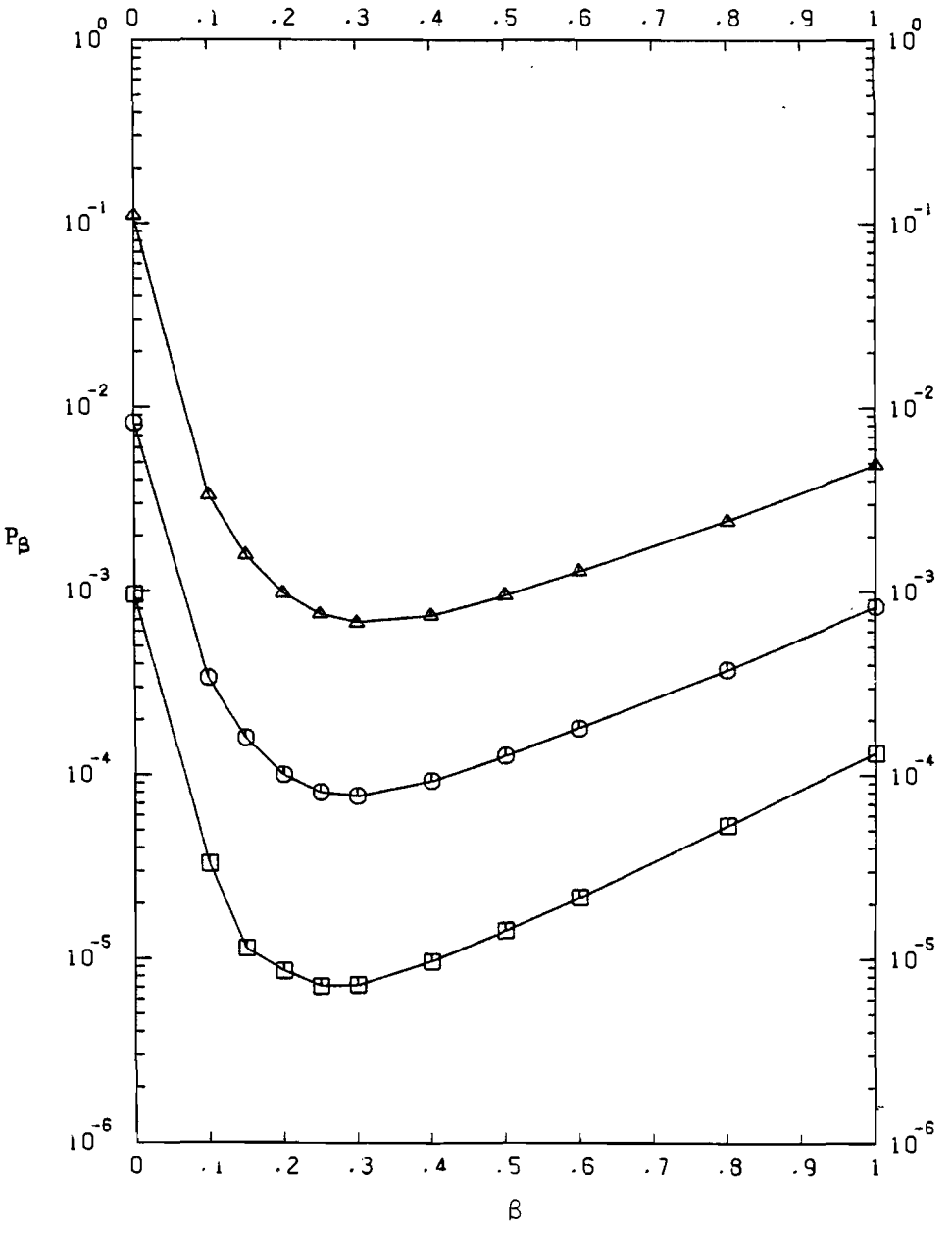


fig. 13 The bit error probability curves as function of the threshold spacing normalized to the signal energy per code symbol with parameter E_b/N_o (Q=8)

E_b/N_o = 6 dB □
E_b/N_o = 5 dB ○
E_b/N_o = 4 dB △

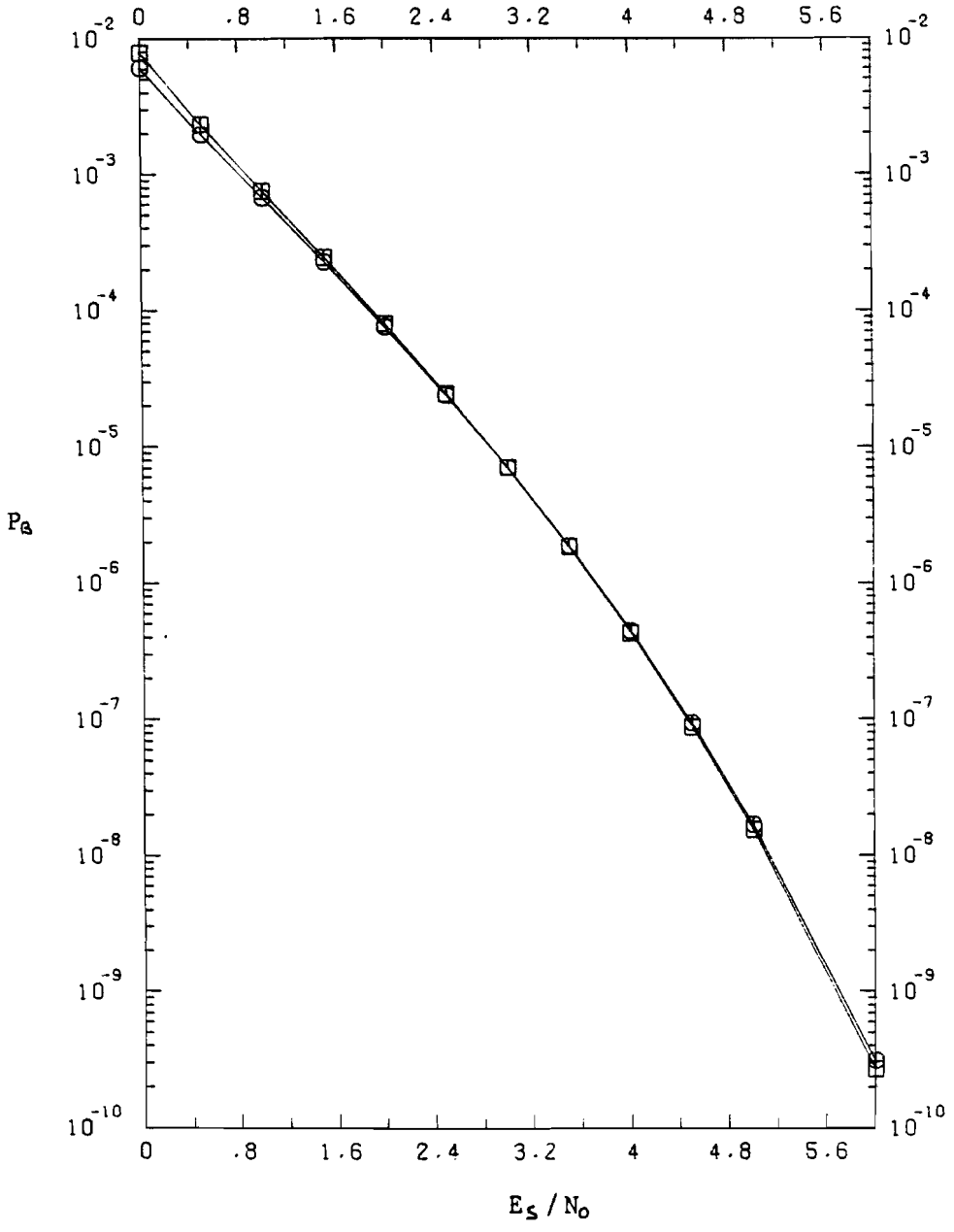


fig.14 The bit error probability curves as function of E_S/N_0 with parameters the threshold spacing normalized to the noise power desity (α) and the threshold spacing normalized to the signal energy per code symbol (β)

$\alpha = 0.5$ O
$\beta = 0.25$ □

4. THE DESIGN OF THE VITERBI CODEC

The Viterbi codec consists of a convolutional encoder and a maximum-likelihood decoder using the Viterbi algorithm. In contrast to the encoder, the decoder is a complex circuit. Because of the desired transmission rate, both the decoder and the encoder are implemented in low-power Schottky TTL (LSTTL). The codec system uses a convolutional code with constraint length K=3 and rate R=1/2 (see fig. 1).

4.1. The encoder

The encoder uses the convolutional code with constraint length K=3 and rate R=1/2 which has a maximum free distance d_f =5 [2]. The free distance determines the error-correcting properties of the code. The encoder consists of a shift register (74LS164) and two exclusive-or gates (74LS86). The data is shifted one bit at a time and two code bits are formed (fig. 15).

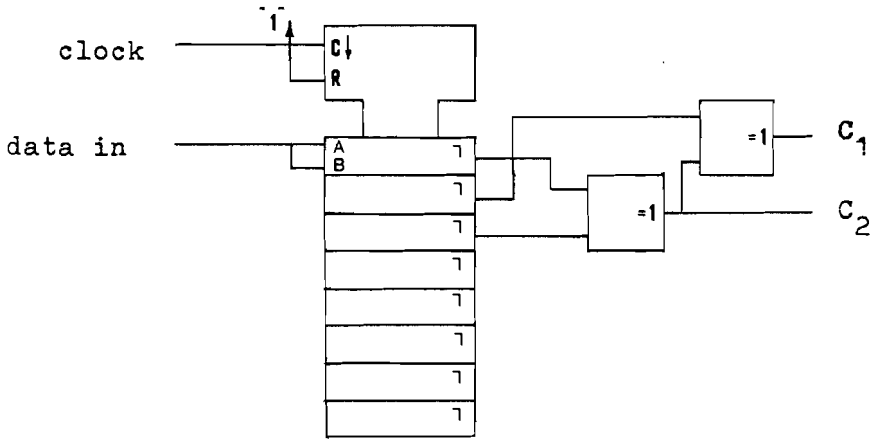


fig.15 Convolutional encoder

4.2. The decoder

The decoder is a soft-decision Viterbi decoder with 8 quantization levels. It is a complex circuit and therefore the decoder is divided into 6 functional blocks (fig. 16).

1) input section
2) branch metric computation section
3) state metric computation section
4) path memory
5) output select section
6) control section

Each of the functional blocks are discussed in the following sections. Because of the desired transmission rate, a parallel realisation of the decoder is chosen. All

branch and state metrics have their own computational sections. Furthermore parallel processing is applied to increase the decoder rate: At clock cycle t, the input signals are converted and stored. The next clock cycle t+1,the branch and state metrics are computed and stored and the data associated with the selected branches are stored in the path memory. At the clock cycle t+2, the most likely survivor is chosen from the state metrics of the clockcycle t+1. A schematic diagram of the decoder is given in appendix B1.

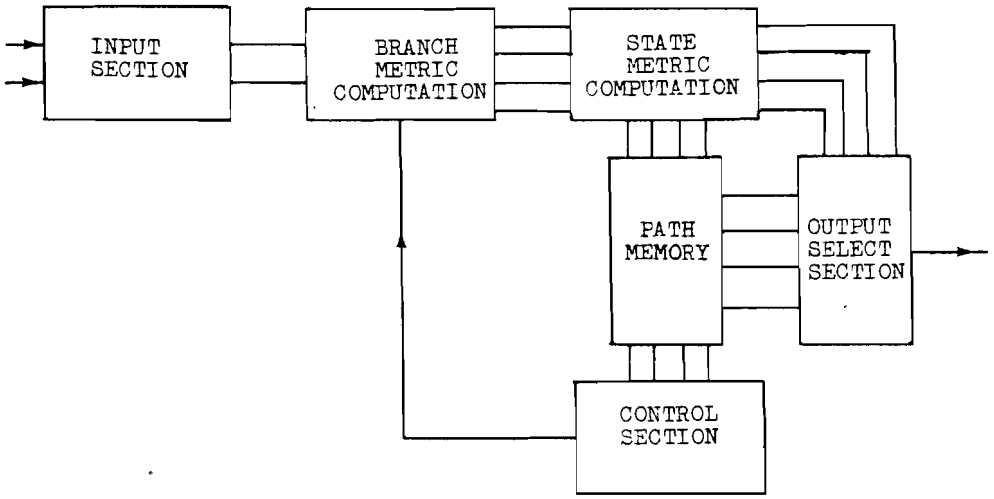


fig. 16 Block diagram of the Viterbi decoder

4.2.1. The input section

The input section is the interface between the analog output signal of the demodulator and the digital Viterbi decoder. The analog signal of the demodulator is processed by an integrate-and dump-filter or by a low pass filter and delivered to the input section. This section (fig. 17) consists of two A/D converters with equally spaced thresholds. The A/D converters consist of a set of comparators. The input signal is compared to 7 thresholds generated by a reference network. The reference network consists of a 150 ohm resistor network. The threshold spacing can be varied between 0.2 and 0.5 volt. The output of the comparators is fed into a code converter which forms a 3 bits digital word. The digital words of both the A/D converters are stored in a latch. A detailed diagram is found in appendix B2.

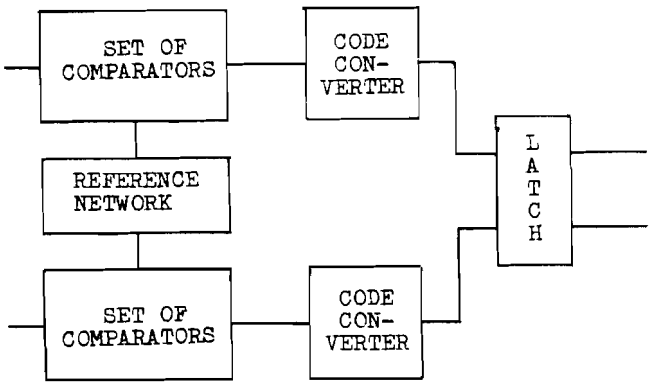


fig. 17 Schematic diagram of the input section

4.2.2. The branch metric computation section

This section computes the log likelihood of the branches. We adopt the convention that the binary word 000 corresponds to a highly reliable received "0" and the binary word 111 corresponds to a highly reliable received "1".The branch metric is the sum of the metrics of the individual code symbols. The metric of a code symbol is the received word when the a code symbol x=1 is sent and the inversion of the word when a code symbol x=0 is sent. The branch metric is a 4 bit word an can vary from 0 to 14 (fig. 18). As a coherent demodulator might be applied, phase ambiguity could occur, since the carrier recovery circuit of the demodulator can only recover the carrier frequency and not the reference phase. For a coherent QPSK demodulator, a phase error of $n*\pi/2$ (n=1, 2, 3)could occur. Thus both output signals could be inverted. Therefore controllable inverters are inserted in the input lines. The control section determines the correct phases and the corresponding control signals. A detailed diagram is given in appendix B3.

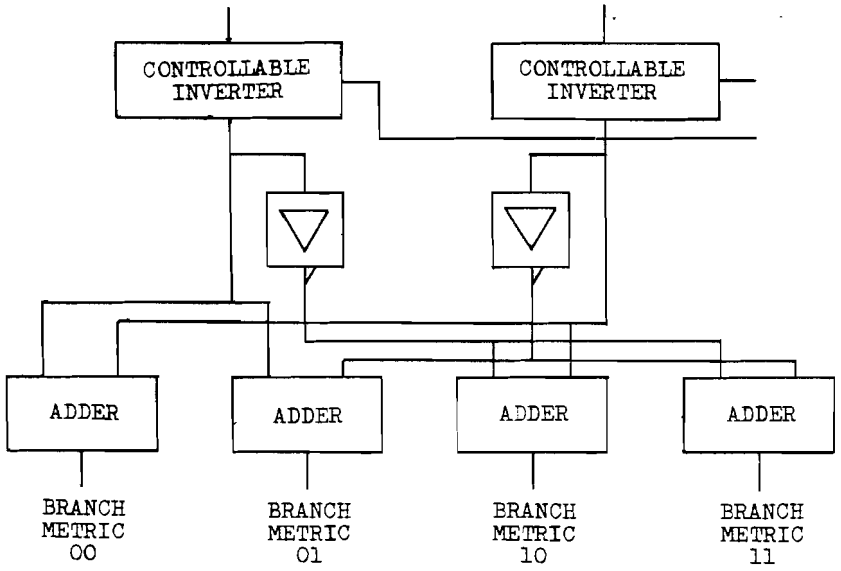


fig. 18 Schematic diagram of the branch metric computaion section

4.2.3. The state metric computation section

The state metric computation section consists of four identical units, one for each state. A unit consists of 2 adders, a comparator and a selector, and therefore this unit is called an add-compare-select (ACS) section (fig. 19) new state metrics is computed with the branch metrics of the branches ending in that particular state and the old state metrics of the states at the beginning of the branches (see table 1). The maximum of the two additions is chosen. Because the state metrics will grow continuously, normalization is needed.Only the difference of the values of the state metrics are important and therefore the state metrics can be normalized by subtracting the same value from all the metrics. It is known that the difference between the maximum and minimum value of the state metrics is bounded by 2(K-1)(Q-1) [9], in this case 28. Thus the state metric words are at least 5 bits wide. Although the ACS sections are 8 bit wide, a state metric words of 6 bits wide is chosen, since a simple normalization circuit can be used. If one of the state metrics is greater than 63, bit 7 of the state metric word is "1", we normalize the metrics by subtracting 32 from the metrics. Before normalization, the values of the state metrics are between 32 and 95 and therefore subtracting 32 is the same as inverting bit 6 of the state metric words. A detailed diagram is found in appendix B4, and the relations between the ACS sections are given in appendix B1.

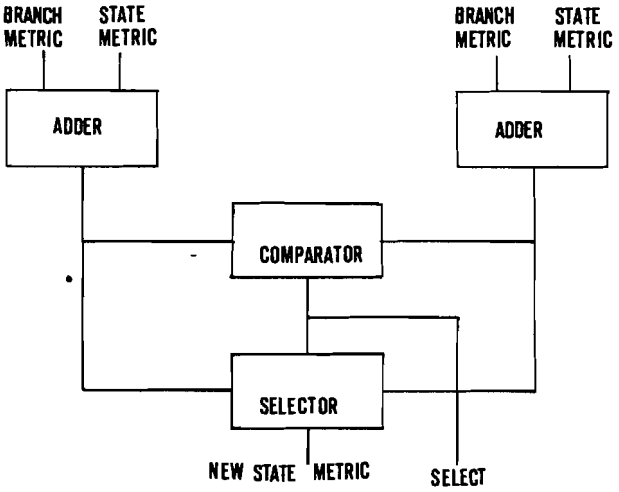


fig. 19 an ACS section

4.2.4. The path memory

The ACS section computes the new state metrics and chooses a new branch for survivor paths. Every branch has a data symbol associated with it. The data symbols associated with the chosen branches are stored in the path memory. The data is shifted in the memory according to the transition of the chosen branch (fig. 20).

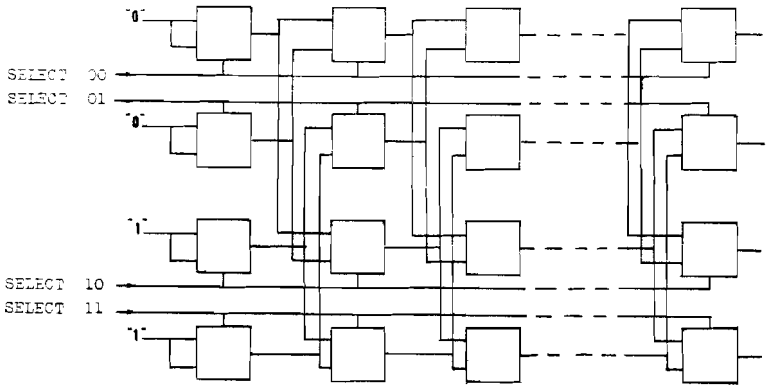


fig. 20 Path memory

The branches ending in the states a=00 and b=01 correspond with the data symbol "0". This data symbol is shifted into the part of the path memory associated with these states. Likewise, a data symbol "1" is shifted into the part of the path memory associated with the states c=10 and d=11 (see fig. 20). Because the output of the first two stages of the memory is independent of the selected branches, those stages can be omitted and the input data has to be adjusted (see table 2)

	fitst stage		second stage		third stage	
state	in	out	in	out	in	out
a=00	0 0	0	0 0	0	0 1	?
b=01	0 0	0	1 1	1	0 1	?
c=10	1 1	1	0 0	0	0 1	?
d=11	1 1	1	1 1	1	0 1	?

table 2

4.2.5. The output select section

The survivor paths are stored in the path memory. The state with the highest metric is the most likely state. The corresponding data sequence, stored in the path memory,is the most likely sequence. The last bit of this sequence is the output of the decoder. The output select section compares the state metrics and determines the state with the highest metric. The comparison is done in two stages. First, the metrics of states a and b and the metrics of states c and d are compared and the highest metrics are selected. Then both selected metrics are compared (see appendix B6). The most likely output of the path memory is selected and stored.

4.2.6. The control section

This section counts the number of non-unanimous decisions in the output of the path memory. If the decoder is in a correct state, most of the decisions will be unanimous [10]. Therefore a phase ambiguity of the demodulator can be detected by the decoder. When the number of non-unanimous decisions exceeds a threshold, an incorrect phase is assumed and the control signals sent to the branch metric computation section are adjusted (see appendix B7). This control mechanism can only resolve the phase ambigiuty of the link for code symbols c_2. The link for code symbols c_1 is transparant for phase inversion. The inversion of the code symbol c_1 is the same as the inversion of the data (see table 3).

data	state	c_1	c_2	data	state	c_1	c_2
0	00	0	0	1	11	1	0
0	01	1	1	1	10	0	1
0	10	1	0	1	01	0	0
0	11	0	1	1	00	1	1

table 3

The phase ambiguity of the link for code symbols c_1 means that the data could be inverted. Differential coding and decoding is used to resolve this problem [11]. However, differential coding increases the bit error rate with a factor 2 (fig. 21).

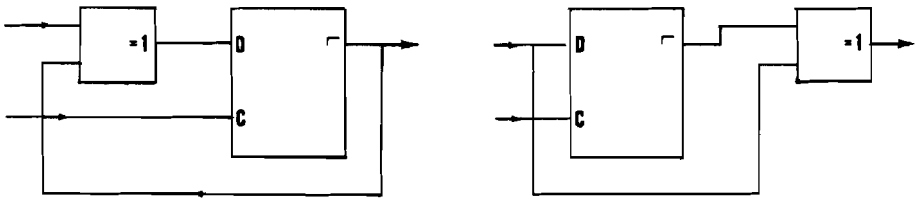


fig. 21 a) differential encoder
b) differential decoder

4.2.7. The decoder timing

Several processes of the decoder work in parallel and therefore the process with the largest propagation delay determines the decoding rate.The following processes occur simultaneously (see table 4). The propagation delay of the sections are calculated with the propagation times of the separate IC given in the TTL data book [12].

proces	propagation delay typ.	max.	
branch metric computation + state metric computation	141	204	ns
normalization + state metric computation	163	239	ns
state metric comparison + output select	157	239	ns
path memory	36	47	ns

table 4 proagation delay

The theoritical minimum decodig rate will be at least 4 Mbit/s. In the design of the decoder, there is assumed that the input signal will be stable at the leading edge of the clock signal. The clock circuit (fig. 22) is used

to compensate the propagation delay of the input section and to increase the fan-out of the clock for the rest of the circuit [13].

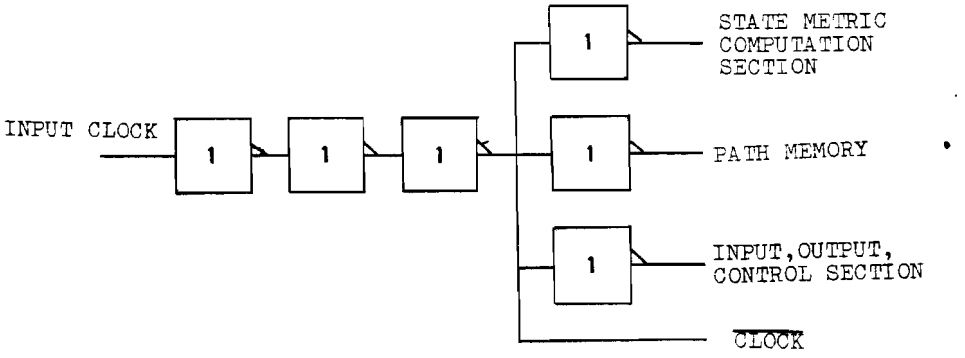


fig. 22 clock circuit

5. MEASUREMENTS

In chapter 3, we derived the bit error probability bound for the adopted code and computed several bit-error probability curves (see fig. 9-14).The performance of the decoder is measured and compared to the analytical results. The performance criterion is the bit error rate (BER). This is the number of errors in a sequence normalized to the total number of bits in the sequence. Since bit error probability and bit error rate are similar for long random sequences, the computed and the measured results can be compared.

5.1. The test setup

The bit error rate of the decoder is measured in a test setup. This test setup consist of a data generator, a channel and a bit-error-rate measurement equipment (fig. 23). The convolutional encoder is inserted between the data generator and the channel. The Viterbi decoder is inserted between the channel and the BER measurement equipment. All the equipment is synchronized to the same clock.

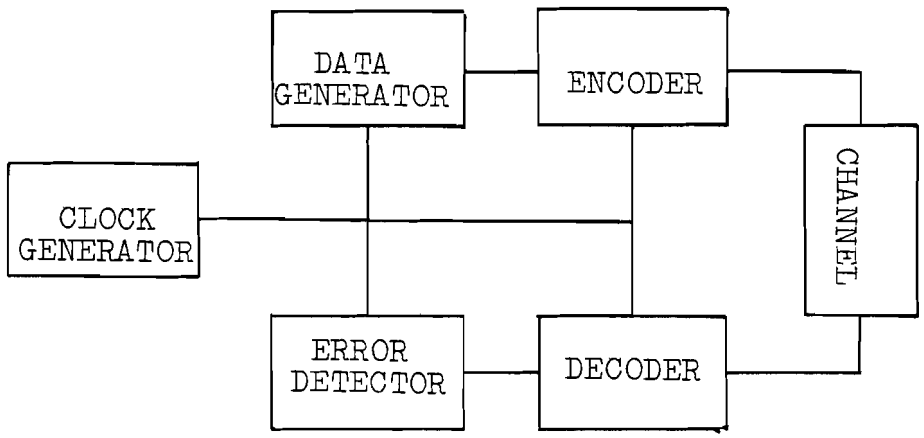


fig. 23 Schematic diagram of the test setup

The two code symbols c_1 and c_2 will be modulated on the same carrier by a QPSK modulator. A polar baseband channel is realized to test the decoder. A test channel using a QPSK-modem is not realized, since the available modem had to be adjusted for these measurements and the time was missing for the realization of this channel and the measurements. But as a baseband link is used, there have to be two identical channels (fig. 24). A RS-232 driver (75188) is used to convert the TTL output of the encoder into a bipolar signal.The measurement setup works with a 50 ohm impedance and therefore a amplifier is used to adjust the output impedance of the converter. In the test setup, the codec system works at a rate of 250 kbit/s

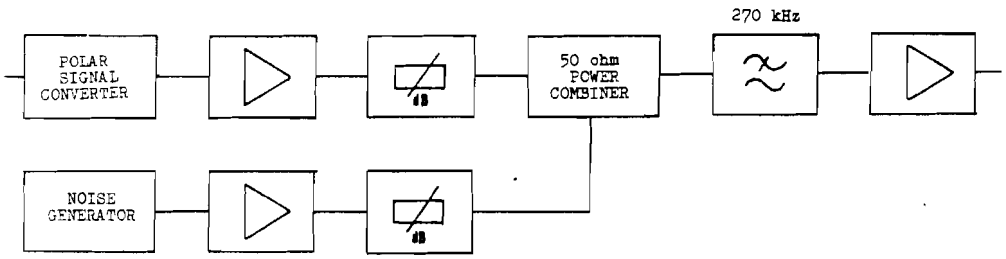


fig. 24 Schematic diagram of the bipolar channel

The noise sourve delivers gaussian noise spectrum flat over 500 kHz. This noise source has a output impedance of 600 ohm, and therefore a amplifier is used to convert the output impedance. The channel filter is a 5th order Butterworth lowpass filter with cutoff frequency f =270 kHz. The power combiner is a resistor network and has a loss of 6 dB. Therefore a amplifier is inserted after the channel filter to provide the proper input level for the decoder. The noise power density is measured at the output of the channel with a spectrum analyzer. The noise power measured with an attenuation of 0 dB is 4.8 dBm. Likewise, the signal power measured at the output is 13.8 dBm.

5.2. The results

All the computed bit error probability curves are also compared with the measured results. These measurements are carried out without differential coding because a baseband channel has no phase ambiguity. Thus the computed and measured results can be compared. The BER curves (fig.28-29) for the threshold normalized to the signal energy per code symbol (β) and for the threshold normalized to the noise power (α) are measured with E_b/N_0 =4, 5 and 6 dB. This is the same as E_s/N_0 =1, 2 and 3 dB, because one information bit consist of two code bits and therefore the energy per informationbit E_b is twice the signal energy per code symbol E_s. The BER curves are measured as function of E_b/N_0 with the near optimum values of both the normalized thresholds (fig. 30). The results of the measurements of all these curves are represented in the following tables.

β	E_b/N_o (dB) 6	5	4
0.16	$3.6\ 10^{-5}$	$2.1\ 10^{-4}$	$1.3\ 10^{-3}$
0.20	$2.7\ 10^{-5}$	$1.4\ 10^{-4}$	$1.1\ 10^{-3}$
0.25	$2.1\ 10^{-5}$	$1.2\ 10^{-4}$	$8.9\ 10^{-4}$
0.30	$1.8\ 10^{-5}$	$1.0\ 10^{-4}$	$8.9\ 10^{-4}$
0.35	$2.2\ 10^{-5}$	$1.4\ 10^{-4}$	$9.3\ 10^{-4}$
0.40	$2.6\ 10^{-5}$	$1.6\ 10^{-4}$	$1.0\ 10^{-3}$
0.45	$3.6\ 10^{-5}$	$1.9\ 10^{-4}$	$1.1\ 10^{-3}$
0.50	$4.8\ 10^{-5}$	$2.2\ 10^{-4}$	$1.2\ 10^{-3}$
0.75	$1.1\ 10^{-4}$	$3.8\ 10^{-4}$	$1.6\ 10^{-3}$

Table 5 BER results as function of the normalized threshold spacing β

α	E_s/N_o (db) 6	5	4
0.27	$1.6\ 10^{-4}$	$4.2\ 10^{-4}$	$1.4\ 10^{-3}$
0.34	$1.1\ 10^{-4}$	$2.8\ 10^{-4}$	$1.1\ 10^{-3}$
0.43	$6.3\ 10^{-5}$	$2.2\ 10^{-4}$	$8.3\ 10^{-4}$
0.51	$4.9\ 10^{-5}$	$1.9\ 10^{-4}$	$8.2\ 10^{-4}$
0.60	$4.9\ 10^{-5}$	$1.7\ 10^{-4}$	$7.5\ 10^{-4}$
0.68	$4.7\ 10^{-5}$	$1.9\ 10^{-4}$	$8.2\ 10^{-4}$
0.77	$5.4\ 10^{-5}$	$2.0\ 10^{-4}$	$8.7\ 10^{-4}$
0.86	$6.6\ 10^{-5}$	$2.4\ 10^{-4}$	$1.0\ 10^{-3}$
1.07	$8.2\ 10^{-5}$	$3.4\ 10^{-4}$	$1.3\ 10^{-3}$
1.29	$1.3\ 10^{-4}$	$4.9\ 10^{-4}$	$1.8\ 10^{-3}$

Table 6 BER results as function of the normalized threshold spacing α

E_b/N_o (dB)	β=0.28	α=0.6
3.0	$4.3\ 10^{-3}$	$3.3\ 10^{-3}$
3.5	$2.1\ 10^{-3}$	$1.4\ 10^{-3}$
4.0	$8.7\ 10^{-4}$	$8.1\ 10^{-4}$
4.5	$4.3\ 10^{-4}$	$2.9\ 10^{-4}$
5.0	$1.0\ 10^{-4}$	$1.3\ 10^{-4}$
5.5	$5.2\ 10^{-5}$	$6.3\ 10^{-5}$
6.0	$1.5\ 10^{-5}$	$2.5\ 10^{-5}$
6.5	$5.2\ 10^{-6}$	$1.2\ 10^{-5}$
7.0	$2.0\ 10^{-6}$	$5.2\ 10^{-6}$
7.5	$5.8\ 10^{-7}$	-----
8.0	$7.8\ 10^{-8}$	-----

Table 7 BER results as function of E_b/N_o

Although the decoder is a 8-level soft decision decoder, the BER curves for Q=4 are also measured. The least significant bit of the digital input words is then made a "0". Therefore the quantization scheme will change into (fig. 19):

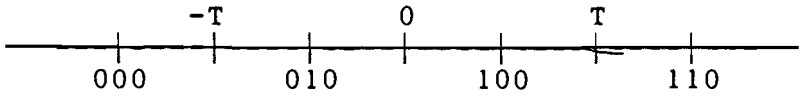


fig. 19 Modified quantization scheme

The BER curves as function of the threshold normailzed to the noise power (fig.25) and the threshold normalized to the signal energy per code symbol (fig. 26) are also measured for Q=4. The BER curve as function of E_b/N_o is measured (fig. 27) and the results are represented in the following tables.

β	E_b / N_o (dB)		
	6	5	4
0.30	$3.7\ 10^{-5}$	$4.6\ 10^{-4}$	$1.7\ 10^{-3}$
0.36	$2.9\ 10^{-5}$	$3.4\ 10^{-4}$	$1.5\ 10^{-3}$
0.45	$2.2\ 10^{-5}$	$2.8\ 10^{-4}$	$1.3\ 10^{-3}$
0.55	$2.3\ 10^{-5}$	$2.0\ 10^{-4}$	$1.2\ 10^{-3}$
0.64	$3.7\ 10^{-5}$	$2.7\ 10^{-4}$	$1.5\ 10^{-3}$
0.73	$7.0\ 10^{-5}$	$3.9\ 10^{-4}$	$1.8\ 10^{-3}$
0.82	$1.3\ 10^{-4}$	$5.4\ 10^{-4}$	$2.5\ 10^{-3}$
0.91	$2.5\ 10^{-4}$	$8.6\ 10^{-4}$	$3.5\ 10^{-3}$

Table 8 BER results as function of the normlized threshold spacing β

α	E_b / N_o (dB)		
	6	5	4
0.27	$2.8\ 10^{-4}$	$1.4\ 10^{-3}$	$5.2\ 10^{-3}$
0.32	$2.4\ 10^{-4}$	$1.2\ 10^{-3}$	$4.6\ 10^{-3}$
0.38	$2.0\ 10^{-4}$	$1.1\ 10^{-3}$	$3.9\ 10^{-3}$
0.43	$1.7\ 10^{-4}$	$1.0\ 10^{-3}$	$3.3\ 10^{-3}$
0.49	$1.6\ 10^{-4}$	$9.3\ 10^{-4}$	$3.3\ 10^{-3}$
0.54	$1.5\ 10^{-4}$	$9.3\ 10^{-4}$	$3.5\ 10^{-3}$
0.59	$1.8\ 10^{-4}$	$1.0\ 10^{-3}$	$4.1\ 10^{-3}$
0.65	$1.9\ 10^{-4}$	$1.2\ 10^{-3}$	$4.7\ 10^{-3}$
0.70	$2.3\ 10^{-4}$	$1.3\ 10^{-3}$	$5.7\ 10^{-3}$
0.76	$2.4\ 10^{-4}$	$1.3\ 10^{-3}$	$6.7\ 10^{-3}$

Table 9 BER results as function of the normalized threshold spacing α

E /N (dB)	$\beta = 0.5$	$\alpha = 0.54$
3.0	$5.7\ 10^{-3}$	$1.8\ 10^{-2}$
3.5	$2.9\ 10^{-3}$	$7.5\ 10^{-3}$
4.0	$1.4\ 10^{-3}$	$3.6\ 10^{-3}$
4.5	$5.3\ 10^{-4}$	$2.0\ 10^{-3}$
5.0	$2.1\ 10^{-4}$	$8.3\ 10^{-4}$
5.5	$6.8\ 10^{-5}$	$3.7\ 10^{-4}$
6.0	$2.9\ 10^{-5}$	$1.5\ 10^{-4}$
6.5	$8.1\ 10^{-6}$	$6.8\ 10^{-5}$
7.0	$3.8\ 10^{-6}$	$2.1\ 10^{-5}$
7.5	$8.3\ 10^{-7}$	-----
8.0	$2.6\ 10^{-7}$	-----

Table 10 BER results as fuction of E_b / N_o

For comparison, the bit error rate is measured for data sent over the polar baseband channel witout coding. As receiver the a hard decision device is used. We use the most significant bit of the A/D converter of the decoder as hard decision device. The results are presented in the following table

E /N (dB)	without coding
6.0	$2.4\ 10^{-3}$
6.5	$1.4\ 10^{-3}$
7.0	$7.8\ 10^{-4}$
7.5	$4.0\ 10^{-4}$
8.0	$2.0\ 10^{-4}$
8.5	$8.6\ 10^{-5}$
9.0	$3.7\ 10^{-5}$
9.5	$1.2\ 10^{-5}$
10.0	$3.7\ 10^{-6}$

Table 11 BER result as function of E_b / N_o

The computed and measured BER curves are drawn in the same diagrams. We note that the measured results for the low levels of E_b / N_o agree well with the computed results. For higher levels of E_b / N_o, the measured results differ 0.8 dB for the measurements as function of the threshold normalized to the noise power and 0.2 dB for the measurements as function of the threshold normalized to the signal energy. The degradation is caused by a deviation of the equally spaced thresholds. We observe that the degradation of the measurements as function of the threshold normalized to the noise power is greater as

well for Q=4 as for Q=8. In this case, the noise was kept power constant at its highest level (N =4.8 dBm) and the signal energy was varied. As the noise has a higher peak-to-rms voltage ratio, the amplifier is probably more subjected to saturation. The code gain is determined by comparing the bit error rate curves of the decoder with the bit error rate curve measured without coding. A receiver with an AGC will have a code gain of 3.1 dB for Q=4 and a code gain of 3.4 for Q=8 at a bit error rate of 10 .

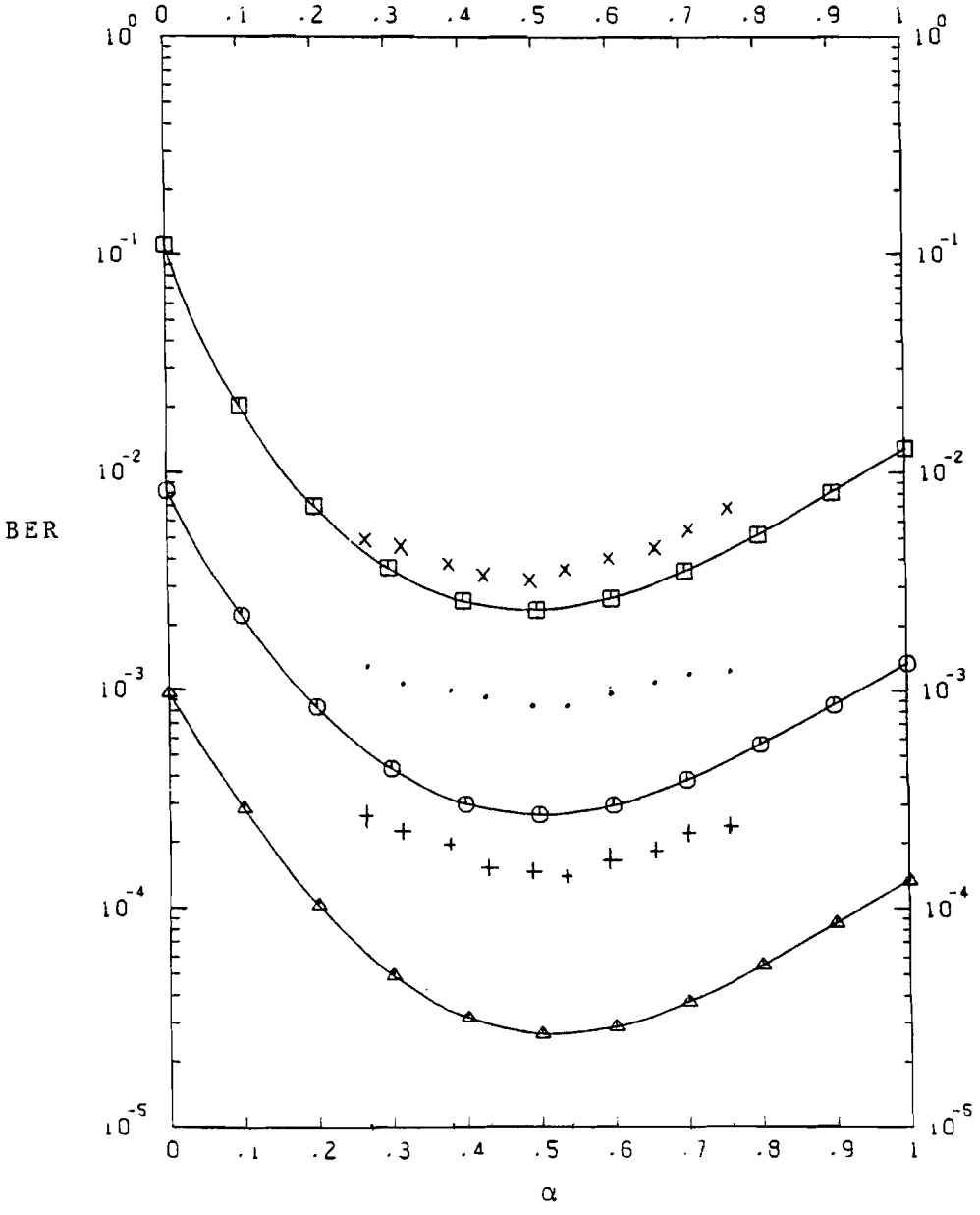


fig. 25 The BER curves as function of the threshold spacing normalized to the noise power density (α) with E_b/N_o as parameter (Q=4)

	MEASURED	COMPUTED
E_b/N_o = 6 dB	+	△
E_b/N_o = 5 dB	·	○
E_b/N_o = 4 dB	×	□

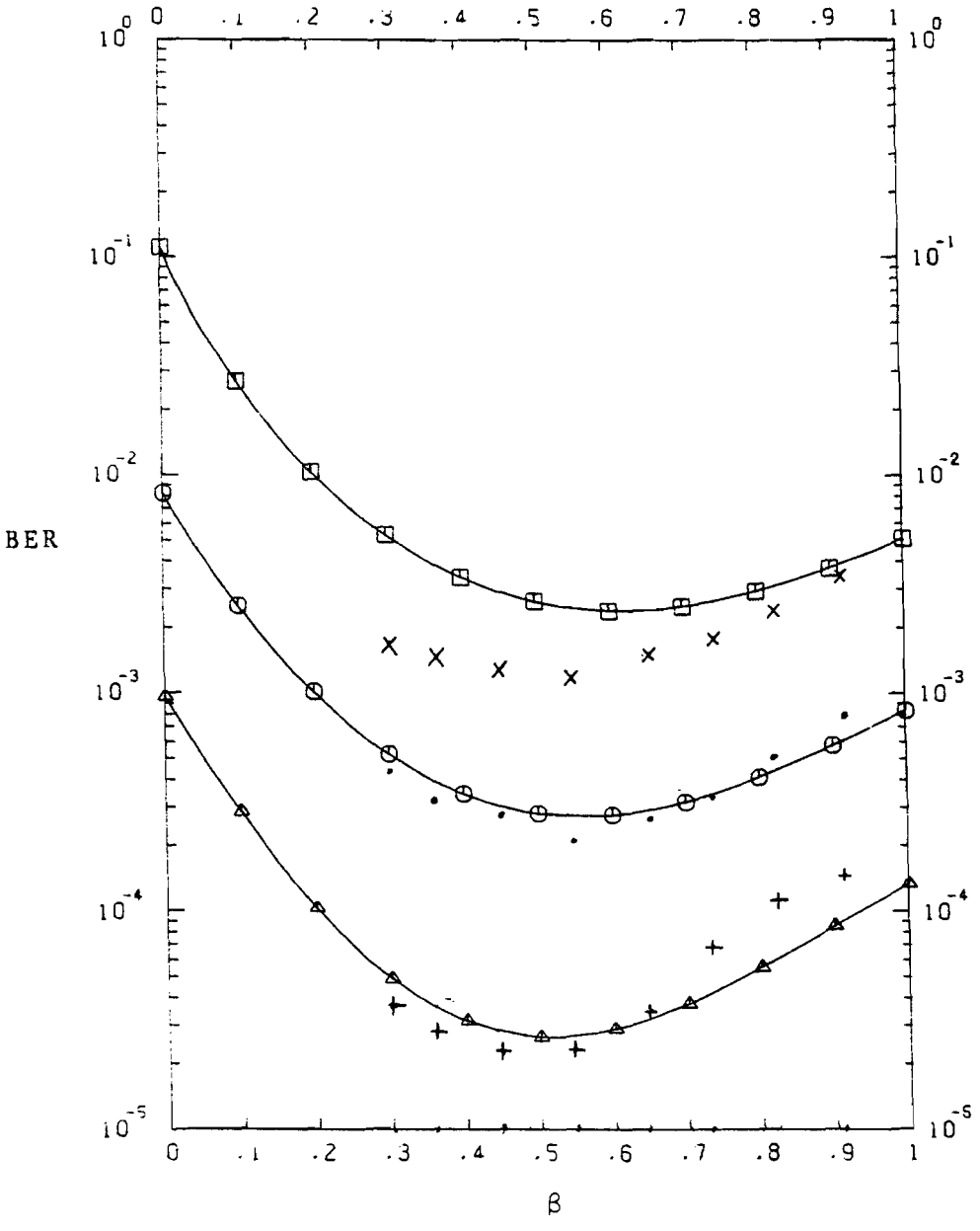


fig. 26 The BER curves as function of the threshold spacing normalized to the signal energy per code symbol (β) with parameter E_b/N_o (Q=4).

	MEASURED	COMPUTED
E_b/N_o = 6 dB	+	△
E_b/N_o = 5 dB	•	○
E_b/N_o = 4 dB	×	□

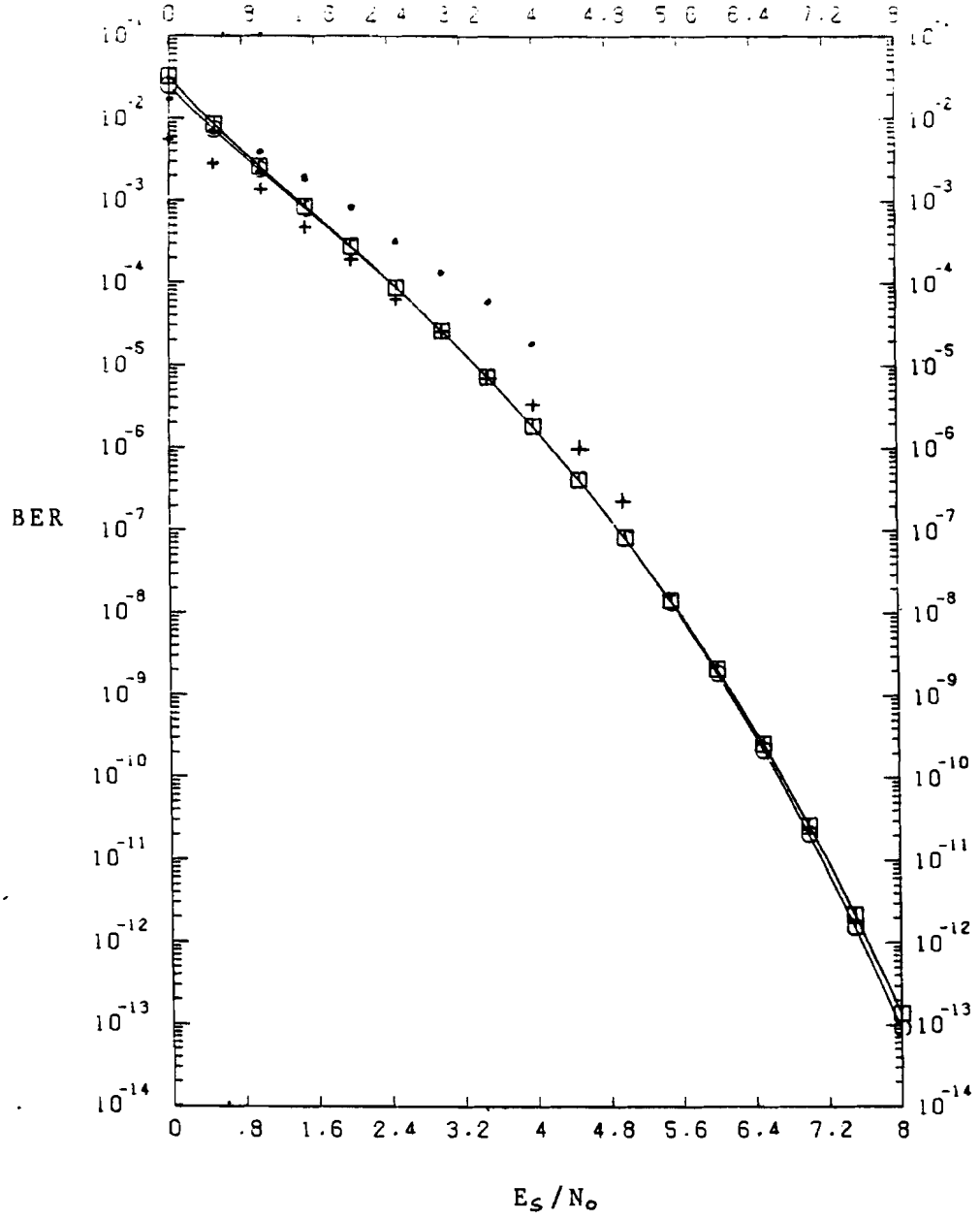


fig.27 The BER curves as function of E_S/N_o with parameters the threshold spacing normalized to the noise power (α) and the threshold spacing normalized to the signal energy per code symbol (β).

MEASURED		COMPUTED	
α = 0,54	•	α = 0.5	o
β = 0.5	+	β = 0.5	□

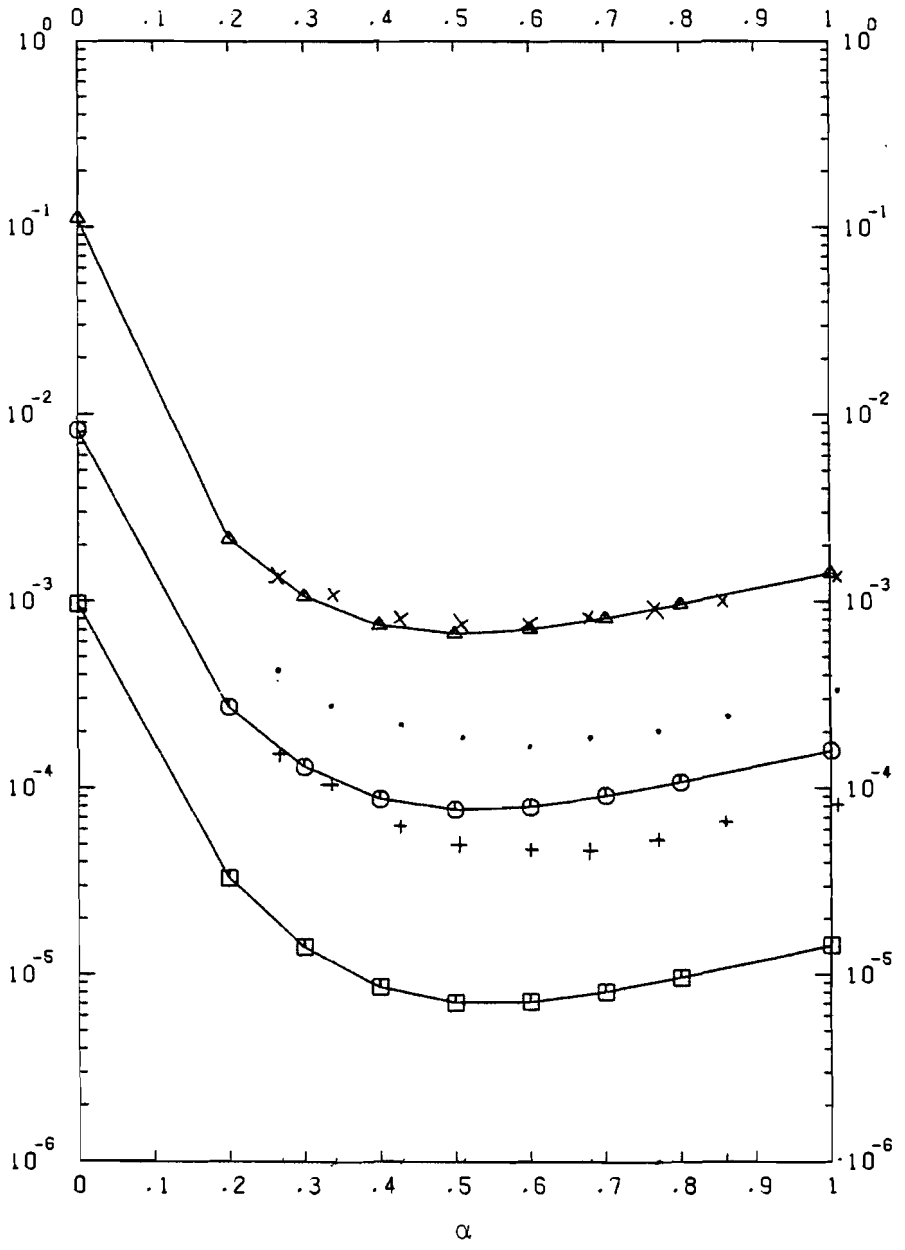


fig. 28 The BER curves as function of the threshold spacing normalized to the noise power density (α) with E_b/N_o as parameter (Q=8).

	MEASURED	COMPUTED
E_b/N_o = 6 dB	+	□
E_b/N_o = 5 dB	•	○
E_b/N_o = 4 dB	×	△

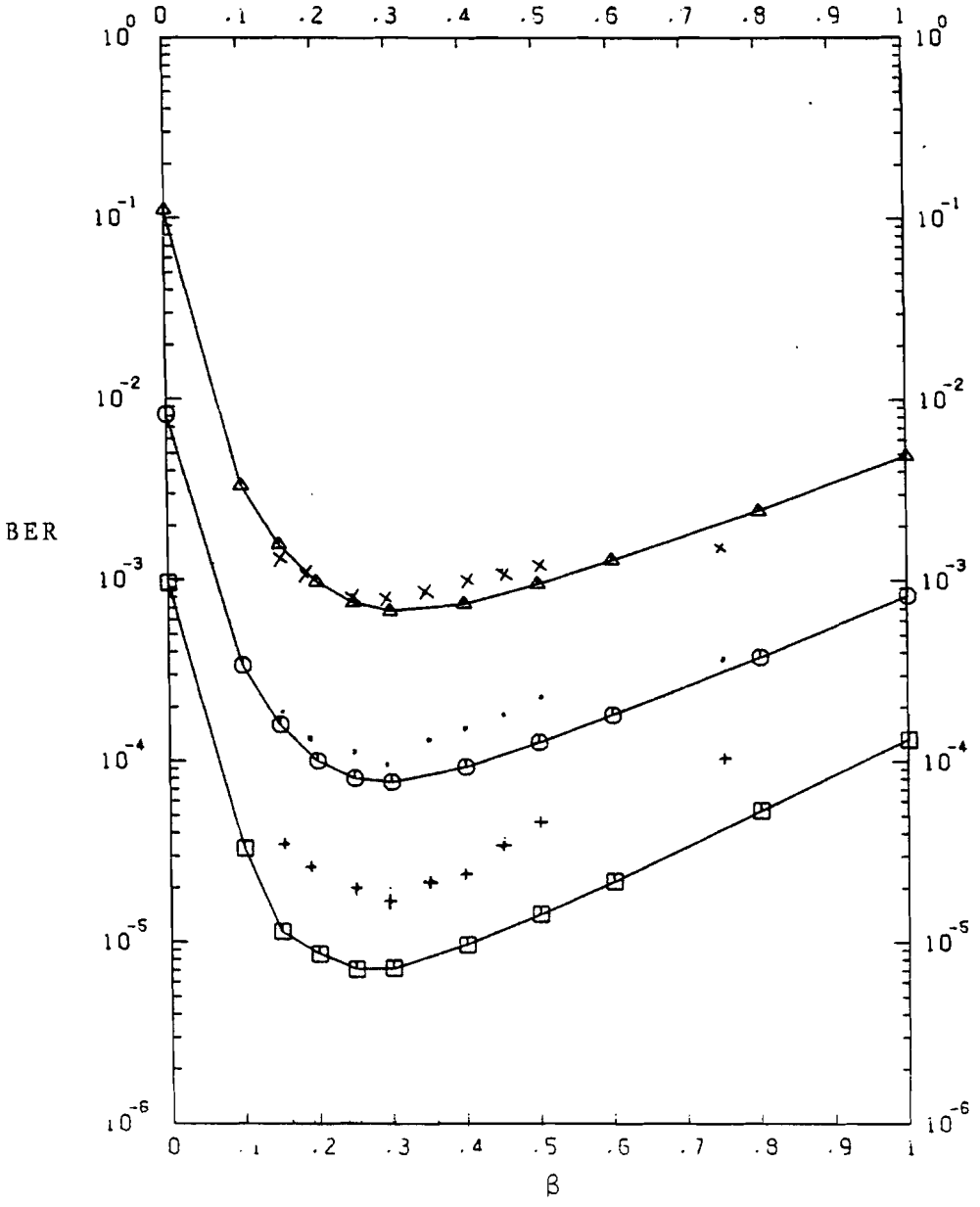


fig. 29 The BER curves as function of the threshold spacing normalized to the signal energy per code symbol (β) with E_b/N_o as parameter (Q=8).

	MEASURED	COMPUTED
E_b/N_o = 6 dB	+	□
E_b/N_o = 5 dB	·	○
E_b/N_o = 4 dB	×	△

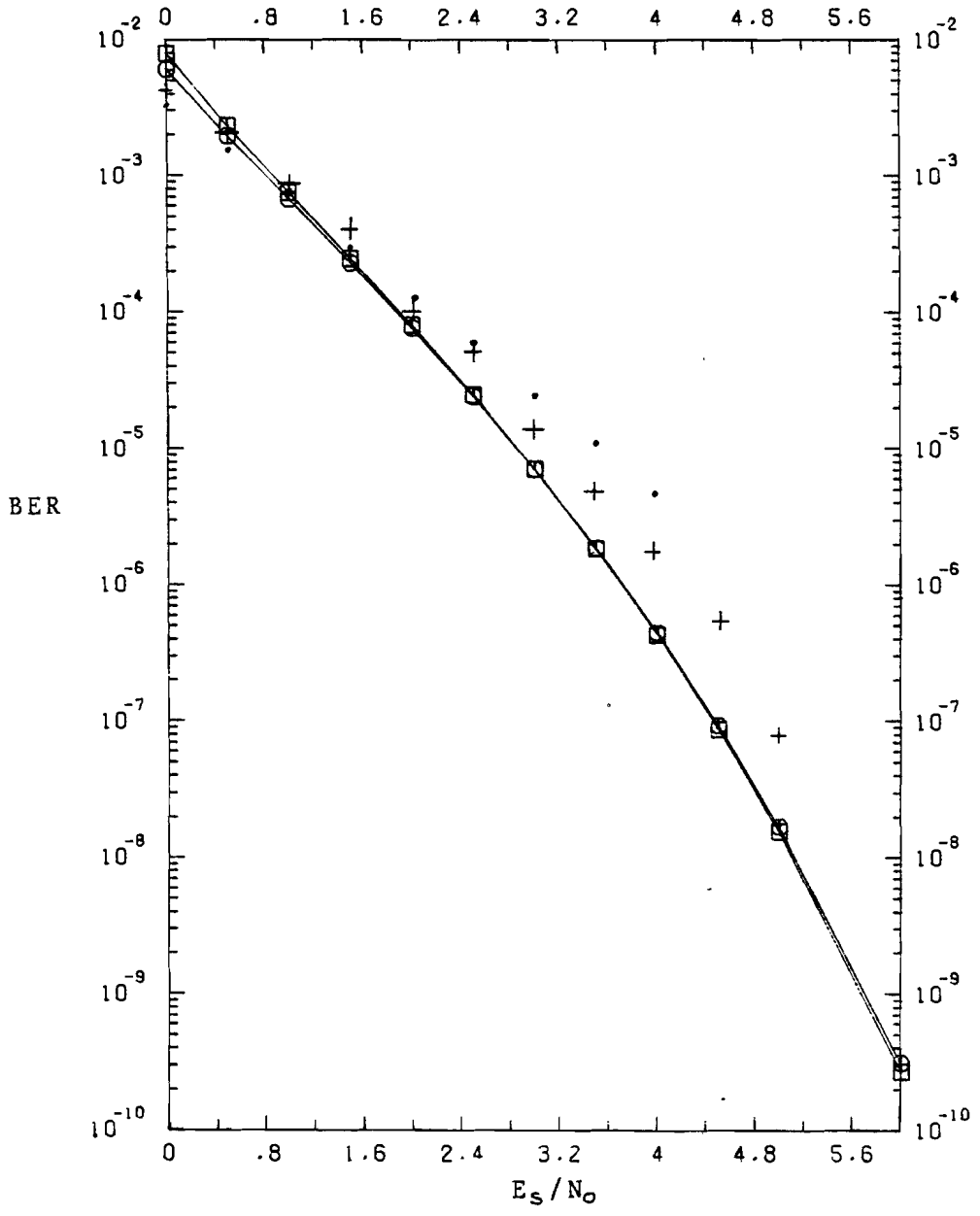


fig.30 The BER curves as function of E_S/N_O with parameters the threshold spacing normalized to the noise power density (α) and the threshold spacing normalized to the signal energy per symbol (β).

	MEASURED		COMPUTED
$\alpha = 0.6$	•	$\alpha = 0.5$	○
$\beta = 0.28$	+	$\beta = 0.25$	□

6. CONCLUSION

A soft decision Viterbi decoder has been designed and built on a 22.5 x 28 cm wire-wrapped board. The theoretical performance of the Viterbi codec is investigated. The bound on the bit error rate (BER) of the code was derived and computed. The BER performance was measure and the theoratical and measured results are compared.

- An upperbound on the bit error probability is computed as a function of α, the threshold normalized to the noise power density and as a function of β, the threshold normailzed to the signal energy per code symbol. Both for Q=8 an Q=4 qantization levels, the degration of the BER performance appears insignificant even if the threshold considerably differs from the optimum values of and . We can see that the optimum value of α is 0.5 for Q=4 as well as for Q=8, while the optimum value of β is 0.5 for Q=4 and 0.25 for Q=8 (at E_b/N_o=6 dB).

- The BER upper bound curves as function of E_b/N_o differ hardly at their optimum values of α and β. Judging from these theoretical curves, any configuration with or without an AGC can be chosen.

- The theoretical code gain of the decoder is 3.6 dB for Q=8 and 3.2 db for Q=4. The theoretical code gain is taken as the difference between the bit error probability for coherent detection of QPSK-modulated signals without and with coding at BER OF 10^{-5}.

- For transmission over a Gaussian channel using QPSK-modulation and coherent QPSK-demodulation, a phase error of $n*\pi/2$ (n=1,2,3) might occur.Thus the the in-phase and the quadrature-phase code symbols could be inverted independent of eachother. The decoder can only resolve the phase ambiguity of one of the code symbols. Therefore differential coding is applied to resolve the phase ambiguity of the other code symbol. However, the differential coding causes a loss of code gain of about 0.4 dB.

- The theoretical and measured BER curves as function of the normalized threshols α and β agree well. The BER measurements as function of β degrade less than 0.4 dB, and the BER measurements as function of α differ less than 0.8 dB, below the theoretical curves

- The decoder has a code gain of 3.1 dB for Q=4 and 3.4 dB for Q=8, as measured with the optimum value of β. The code gain for the optimum value of the threshold normalized to the noise power density (α) is 2.5 dB for Q=4 and 3.0 dB for Q=8. The measured code gain is the difference in E_b/N_o between the BER of the decoder and the BER without coding determined for a polar baseband channel realized an the laboratry.

- All measurements are carried out over this polar baseband channel. More realistic results could be attained by using QPSK-modulation in an AWGN IF-channel. Then the influences of the QPSK-demodulator degradation on the decoding process would be measured, too. Therefore it is recommended to repeat the measurements with a QPSK-modem and a simulated IF-channel.

- Altough the theoretical BER curves as function of E_b/N_o hardly differ, the difference of the two BER performances measured with the threshold normalized to the signal energy per code symbol and with the the threshold normalized to the noise power density, respecticaly, is about 0.4 dB. Thus a receiver with an AGC is recommende in pratice.

ACKNOWLEDGMENTS

The author wish to thank prof dr. J.C. Arnbak and ir. A.P. Verlijsdonk for their guidance and S.H. Mneney M.A.Sc. for the many useful discussions.

LITERATURE

[1] S.H. Mneney, A.P. Verlijsdonk,
"AUDIO-VISUAL BROADCAST TO RURAL AREAS OVER NARROWBAND SATELLITE CHANNELS",
IEEE Proc. AFRICON 83, pp.B.2.2.1-B.2.2.6,december 1983.

[2] S. Lin, D.J. Costello,
ERREOR CONTROL CODING,
Eglewood Cliffs, N.J.: Prentice-Hall, 1983.

[3] A.J Viterbi, J.K. Omura,
Principles of Digital Communication and Coding,
Tokyo: McGraw-Hill Kogakusha, 1979.

[4] J.A. Heller, I.M. Jacobs,
"Viterbi Decoding for Satellite and Space Communications",
IEEE Trans. Commun. Techn. Vol. COM-19, No. 5, pp. 835-848, October 1971.

[5] W.W. Peterson, E.J. Weldon,
Error-correcting codes, second edition,
MIT press, 1972.

[6] A.J. Viterbi,
"Convolutional Codes and Their Performance in Communication Systems",
IEEE Trans. Commun. Techn., Vol. COM-19, No. 5, pp. 751-772, October 1971.

[7] J.M.Wozencraft, I.M. Jacobs,
Principles of Communication Engeneering,
New York: John Wiley & Sons, 1965.

[8] Y. Yasuda, Y. Hirata, A. Ogawa,
"OPTIMUM SOFT DECISION FOR VITERBI DECODING",
5th Int. Conference on Digital Satellite Communications, pp. 251-258, March 1981.

[9] A. Hekstra,
"Een feasibility studie naar de toepassing van een R=1/2, k=3 soft decision Viterbi codec in een experimenteel videofoon systeem",
Stageverslag The ,vakgroep EC,1983.

[10] G.C. Clark,
"IMPLEMENTATION OF MAXIMUM LIKELIHOOD DECODERS FOR CONVOLUTIONAL CODES",
Int. Telemetering Conference, Vol. 7, pp. 499-463, September 1971.

[11] K. Feher,
DIGITAL COMMUNICATIONS
Englewood Cliffs, N.J.: Prentice-Hall, 1983.

[12] The TLL Data Book for Design Engineers, third european edition
Texas Instruments, 1979.

[13] A.S. Acampora, R.P. Gilmore,
"Analog Viterbi Decoding for High Speed Digital Satellite Channels",
IEEE Trans. Commun., Vol. COM-26, pp.1463-1470, October 1978.

[14] V.K. Bhargava, D. Haccoun, R Matyas, P.N. Nuspl,
DIGITAL COMMUNICATIONS BY SATELLITE,
New York: John Wiley & Sons, 1981.

[15] A.B. Carlson,
COMMUNICATION SYSTEMS, second edition,
Tokyo: McGraw-Hill Kogakusha, 1968.

[16] J.J. Spilker,
Digital Communications by Satellite
Englewood Cliffs, N.J.:Prentice-Hall, 1977

APPENDIX A: PROGAMS FOR BIT ERROR PRABABILITY COMPUTATION

A.1. VITERBIBOUND/Q4/E

```
100    %*************************************************************************%
200    %                                                                         %
300    %     THIS PROGRAM COMPUTES THE BIT ERROR PROBABILITY BOUND               %
400    %     FOR A SOFT-DECISION VITERBI DECODER WITH 4 QUANTIZATION             %
500    %     LEVELS AND USING A CONVOLUTIONAL CODE WITH CONSTRAINT               %
600    %     LENGTH K=3, RATE R=1/2 AND FREE DISTANCE D=5.                       %
700    %                                                                         %
800    %     WITH PARAMETERS ES/NO AND T THE THRESHOLD NORMALIZED                %
900    %     TO THE SIGNAL ENERGY PER CODE SYMBOL ES.                            %
1000   %                                                                         %
1100   %*************************************************************************%
1200   %
1300    BEGIN INTEGER A;
1400          FILE INPUT(KIND=READER),OUT(KIND=PRINTER);
1500          READ(INPUT,/,A);
1600    BEGIN REAL ARRAY P[0:3,1:A],F0:32],PK[1:A],QK[11:A],PB[1:A];
1700          REAL ARRAY ES[1:A],T[1:A];
1800          INTEGER K,N,H;
1900          PROCEDURE UPDATE;
2000   %
2100   %*************************************************************************%
2200   %                                                                         %
2300   %       THIS PROCEDURE COMPUTES THE TERMS Q(N) FOR A GIVEN                %
2400   %       COMBINATION OF L[I]                                               %
2500   %                                                                         %
2600   %*************************************************************************%
2700   %
2800          BEGIN INTEGER I,J;
2900                REAL Q;
3000                FOR J:=1 STEP 1 UNTIL A DO BEGIN
3100                Q:=F[K];
3200                FOR I:=0 STEP 1 UNTIL 3 DO Q:=Q/F[L[I]]*(P[I]**L[I]);
3300                QK[J]:=QK[J]+Q END
3400                END;
3500          PROCEDURE SORT;
3600   %
3700   %*************************************************************************%
3800   %                                                                         %
3900   %       THIS PROCEDURE COMPUTES ALL THE POSSIBLE PERMUTATIONS             %
4000   %       OF L[I] WHICH SATISFY THE FOLLOWING EQUATIONS:                    %
4100   %                                                                         %
4200   %       { SUMMATION L[I]} = K              (I FROM 0 TO 3 )               %
4300   %       { SUMMATION L[I]*I} = (3*K+N)/2    (I FROM 0 TO 3 )               %
4400   %                                                                         %
4500   %*************************************************************************%
4600   %
```

```
4700            BEGIN L[0]:=K+1;
4800                  WHILE L[3] NEQ K DO BEGIN
4900                  IF L[3] NEQ 0 THEN L[3]:=0
5000                              ELSE BEGIN
5100                  IF L[2] NEQ 0 THEN BEGIN L[2]:=L[2]-1;
5200                                         L[3]:=K-L[0]-L[1]-L[2] END
5300                              ELSE BEGIN
5400                  IF L[1] NEQ 0 THEN BEGIN L[1]:=L[1]-1;
5500                                         L[2]:=K-L[0]-L[1] END
5600                              ELSE BEGIN
5700                  IF L[0] NEQ 0 THEN BEGIN L[0]:=L[0]-1,
5800                                         L[1]:=K-L[0] END;
5900                                   END
6000                                   END;
6100                  IF (L[1]+2*L[2]+3*L[3])=(3*K+N)/2 THEN
6200                  UPDATE
6300               END
6400          END;
6500          PROCEDURE TRANSP;
6600    %
6700    %*************************************************************************%
6800    %                                                                         %
6900    %     THIS PROCEDURE COMPUTES THE TRANSITION PROBABILITIES P[I]           %
7000    %     FOR A GIVEN ES/NO AND NORMALIZED THRESHOLD TR.                      %
7100    %     THIS IS THE PROBABILITY THAT Y=I IS RECEIVED WHEN                   %
7200    %     X=0 WAS SENT.                                                       %
7300    %                                                                         %
7400    %*************************************************************************%
7500    %
7600          BEGIN REAL X;X:=SQRT(10**(ES[H]/10));
7700                P[3,H]:=ERFC((1+T[H])*X)/2;
7800                P[2,H]:=(ERFC((-1-T[H])*X)-ERFC(-1*X))/2;
7900                P[1,H]:=(ERFC(-1*X)-ERFC((T[H]-1)*X))/2;
8000                P[0,H]:=ERFC((T[H]-1)*X)/2;
8100          END;
8200    %
8300    %     FACTORIAL COMPUTATION
8400    %
8500          F:=0;
8600          FOR H:=1 STEP 1 UNTIL 32 DO F[H]:=F[H-1]*H;
8700    %
8800    %     INPUT
8900    %
9000          FOR H:=1 STEP 1 UNTIL A DO BEGIN
9100          READ(INPUT,/,T[H],ES[H]);TRANSP:PB[H]:=0 END;
9200    %
```

```
 9300    %*************************************************************************%
 9400    %                                                                         %
 9500    %      PROGRAM BODY                                                       %
 9600    %                                                                         %
 9700    %      THE BODY OF THE PROGRAM COMPUTES:                                  %
 9800    %                                                                         %
 9900    %      PB := (SUMMATION CK*PK )        ( K FROM 5 TO 32 )                 %
10000    %                                                                         %
10100    %*************************************************************************%
10200    %
10300          FOR K:=5 STEP 1 UNTIL 32 DO BEGIN
10400          FOR H:=1 STTEP 1 UNTIL A DO BEGIN
10500          PK[H]:=0:QK[H]:=0 END;N:=0;
10600          IF (N+K) MOD 2=0 THEN SORT;
10700          FOR H:=1 STEP 1 UNTIL A DO PKK[H]:=PK[H]+.5*QK[H];
10800          FOR H:=1 STEP 1 UNTIL 3*K DO BEGIN
10900          IF (N+K) MOD 2=0 THEN SORT:
11000          FOR H:=1 STEP 1 UNTIL A DO QK[H]:=0:
11100          FOR H:=1 STEP 1 UNTIL A DO PK[H]:=PK[H]+QK[H];
11200          END;
11300          FOR H:=1 STEP 1 UNTIL A DO PB[H]:=PB[H]+(K-4)*(2**(K-5))*PK[H]:
11400          END;
11500    %
11600    %      OUTPUT
11700    %
11800          FOR H:=1 STEP 1 UNTIL A DO WRITE(OUTPUT,<3E10>,T[H],ES[H],PB[H])
11900     END
12000     END.
```

A.1. VITERBIBOUND/Q4/N

```
100   %*************************************************************************%
200   %                                                                         %
300   %     THIS PROGRAM COMPUTES THE BIT ERROR PROBABILITY BOUND               %
400   %     FOR A SOFT-DECISION VITERBI DECODER WITH 4 QUANTIZATION             %
500   %     LEVELS AND USING A CONVOLUTIONAL CODE WITH CONSTRAINT               %
600   %     LENGTH K=3, RATE R=1/2 AND FREE DISTANCE D=5.                       %
700   %                                                                         %
800   %     WITH PARAMETERS ES/NO AND T THE THRESHOLD NORMALIZED                %
900   %     TO THE NOISE POWER DENSITY NO.                                      %
1000  %                                                                         %
1100  %*************************************************************************%
1200  %
1300   BEGIN INTEGER A;
1400         FILE INPUT(KIND=READER),OUT(KIND=PRINTER);
1500         READ(INPUT,/,A);
1600   BEGIN REAL ARRAY P[0:3,1:A],F[0:32],PK[1:A],QK[11:A],PB[1:A];
1700         REAL ARRAY ES[1:A],T[1:A];
1800         INTEGER K,N,H;
1900         PROCEDURE UPDATE;
2000  %
2100  %*************************************************************************%
2200  %                                                                         %
2300  %       THIS PROCEDURE COMPUTES THE TERMS Q(N) FOR A GIVEN                %
2400  %       COMBINATION OF L[I]                                               %
2500  %                                                                         %
2600  %*************************************************************************%
2700  %
2800         BEGIN INTEGER I,J;
2900               REAL Q;
3000               FOR J:=1 STEP 1 UNTIL A DO BEGIN
3100               Q:=F[K];
3200               FOR I:=0 STEP 1 UNTIL 3 DO Q:=Q/F[L[I]]*(P[I]**L[I]);
3300               QK[J]:=QK[J]+Q END
3400               END;
3500         PROCEDURE SORT;
3600  %
3700  %*************************************************************************%
3800  %                                                                         %
3900  %       THIS PROCEDURE COMPUTES ALL THE POSSIBLE PERMUTATIONS             %
4000  %       OF L[I] WHICH SATISFY THE FOLLOWING EQUATIONS:                    %
4100  %                                                                         %
4200  %       { SUMMATION L[I]} = K              (I FROM 0 TO 3 )               %
4300  %       { SUMMATION L[I]*I} = (3*K+N)/2    (I FROM 0 TO 3 )               %
4400  %                                                                         %
4500  %*************************************************************************%
4600  %
```

```
4700            BEGIN L[0]:=K+1;
4800                  WHILE L[3] NEQ K DO BEGIN
4900                  IF L[3] NEQ 0 THEN L[3]:=0
5000                                ELSE BEGIN
5100                  IF L[2] NEQ 0 THEN BEGIN L[2]:=L[2]-1;
5200                                           L[3]:=K-L[0]-L[1]-L[2] END
5300                                ELSE BEGIN
5400                  IF L[1] NEQ 0 THEN BEGIN L[1]:=L[1]-1;
5500                                           L[2]:=K-L[0]-L[1] END
5600                                ELSE BEGIN
5700                  IF L[0] NEQ 0 THEN BEGIN L[0]:=L[0]-1;
5800                                           L[1]:=K-L[0] END;
5900                                     END
6000                                     END;
6100                  IF (L[1]+2*L[2]+3*L[3])=(3*K+N)/2 THEN
6200                  UPDATE
6300               END
6400            END;
6500            PROCEDURE TRANSP;
6600      %
6700      %*************************************************************************%
6800      %                                                                         %
6900      %     THIS PROCEDURE COMPUTES THE TRANSITION PROBABILITIES P[I]           %
7000      %     FOR A GIVEN ES/NO AND NORMALIZED THRESHOLD TR.                      %
7100      %     THIS IS THE PROBABILITY THAT Y=I IS RECEIVED WHEN                   %
7200      %     X=0 WAS SENT.                                                       %
7300      %                                                                         %
7400      %*************************************************************************%
7500      %
7600            BEGIN REAL X,TR;X:=SQRT(10**(ES[H]/10));TR:=T[H]*SQRT(2);
7700                  P[3,H]:=ERFC(TR+X)/2;
7800                  P[2,H]:=(ERFC(-1*(TR+X))-ERFC(-1*X))/2;
7900                  P[1,H]:=(ERFC(-1*X)-ERFC(TR-X))/2;
8000                  P[0,H]:=ERFC(TR-X)/2;
8100            END;
8200      %
8300      %     FACTORIAL COMPUTATION
8400      %
8500            F:=0;
8600            FOR H:=1 STEP 1 UNTIL 32 DO F[H]:=F[H-1]*H;
8700      %
8800      %     INPUT
8900      %
9000            FOR H:=1 STEP 1 UNTIL A DO BEGIN
9100            READ(INPUT,/,T[H],ES[H]);TRANSP;PB[H]:=0 END;
9200      %
```

```
 9300    %*************************************************************************%
 9400    %                                                                         %
 9500    %      PROGRAM BODY                                                       %
 9600    %                                                                         %
 9700    %      THE BODY OF THE PROGRAM COMPUTES:                                  %
 9800    %                                                                         %
 9900    %      PB := (SUMMATION CK*PK )        ( K FROM 5 TO 32 )                 %
10000    %                                                                         %
10100    %*************************************************************************%
10200    %
10300          FOR K:=5 STEP 1 UNTIL 32 DO BEGIN
10400          FOR H:=1 STTEP 1 UNTIL A DO BEGIN
10500          PK[H]:=0;QK[H]:=0 END;N:=0;
10600          IF (N+K) MOD 2=0 THEN SORT:
10700          FOR H:=1 STEP 1 UNTIL A DO PKK[H]:=PK[H]+.5*QK[H]:
10800          FOR H:=1 STEP 1 UNTIL 3*K DO BEGIN
10900          IF (N+K) MOD 2=0 THEN SORT;
11000          FOR H:=1 STEP 1 UNTIL A DO QK[H]:=0:
11100          FOR H:=1 STEP 1 UNTIL A DO PK[H]:=PK[H]+QK[H]:
11200          END;
11300          FOR H:=1 STEP 1 UNTIL A DO PB[H]:=PB[H]+(K-4)*(2**(K-5))*PK[H];
11400          END;
11500    %
11600    %      OUTPUT
11700    %
11800          FOR H:=1 STEP 1 UNTIL A DO WRITE(OUTPUT,<3E10>,T[H],ES[H],PB[H])
11900     END
12000     END.
```

A.3. VITERBIBOUND/Q8/E

```
100    %*************************************************************************%
200    %                                                                         %
300    %   THIS PROGRAM COMPUTES THE BIT ERROR PROBABILITY BOUND                 %
400    %   FOR A SOFT DECISION VITERBI DECODER WITH 8 QUANTIZATION               %
500    %   LEVELS AND USING A CONVOLUTIONAL CODE WITH CONSTRAINT                 %
600    %   LENGTH K=3, RATE R=1/2 AND FREE DISTANCE D=5.                         %
700    %                                                                         %
800    %   WITH PARAMETERS ES/NO AND T THE THRESHOLD NORMALIZED                  %
900    %   TO THE SIGNAL ENERGY PER CODE SYMBOL ES                               %
1000   %                                                                         %
1100   %*************************************************************************%
1200   %
1300   BEGIN INTEGER K,N,H,A;
1400         FILE OUT(KIND=PRINTER),IN(KIND=READER);
1500         READ(IN,/,A);
1600   BEGIN REAL ARRAY PK[1:A],QK[1:A],PB[1:A],T[1:A],ES[1:A];
1700         INTEGER ARRAY L[0:7];
1800         REAL ARRAY P[0:7,1:A],F[0:32];
1900         PROCEDURE UPDATE;
2000   %
2100   %*************************************************************************%
2200   %                                                                         %
2300   %      THIS PROCEDURE COMPUTES THE TERMS Q(N) FOR A GIVEN                 %
2400   %      COMBINATION OF L[I]                                                %
2500   %                                                                         %
2600   %*************************************************************************%
2700   %
2800         BEGIN INTEGER I,J;
2900               REAL Q;
3000               FOR J:=1 STEP 1 UNTIL A DO BEGIN
3100               Q:=F[K];
3200               FOR I:=0 STEP 1 UNTIL 7 DO Q:=Q/F[L[I]]*(P[I,J]**L[I]);
3300               QK[J]:=QK[J]+Q END
3400         END;
3500         PROCEDURE SORT;
3600   %*************************************************************************%
3700   %                                                                         %
3800   %      THIS PROCEDURE COMPUTES ALL THE POSSIBLE PERMUTATIONS              %
3900   %      OF L[I] WHICH SATISFY THE FOLLOWING EQUATIONS:                     %
4000   %                                                                         %
4100   %      { SUMMATION L[I]} = K                  (I FROM 0 TO Q-1 )         %
4200   %      { SUMMATION L[I]*I} = {(Q-1)*K+N}/2    (I FROM 0 TO Q-1 )         %
4300   %                                                                         %
4400   %*************************************************************************%
4500   %
```

```
4600          BEGIN
4700                L[0]:=K+1;
4800                WHILE L[7] NEQ K DO BEGIN
4900                IF L[7] NEQ 0 THEN L[7]:=0
5000                                  ELSE BEGIN
5100                IF L[6] NEQ 0 THEN BEGIN L[6]:=L[6]-1;
5200                                  L[7]:=K-L[6]-L[5]-L[4]-L[3]-L[2]-L[1]-L[0] END
5300                                  ELSE BEGIN
5400                IF L[5] NEQ 0 THEN BEGIN L[5]:=L[5]-1;
5500                                  L[6]:=K-L[5]-L[4]-L[3]-L[2]-L[1]-L[0] END
5600                                  ELSE BEGIN
5700                IF L[4] NEQ 0 THEN BEGIN L[4]:=L[4]-1;
5800                                  L[5]:=K-L[4]-L[3]-L[2]-L[1]-L[0] END
5900                                  ELSE BEGIN
6000                IF L[3] NEQ 0 THEN BEGIN L[3]:=L[3]-1;
6100                                  L[4]:=K-L[3]-L[2]-L[1]-L[0] END
6200                                  ELSE BEGIN
6300                IF L[2] NEQ 0 THEN BEGIN L[2]:=L[2]-1;
6400                                  L[3]:=K-L[2]-L[1]-L[0] END
6500                                  ELSE BEGIN
6600                IF L[1] NEQ 0 THEN BEGIN L[1]:=L[1]-1;
6700                                  L[2]:=K-L[1]-L[0] END
6800                                  ELSE BEGIN
6900                IF L[0] NEQ 0 THEN BEGIN L[0]:=L[0]-1;L[1]:=K-L[0] END;
7000                                       END END END END END END;
7100          IF (L[1]+2*L[2]+3*L[3]+4*L[4]+5*L[5]+6*L[6]+7*L[7])=(7*K+N)/2 THEN
7200          UPDATE
7300                                       END
7400                                       END
7500          END;
7600          PROCEDURE TRANSP;
7700    %*************************************************************************%
7800    %                                                                         %
7900    %     THIS PROCEDURE COMPUTES THE TRANSITION PROBABILITIES P[I]           %
8000    %     FOR A GIVEN ES/NO AND NORMALIZED THRESHOLD T.                       %
8100    %     THIS IS THE PROBABILITY THAT Y=I IS RECEIVED WHEN                   %
8200    %     X=0 WAS SENT.                                                       %
8300    %                                                                         %
8400    %*************************************************************************%
8500    %
8600          BEGIN INTEGER I;
8700                REAL X;
8800                X:=SQRT(10**(ES[H]/10));
8900                P[7,H]:=ERFC((3*T[H]+1)*X)/2;
9000                FOR I:=1 STEP 1 UNTIL 6 DO
9100                P[7-I,H]:=(ERFC(((I-4)*T[H]-1)*X)-ERFC(((I-3)*T[H]-1)*X))/2;
9200                 P[0,H]:=ERFC((3*T[H]-1)*X)/2
9300          END;
```

```
 9400    %
 9500    %      FACTORIAL COMPUTATION
 9600    %
 9700           F[0]:=1:
 9800           FOR H:=1 STEP 1 UNTIL 32 DO F[H]:=F[H-1]*H;
 9900    %
10000    %      INPUT
10100    %
10200           FOR H:=1 STEP 1 UNTIL A DO BEGIN
10300           READ(IN,/,T[H],ES[H]);TRANSP;PB[H]:=0 END;
10400    %
10500    %**************************************************************************%
10600    %                                                                          %
10700    %      PROGRAM BODY                                                        %
10800    %                                                                          %
10900    %      THE BODY OF THE PROGRAM COMPUTES:                                   %
11000    %                                                                          %
11100    %      PB := (SUMMATION CK*PK )      ( K FROM 5 TO 32 )                    %
11200    %                                                                          %
11300    %**************************************************************************%
11400    %
11500           FOR K:=5 STEP 1 UNTIL 32 DO BEGIN
11600           FOR H:=1 STEP 1 UNTIL A DO BEGIN
11700           PK[H]:=0;QK[H]:=0 END;N:=0.
11800           IF (N+K) MOD 2=0 THEN SORT·
11900           FOR H:=1 STEP 1 UNTIL A DO PK[H]:=PK[H]+.5*QK[H];
12000           FOR N:=1 STEP 1 UNTIL 7*K DO BEGIN
12100           FOR H:=1 STEP 1 UNTIL A DO QK[H]:=0;
12200           IF (N+K) MOD 2=0 THEN SORT;
12300           FOR H:=1 STEP 1 UNTIL A DO PK[H]:=PK[H]+QK[H]
12400                                          END;
12500           FOR H:=1 STEP 1 UNTIL A DO PB[H]:=PB[H]+(K-4)*(2**(K-5))*PK[H];
12600                                         END;
12700    %
12800    %      OUTPUT
12900    %
13000           FOR H:=1 STEP 1 UNTIL A DO WRITE(OUT,<3E10.3>,T[H],ES[H],PB[H])
13100     END
13200     END.
```

A.4. VITERBIBOUND/Q8/N

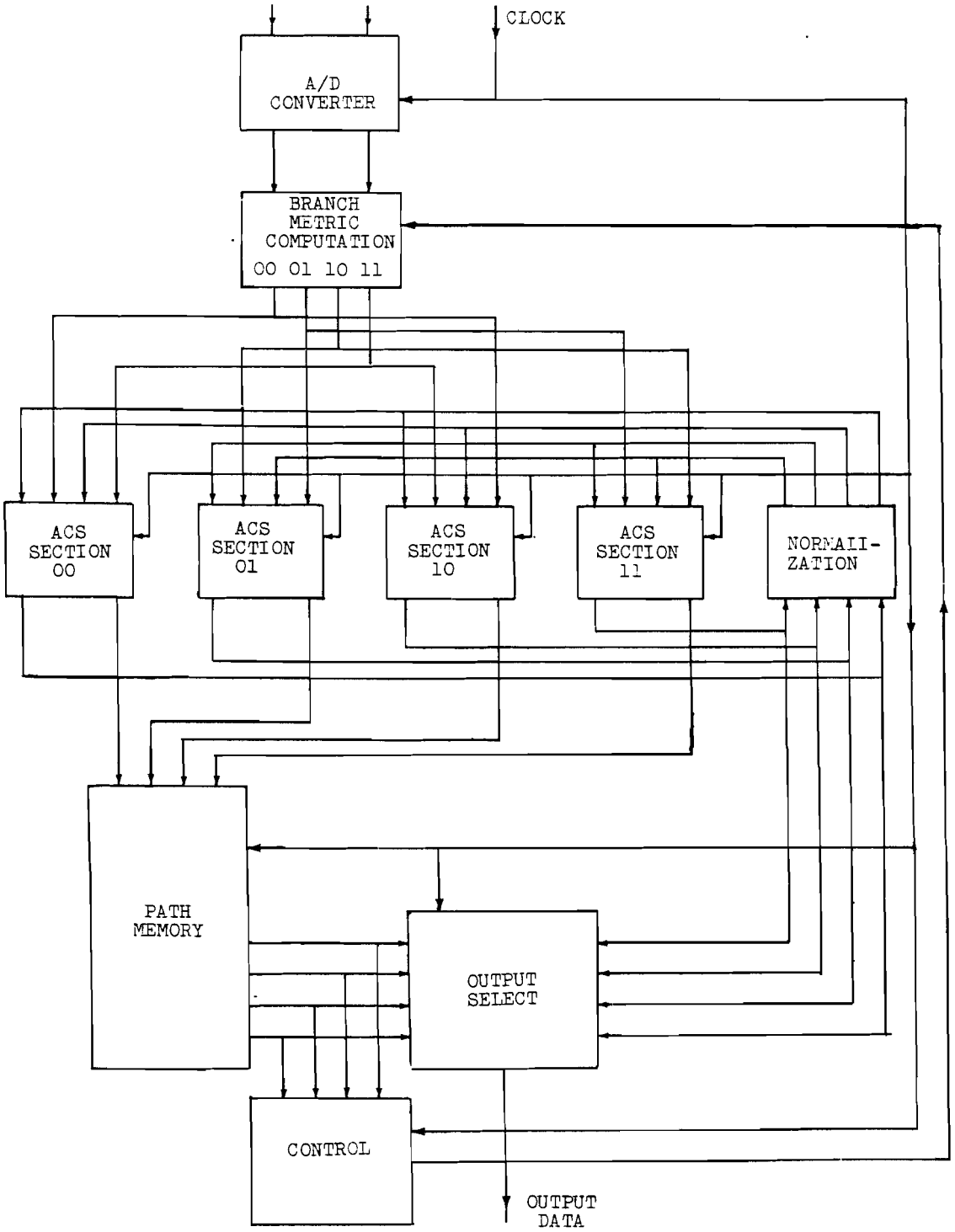
```
100   %*************************************************************************%
200   %                                                                         %
300   %   THIS PROGRAM COMPUTES THE BIT ERROR PROBABILITY BOUND                 %
400   %   FOR A SOFT DECISION VITERBI DECODER WITH 8 QUANTIZATION               %
500   %   LEVELS AND USING A CONVOLUTIONAL CODE WITH CONSTRAINT                 %
600   %   LENGTH K=3, RATE R=1/2 AND FREE DISTANCE D=5.                         %
700   %                                                                         %
800   %   WITH PARAMETERS ES/NO AND T THE THRESHOLD NORMALIZED                  %
900   %   TO THE NOISE POWER DENSITY NO.                                        %
1000  %                                                                         %
1100  %*************************************************************************%
1200  %
1300  BEGIN INTEGER K,N,H,A;
1400        FILE OUT(KIND=PRINTER),IN(KIND=READER);
1500        READ(IN,/,A);
1600  BEGIN REAL ARRAY PK[1:A],QK[1:A],PB[1:A],T[1:A],ES[1:A];
1700        INTEGER ARRAY L[0:7];
1800        REAL ARRAY P[0:7,1:A],F[0:32];
1900        PROCEDURE UPDATE;
2000  %
2100  %*************************************************************************%
2200  %                                                                         %
2300  %     THIS PROCEDURE COMPUTES THE TERMS Q(N) FOR A GIVEN                  %
2400  %     COMBINATION OF L[I]                                                 %
2500  %                                                                         %
2600  %*************************************************************************%
2700  %
2800        BEGIN INTEGER I,J;
2900              REAL Q;
3000              FOR J:=1 STEP 1 UNTIL A DO BEGIN
3100              Q:=F[K];
3200              FOR I:=0 STEP 1 UNTIL 7 DO Q:=Q/F[L[I]]*(P[I,J]**L[I]);
3300              QK[J]:=QK[J]+Q END
3400        END;
3500        PROCEDURE SORT;
3600  %*************************************************************************%
3700  %                                                                         %
3800  %     THIS PROCEDURE COMPUTES ALL THE POSSIBLE PERMUTATIONS               %
3900  %     OF L[I] WHICH SATISFY THE FOLLOWING EQUATIONS:                      %
4000  %                                                                         %
4100  %     { SUMMATION L[I]} = K                (I FROM 0 TO Q-1 )             %
4200  %     { SUMMATION L[I]*I} = {(Q-1)*K+N}/2  (I FROM 0 TO Q-1 )             %
4300  %                                                                         %
4400  %*************************************************************************%
4500  %
```

```
4600          BEGIN
4700                L[0]:=K+1;
4800                WHILE L[7] NEQ K DO BEGIN
4900                IF L[7] NEQ 0 THEN L[7]:=0
5000                              ELSE BEGIN
5100                IF L[6] NEQ 0 THEN BEGIN L[6]:=L[6]-1;
5200                              L[7]:=K-L[6]-L[5]-L[4]-L[3]-L[2]-L[1]-L[0] END
5300                              ELSE BEGIN
5400                IF L[5] NEQ 0 THEN BEGIN L[5]:=L[5]-1;
5500                              L[6]:=K-L[5]-L[4]-L[3]-L[2]-L[1]-L[0] END
5600                              ELSE BEGIN
5700                IF L[4] NEQ 0 THEN BEGIN L[4]:=L[4]-1;
5800                              L[5]:=K-L[4]-L[3]-L[2]-L[1]-L[0] END
5900                              ELSE BEGIN
6000                IF L[3] NEQ 0 THEN BEGIN L[3]:=L[3]-1;
6100                              L[4]:=K-L[3]-L[2]-L[1]-L[0] END
6200                              ELSE BEGIN
6300                IF L[2] NEQ 0 THEN BEGIN L[2]:=L[2]-1;
6400                              L[3]:=K-L[2]-L[1]-L[0] END
6500                              ELSE BEGIN
6600                IF L[1] NEQ 0 THEN BEGIN L[1]:=L[1]-1,
6700                              L[2]:=K-L[1]-L[0] END
6800                              ELSE BEGIN
6900                IF L[0] NEQ 0 THEN BEGIN L[0]:=L[0]-1;L[1]:=K-L[0] END;
7000                                   END END END END END END;
7100          IF (L[1]+2*L[2]+3*L[3]+4*L[4]+5*L[5]+6*L[6]+7*L[7])=(7*K+N)/2 THEN
7200          UPDATE
7300                                   END
7400                                   END
7500          END;
7600          PROCEDURE TRANSP;
7700    %*************************************************************************%
7800    %                                                                         %
7900    %     THIS PROCEDURE COMPUTES THE TRANSITION PROBABILITIES P[I]           %
8000    %     FOR A GIVEN ES/NO AND NORMALIZED THRESHOLD T.                       %
8100    %     THIS IS THE PROBABILITY THAT Y=I IS RECEIVED WHEN                   %
8200    %     X=0 WAS SENT.                                                       %
8300    %                                                                         %
8400    %*************************************************************************%
8500    %
8600          BEGIN INTEGER I;
8700                REAL X,TR;
8800                X:=SQRT(10**(ES[H]/10));TR:=T[H]/SQRT(2);
8900                P[7,H]:=ERFC((3*TR)+X)/2;
9000                FOR I:=1 STEP 1 UNTIL 6 DO
9100                P[I,H]:=(ERFC(((I-4)*TR)+X)-ERFC(((I-3)*TR)+X))/2;
9200                P[0,H]:=ERFC((3*TR)-X)/2
9300          END.
```

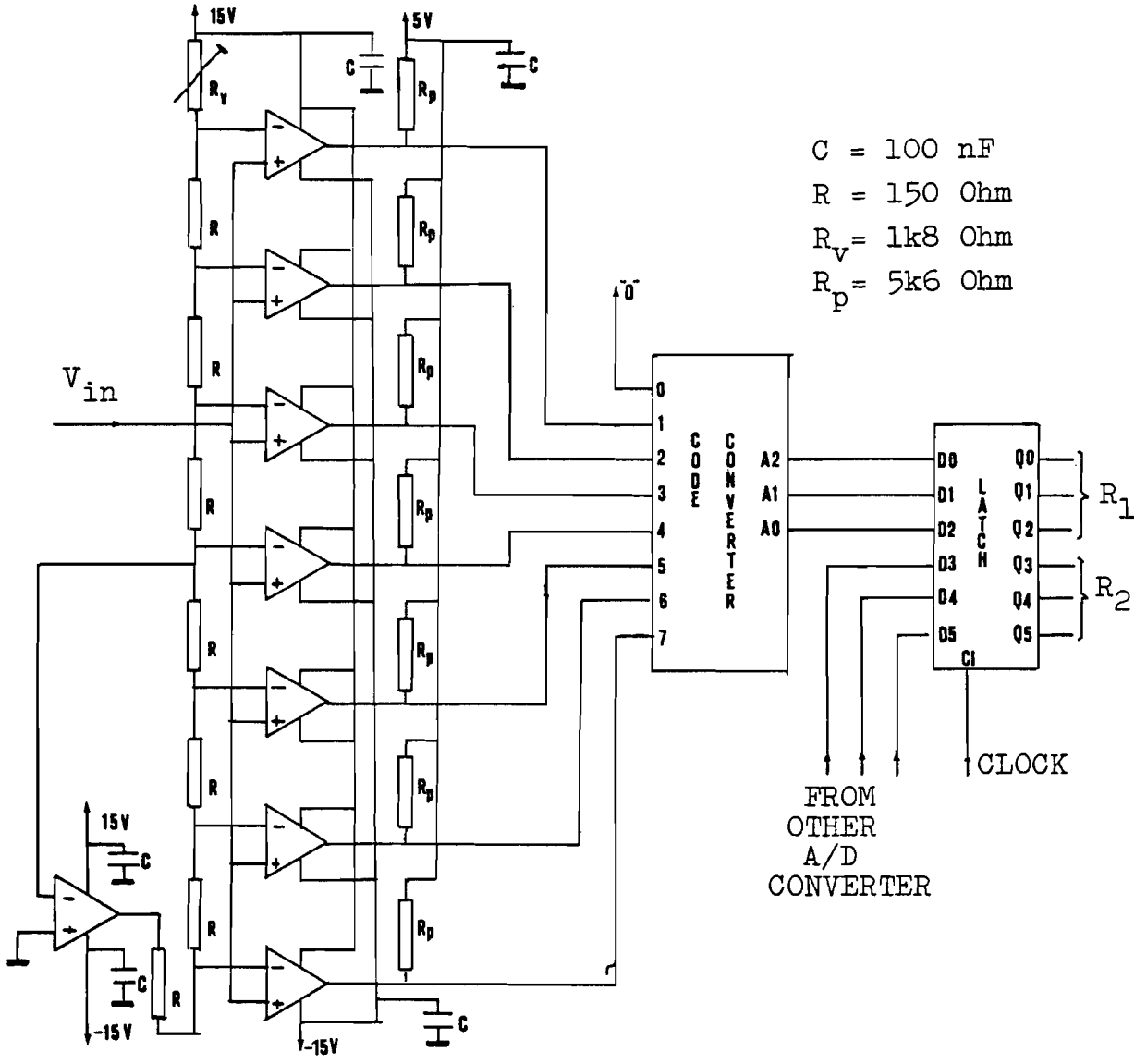
```
 9400    %
 9500    %       FACTORIAL COMPUTATION
 9600    %
 9700            F[0]:=1;
 9800            FOR H:=1 STEP 1 UNTIL 32 DO F[H]:=F[H-1]*H;
 9900    %
10000    %       INPUT
10100    %
10200           FOR H:=1 STEP 1 UNTIL A DO BEGIN
10300           READ(IN,/,T[H],ES[H]);TRANSP;PB[H]:=0 END;
10400    %
10500    %*************************************************************************%
10600    %                                                                         %
10700    %       PROGRAM BODY                                                      %
10800    %                                                                         %
10900    %       THE BODY OF THE PROGRAM COMPUTES:                                 %
11000    %                                                                         %
11100    %       PB := (SUMMATION CK*PK )      ( K FROM 5 TO 32 )                  %
11200    %                                                                         %
11300    %*************************************************************************%
11400    %
11500           FOR K:=5 STEP 1 UNTIL 32 DO BEGIN
11600           FOR H:=1 STEP 1 UNTIL A DO BEGIN
11700           PK[H]:=0;QK[H]:=0 END;N:=0;
11800           IF (N+K) MOD 2=0 THEN SORT;
11900           FOR H:=1 STEP 1 UNTIL A DO PK[H]:=PK[H]+.5*QK[H];
12000           FOR N:=1 STEP 1 UNTIL 7*K DO BEGIN
12100           FOR H:=1 STEP 1 UNTIL A DO QK[H]:=0;
12200           IF (N+K) MOD 2=0 THEN SORT;
12300           FOR H:=1 STEP 1 UNTIL A DO PK[H]:=PK[H]+QK[H]
12400                                       END;
12500           FOR H:=1 STEP 1 UNTIL A DO PB[H]:=PB[H]+(K-4)*(2**(K-5))*PK[H];
12600                                       END;
12700    %
12800    %       OUTPUT
12900    %
13000           FOR H:=1 STEP 1 UNTIL A DO WRITE(OUT,<3L10.3>,T[H],ES[H],PB[H])
13100    END
13200    END.
```

APPENDIX B: THE DESIGN DIAGRAMS

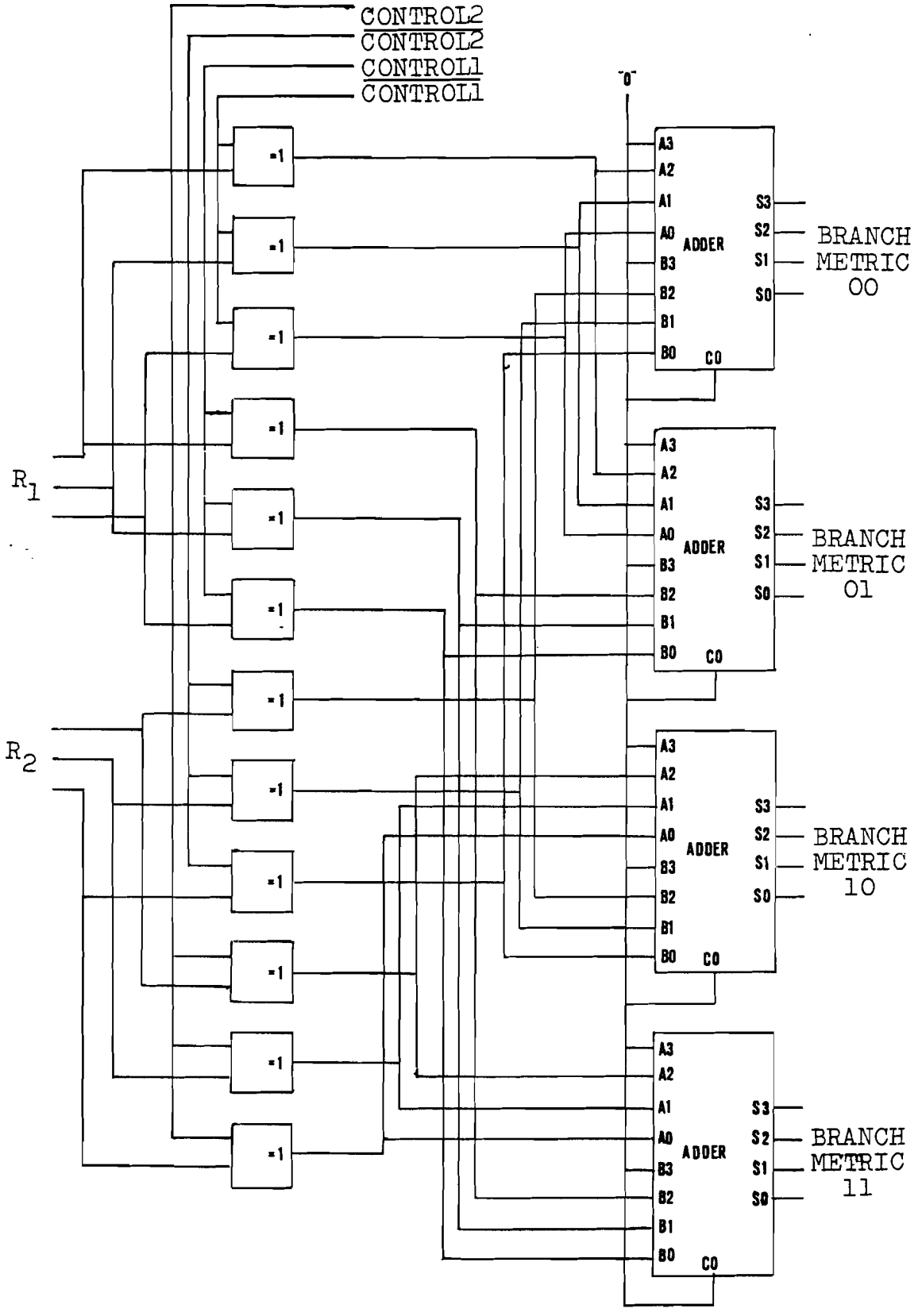
B.1. Schematic diagram

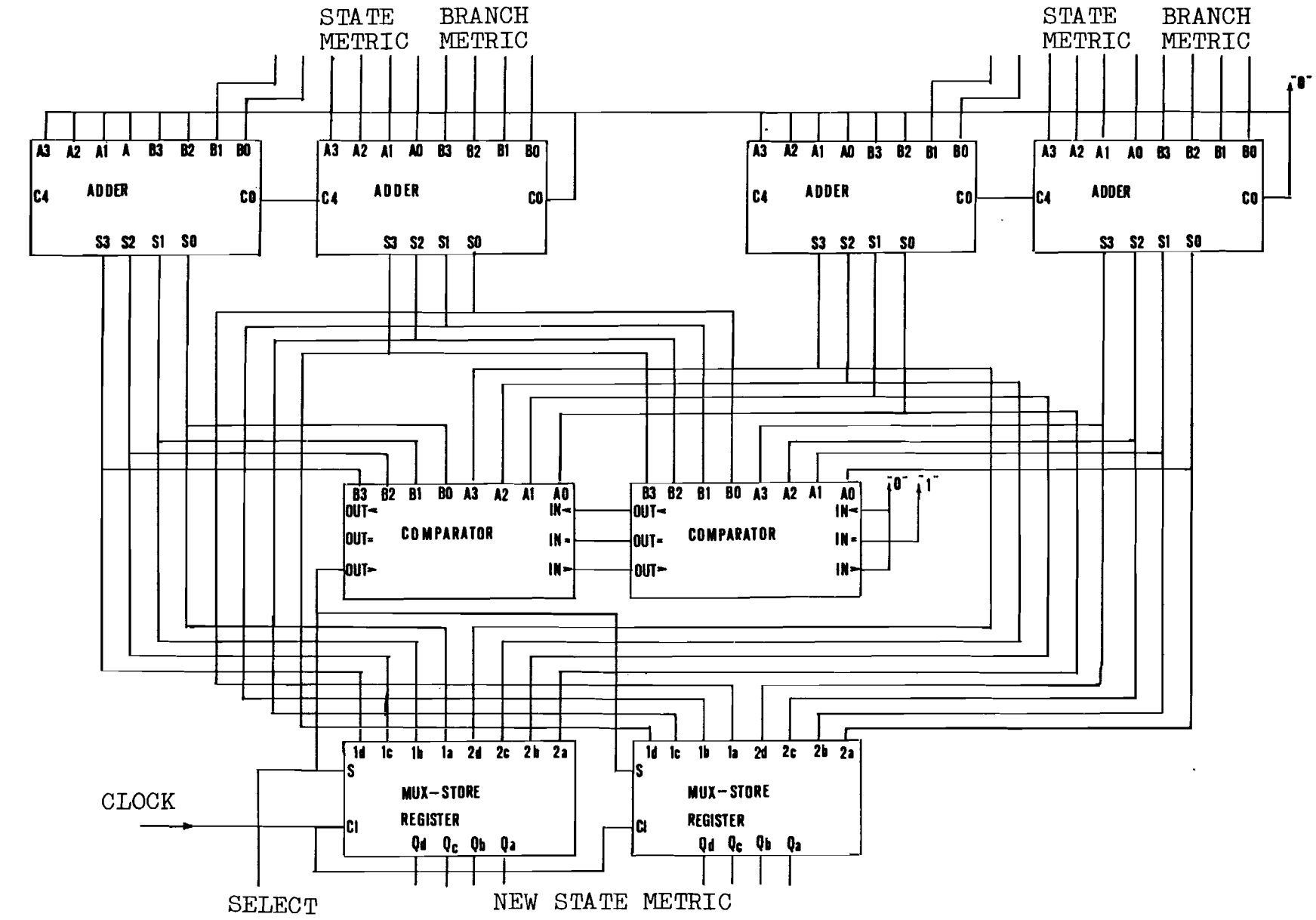


B.2. Input section

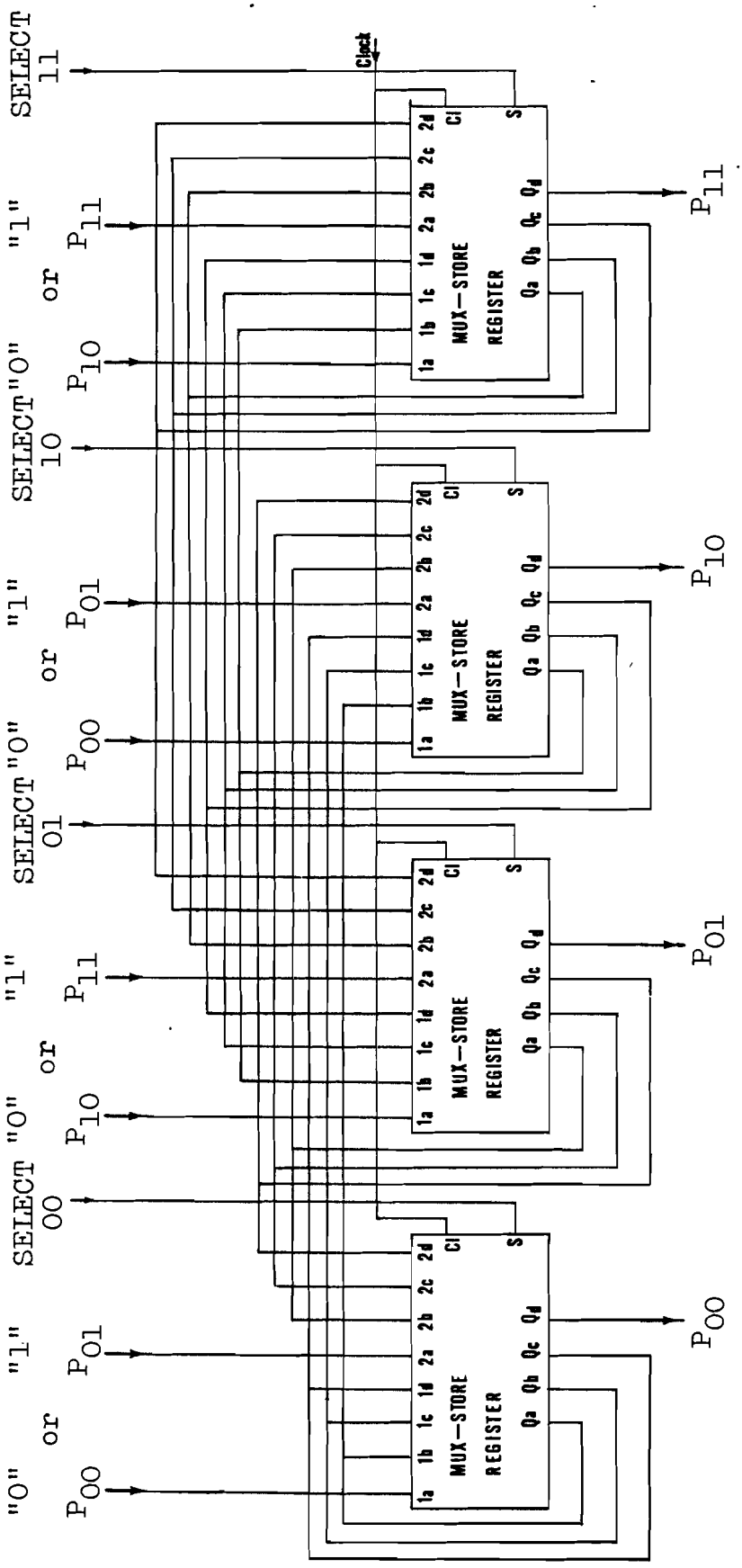


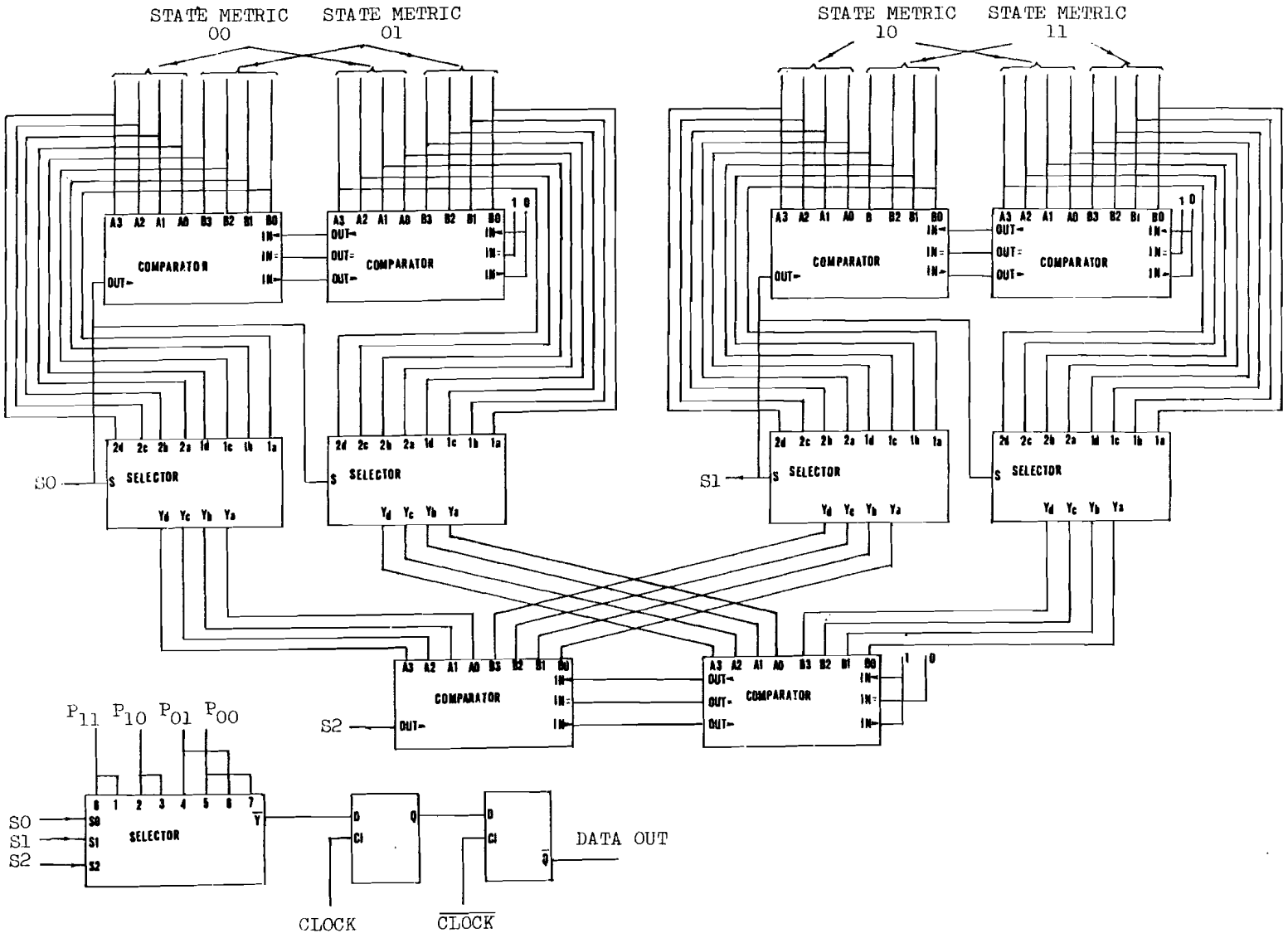
B.3. Branch metric computation section



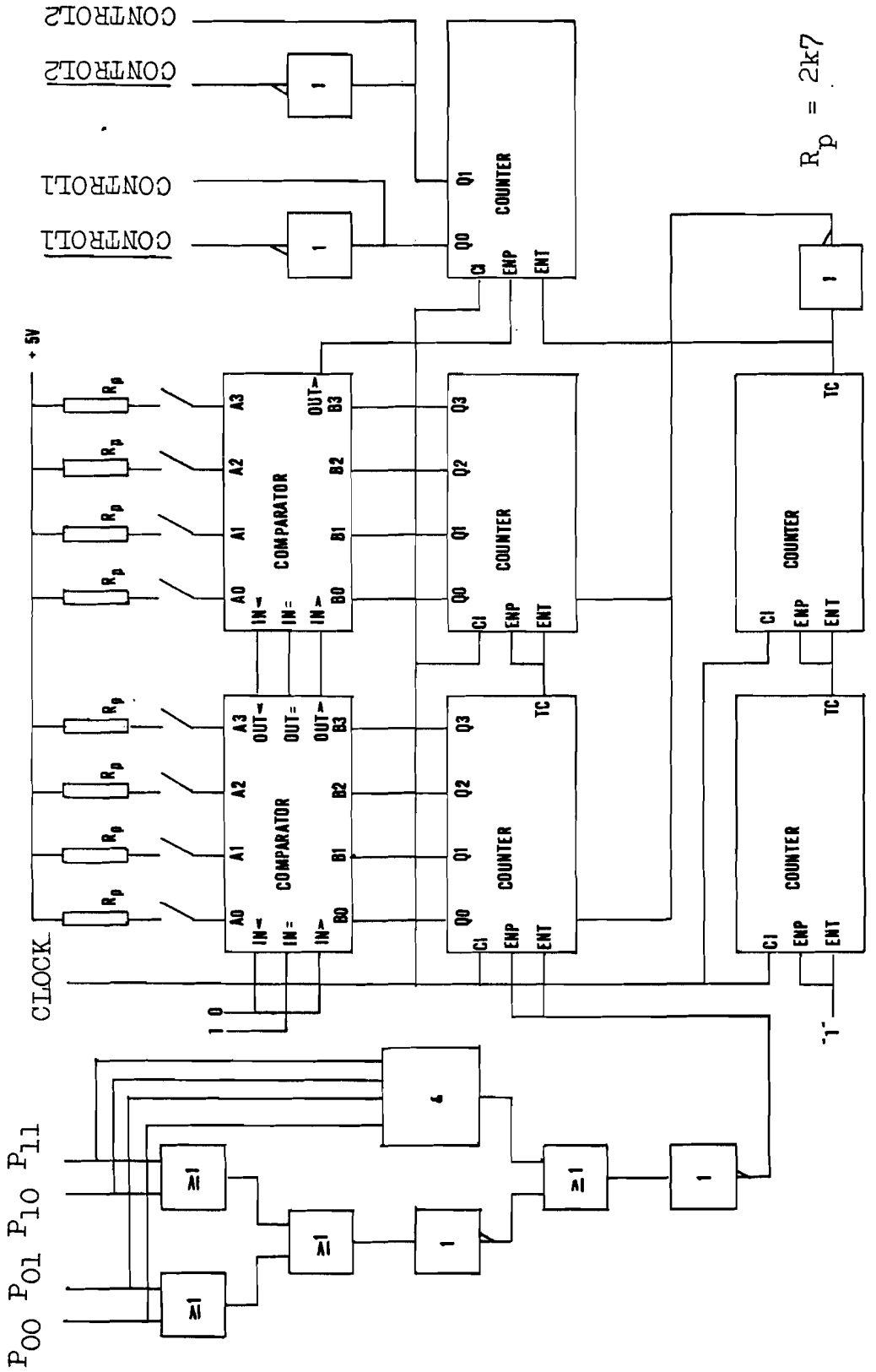

STATE METRIC
BRANCH METRIC
STATE METRIC
BRANCH METRIC
"0"
A3 A2 A1 A B3 B2 B1 B0
C4 ADDER C0
S3 S2 S1 S0
A3 A2 A1 A0 B3 B2 B1 B0
C4 ADDER C0
S3 S2 S1 S0
A3 A2 A1 A0 B3 B2 B1 B0
C4 ADDER C0
S3 S2 S1 S0
A3 A2 A1 A0 B3 B2 B1 B0
C4 ADDER C0
S3 S2 S1 S0
B3 B2 B1 B0 A3 A2 A1 A0
OUT< OUT= OUT> COMPARATOR IN< IN= IN>
B3 B2 B1 B0 A3 A2 A1 A0
OUT< OUT= OUT> COMPARATOR IN< IN= IN>
"0" "1"
1d 1c 1b 1a 2d 2c 2b 2a
S
CI
MUX-STORE REGISTER
Qd Qc Qb Qa
1d 1c 1b 1a 2d 2c 2b 2a
S
CI
MUX-STORE REGISTER
Qd Qc Qb Qa
CLOCK
SELECT
NEW STATE METRIC

B.5. Path memory

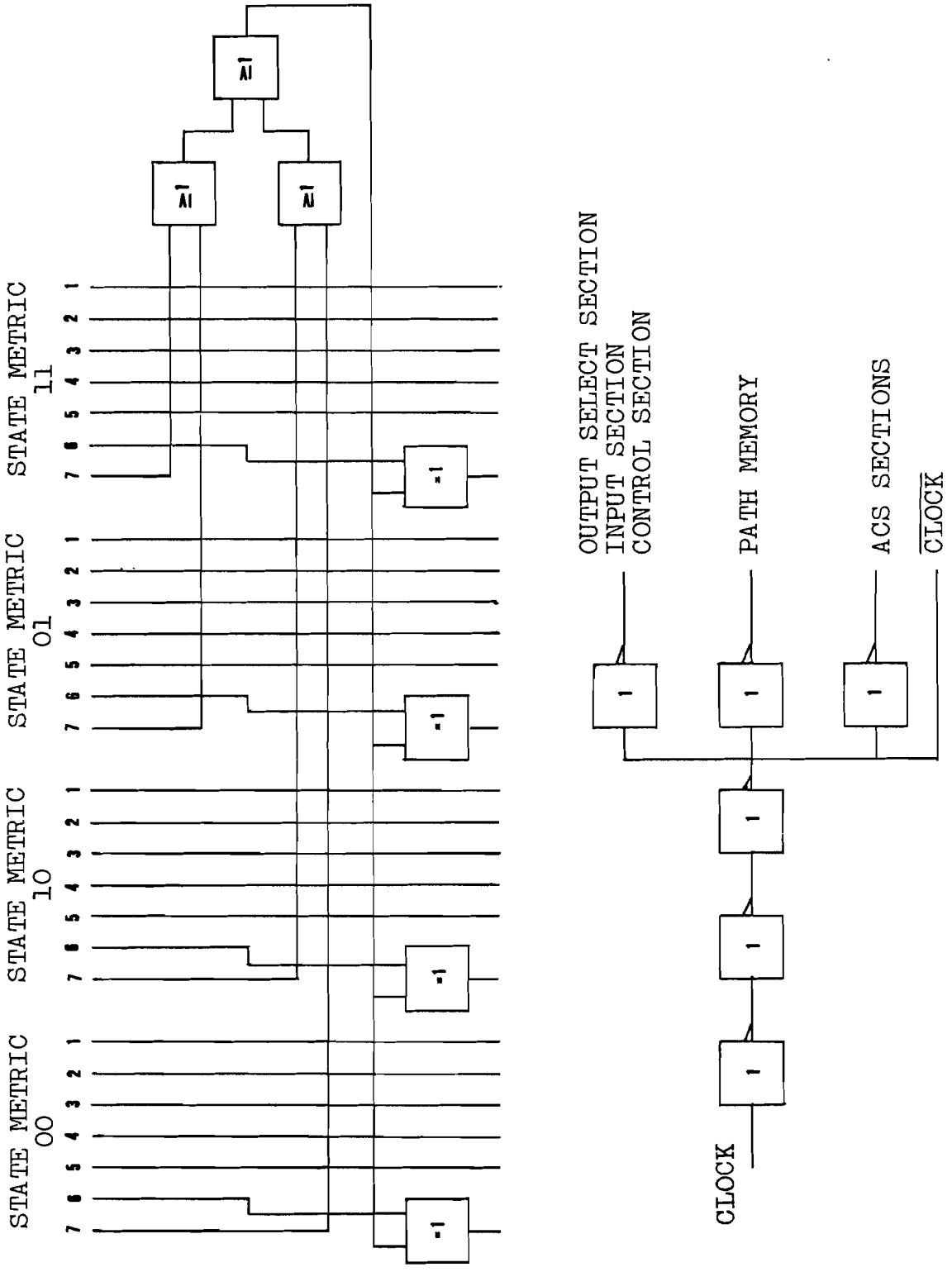



STATE METRIC 00
STATE METRIC 01
STATE METRIC 10
STATE METRIC 11
COMPARATOR
SELECTOR
S0
S1
S2
P_{11} P_{10} P_{01} P_{00}
SELECTOR
S0
S1
S2
CLOCK
$\overline{CLOCK}$
DATA OUT

B.7. Control section



B.8. Normalization and clock sections



B.9. List of Integrated circuits

Decoder

Input section

1	opamp	741
14	comparator	LM319
2	priority encoder	74LS148
1	hex D-flip flop	74LS174

Branch metric computation section

4	adder	74LS283
3	quad exclusive or gates	74LS86

ACS section (four times)

4	adder	74LS283
2	comparator	74LS85
2	multiplex store register	74LS298

Path memory

16	multiplex store register	74LS298

Output select section

4	comparator	74LS84
6	multiplexer	74LS257
1	multiplexer	74LS151
1	dual D-flip flop	74LS74

Control section

5	counter	74LS163
2	comparator	74LS85
1	quad 2-input or gates	74LS32
1	dual 4-input and gates	74LS20
1	hex inverter	74LS04

Normalization section

1	quad 2-input or gates	74LS32
1	quad exclusive or gates	74LS86

Clock section

1	hex inverter	74LS04

Differential decoder

1	dual D-flip flop	74LS74
1	quad exclusive or gates	74LS86

ENCODER

1	shift register	74LS164
1	quad exclusive or gates	74LS86

Differential encoder

1	dual D-flip flop	74LS74
1	quad exclusive or gates	74LS86

APPENDIX C: LIST OF THE MEASURING EQUIPMENT

1 PULSE GENERATOR 8015A
HEWLETT PACKARD

1 DATA GENERATOR 3760A
HEWLETT PACKARD

1 ERROR DETECTOR 3761A
HEWLETT PACKARD

2 NOISE GENERATOR 1390-B
GENERAL RADIO COMPANY

2 POWER COMBINER

1 OSCILLOSCOPE 547
TEKTRONIC

1 SPECTRUM ANALYZER 8566A
HEWLETT PACKARD

1 ELECTRONIC COUNTER 5216A
HEWLETT PACKARD

1 POWER SUPPLY UNIT D-16
INDAL HOLLAND

1 POWER SUPPLY UNIT E 18-0.6D
DELTA ELEKTRONIKA

1 POWER SUPPLY UNIT D 015-1.5
DELTA ELEKTRONIKA

ATTENUATORS
HP 355C
HP 355D

6 Amplifier