MICROPROCESSOR-CONTROLLED SOFT-DECISION DECODING OF ERROR-CORRECTING BLOCK CODES.

R.M.F. Goodman* B.Sc., Ph.D.

A.D. Green*, B.Sc.

Summary:
Soft-decision decoding offers a means of bridging the performance gap between a block error-control system that uses hard-decision bounded-distance decoding, and one that uses maximum-likelihood decoding. Existing algorithms, however, tend to be complex in terms of hardware requirement. This paper presents a new class of soft-decision decoding algorithms, based on error-trapping decoding, which have a simpler hardware requirement in exchange for coding gain and decoding delay trade-offs. In addition, a microprocessor-controlled implementation of these algorithms is described, and decoder performance and trade-offs under Gaussian noise conditions are investigated.

1. Introduction:

A soft-decision binary block decoding scheme is one in which the demodulator assigns a 'confidence' value to each output bit, in addition to the 'hard' binary 0 or 1 decision. In practice this involves replacing the one bit hard decision device at the demodulator output with a J bit analogue-to-digital converter. The confidence information can then be used to improve the decoding process in such a way that the probability of decoding error, or the decoding delay, is reduced.

The increase in coding gain (over that achievable with hard-decision) that can be expected by using soft-decision decoding depends on a number of factors. These include the number and spacing of the quantisation levels, the decoding algorithm used, and the channel characteristics. It has, however, been shown (ref.1) that for the additive white Gaussian channel with optimum-spacing infinite-level quantisation the maximum coding gain is lower-bounded by about 2dB. Fortunately, the degradation involved in using the much more practical equal-spacing, 8-level quantisation is only about 0.1dB, and even optimum-spacing 3-level quantisation only degrades by about 1dB. In addition, it has been shown (ref.2) that for both the Gaussian and Rayleigh-fading channels the performance of a soft-decision decoder approaches that of the optimum maximum-likelihood decoder. In the case of the Gaussian channel this implies a coding gain improvement over hard-decision of 3dB at high signal-to-noise ratios.

The main objection to the use of soft-decision block decoding algorithms has been one of hardware complexity. This is because in addition to having to handle J bits instead of one bit, existing algorithms assume that for the particular code in use, a bounded-distance hard-decision decoder capable of correcting errors (or errors-and-erasures) is already available. Also, most existing algorithms operate iteratively and this raises the question of decoding delay as well as implying complex control and computation hardware. The algorithms proposed in this paper, however, are designed to work with a syndrome generator, which is a much simpler item of hardware to implement than a full hard-decision decoder for a multiple-error-correcting code. Two algorithms are proposed, one of which operates

*Department of Electronic Engineering, University of Hull

iteratively, and one which is non-iterative. For the Gaussian channel, the latter case implies a sacrifice of coding gain for speed of decoding, whilst the iterative algorithm enables a performance approaching that of maximum-likelihood decoding to be achieved, but at the price of increased decoding delay. The algorithms also have an inherent burst-error-correction capability. This means that the amount of coding gain sacrificed will be worst under Gaussian noise conditions, but much smaller for a diffuse burst-noise channel such as the H.F. radio channel.

The algorithms were designed for use on a narrowband H.F. link operating at a maximum throughput rate of 1 Kbit/sec. At this relatively slow speed a mixed hardware/software implementation is feasible, and the problem of providing a comprehensive control and computation facility was solved by using a microprocessor. The system as implemented uses a standard Intel 8080A microprocessor to control a fast hardware syndrome generator. The flexibility of stored program control not only enables different variations of the algorithms to be easily implemented on the system by simply changing program P.R.O.M.s, but also means that the system is not dedicated to one particular code, thereby enabling channel-adaptive algorithms (ref.3) to be easily implemented if required.

This paper develops in the following way. Firstly, the basic concepts of soft-decision decoding are outlined. Hard-decision error-trapping decoding is next reviewed, and then the soft-decision algorithms are introduced. Finally, the microprocessor implementation of the algorithms is described, and system performance results under Gaussian noise conditions are presented and discussed.

2. Soft-Decision Decoding:

Given an error-control system operating with an (n,k) binary block code, let us assume that the soft-decision demodulator quantises each output digit y_i ($1 \leq i \leq n$) to $2^J = Q$ levels, so that the estimate of the received binary digit is given by the soft-decision J bit byte: $[y_i] = [y_1 y_2 .. y_J]_i$. Under noise-free conditions $[y_i]$ is assumed to be either all-zero or all-ones depending on whether 0 or 1 was transmitted. The first bit of $[y_i]$ is the hard-decision estimate, and the remaining J-1 bits give an indication of the confidence of that estimate. The confidence of the hard-decision may be defined as the J-1 bit byte:

$[c_i] = [c_1 c_2 .. c_{J-1}]_i$, where $[c_i] = [c_1 c_2 .. c_{J-1}]_i = [y_2 y_3 .. y_J]_i$ if $y_1 = 1$, and $[c_i] = [c_1 c_2 .. c_{J-1}]_i = [y_2 y_3 .. y_J]_i \oplus [111..1_{J-1}]$ if $y_1 = 0$.

Thus the confidence of a particular digit can vary from $[c_i] = [000..0_{J-1}]$ (least confident, nearest to the hard-decision 0/1 boundary), to $[c_i] = [111..1_{J-1}]$ (furthest away from the boundary). Alternatively, we may consider that the demodulator output soft-decision digit $[y_i]$ gives an estimate of the soft-decision noise digit $[n_i]$ which has been added to the transmitted digit $[x_i]$ by the channel. Thus $[n_i] = [n_1 n_2 .. n_J]_i = [y_1 y_2 .. y_J]_i$ if $[x_i] = [000..0_J]$, or $[n_i] = [y_1 y_2 .. y_J]_i \oplus [111..1_J]$ if $[x_i] = [111..1_J]$. The value of the noise digit in levels can therefore lie between 0 and (Q-1), and a noise value of $\geq (Q/2)$ causes a hard-decision error.

We may now form an estimate of the error-correcting capability of an (n,k) block code in the soft-decision sense. If the minimum distance of the code is d_h then in the soft-decision sense codewords are $\geq d_s = (Q-1) \times d_h$ soft-decision levels apart, and therefore the bounded-distance guaranteed soft-decision error-correction capability of the code is the largest integer satisfying $t_s \leq [(d_s-1)/2]$. In order to see the improvement obtained from soft-decision bounded-distance decoding, consider the (15,7) $d_h = 5$ cyclic code, and assume Q = 8 level quantisation. The hard-decision bounded distance error-correction capability of this code is $t_h \leq [(d_h - 1)/2] = 2$ errors per block. For soft-decision $d_s = 35$, and therefore correction can be guaranteed for a total of 17 or fewer level errors per block. The smallest number of level errors that constitute a hard-decision bit error is Q/2 = 4; the most probable pattern of 4 hard-decision errors therefore has a soft-decision weight of only 4 x 4 = 16 levels which is within the soft-decision bounded-distance capability of the code, and can therefore be corrected. Thus at high signal-to-noise ratios where the probability of >17 level errors per block is extremely small, the (15,7) code behaves as if it had an error-correction capability of 4 (hard) errors. In general, at high signal-to-noise ratios, soft-decision decoding of an (n,k) code enables up to (d_h-1) errors to be corrected.

Given an (n,k) block code the optimum method of decoding is maximum-likelihood decoding, which for the Gaussian channel is equivalent to minimum-distance decoding. A minimum-distance hard-decision decoder attempts to find the codeword x_m nearest in terms of Hamming distance to the received word y. That is, the codeword x which satisfies

$$\min \left[\sum_{i=1}^{n} (y_i \oplus x_i)\right]$$

Similarly, a soft-decision minimum-distance decoder attempts to find the codeword at minimum soft-decision distance (minimum number of level errors) from the received word. That is, the codeword x which satisfies

$$\min \left[\sum_{i=1}^{n} [y_i] \oplus [x_i]\right].$$

In both the hard-decision and the soft-decision cases sub-optimum bounded-distance decoding can be implemented with a consequent loss in coding gain. In this case the decoder tries to find a codeword within the bounded-distance correction radius (hard or soft) of code, and if no such codeword can be found the decoder refuses to decode. The advantage of implementing bounded-distance decoding is that it may enable the decoding time of an iterative decoding algorithm to be reduced.

The obvious way of implementing soft-decision minimum-distance decoding would be to compute the soft distance between the received word and all 2^k codewords, and then choose the codeword at minimum soft distance from the received word. This is clearly impractical if k is at all large. Two more practical classes of algorithm that have been proposed are successive-erasure decoding, and error pattern testing.

Successive erasure decoding (ref.4) assumes the existance of a decoder which is capable of correcting both erasures and errors. The method is iterative and involves successively erasing (i.e. treating as an erasure) the least reliable digits in the hard-decision estimate of the received

word, and at each step decoding the altered word with the errors and erasures decoder. A set of tentative codewords is thus produced and the codeword at minimum soft distance from the received word is chosen as the estimate of the transmitted codeword. The number of decoding trials is $(d_h-1)/2$ which is not excessive, but coding gain is lost because of the effective three level quantisation. Also, the errors-and-erasures decoder is in practice difficult to implement.

Error pattern testing (ref.2) assumes the existance of a hard-decision minimum-distance or bounded-distance decoder. The method is iterative and involves perturbing the hard-decision estimate of the received word with a set of locally generated test error patterns and decoding the perturbed sequence with the hard-decision decoder. The set of tentative codewords produced is then tested against the received word, and the codeword at minimum soft distance from the received word is chosen as the estimate of the transmitted codeword. A performance approaching that of maximum-likelihood decoding is achievable on the Gaussian channel if all error patterns of weight d-1 or less are tested, but this involves an excessive number of trials.

The algorithms presented in this paper assume only the existance of a syndrome generator and are consequently less complex to implement than the above algorithms. Sub-optimum decoding is possible in one trial, and the coding gain lost depends on the burstyness of the channel. The maximum loss in coding gain is for the Gaussian channel. Optimum decoding can be achieved by using the iterative algorithm but the number of trials can be excessive if the channel is not bursty. It is also possible to trade coding gain for a reduction in the number of trials when using the iterative algorithm, and this trade-off is examined in section 6.

3. Hard-Decision Error-Trapping Decoding:

Given a t-error correcting (n,k) cyclic code with generator polynomial g(x), the syndrome s(x) of a hard-decision received word h(x) can be written

$$s(x) = \text{remainder } [\frac{h(x)}{g(x)}], \text{ and depends only on the}$$

error pattern e(x), and not on the actual codeword transmitted. We may thus write $e(x) = q(x)\ g(x) + s(x)$. If the errors in e(x) are all confined to the n-k parity check positions of h(x), that is $x^{n-k-1},..,x,1$, then q(x) = 0 and e(x) = s(x) and the error pattern is identical to the syndrome. If the errors are not confined to the n-k parity position of h(x) but are confined to any n-k consecutive positions (including the cyclic end-around case) then these errors appear in the n-k parity positions of some other word h'(x) which is a cyclic permutation of h(x). The syndrome of h'(x) thus equals the original error pattern cyclicly shifted. Fig 2 shows the syndrome generator for an (n,k) cyclic code, which can compute the syndrome of a received word in n shifts, and because of the cyclic property of the syndrome, can compute the syndrome of a i-place cyclic shift of h(x) in a further i shifts. The operation of the generator can be explained as follows. Firstly, feedback is suppressed by the inhibit gate, and the (n-k) stage register is loaded in (n-k) shifts. Feedback is then allowed, and after a further k shifts the syndrome of h(x) is in the register. If the weight of the syndrome is $\leq t$ it is assumed that e(x) = s(x). If the weight of the syndrome is >t the generator starts to shift again with the input suppressed and feedback applied. After i shifts the syndrome of an i-place cyclic shift of the received word is contained in the register, and can be tested for a syndrome weight of $\leq t$, which would indicate that the error pattern has been trapped. If, after n shifts the syndrome weight never goes down to t

or less then either (1) a detectable but uncorrectable error pattern has occurred, or (2) the error pattern is not trappable in (n-k) consecutive positions.

Provided that most error patterns of weight t or less can be trapped in (n-k) consecutive positions for a given code then error trapping is an efficient and simple means of decoding. If not, much of the power of the code is wasted. However, error trapping is useful in providing a simple sub-optimum burst-and-random decoding method. In this case the decoder tries to find a syndrome of weight $\leq t$ but also notes the shift which gives the syndrome of shortest burst length. If, after n shifts the syndrome weight does not go down to $\leq t$, the decoder chooses the shortest burst as the error pattern.

4. Soft-Decision Error-Trapping Decoding:

4.1 Algorithm 1.

This algorithm performs non-iterative sub-optimum minimum-distance decoding, and the steps are as follows:

1. Calculate the syndrome s(x) of h(x) using the syndrome generator.
2. Assuming that the tentative error pattern is given by $e(x)=[000..0_k s(x)]$, calculate the soft decision weight of the error pattern.
3. Shift the syndrome generator once to calculate s'(x) and repeat 2.
4. After n shifts choose the tentative error pattern that gives the lowest soft-decision weight as the actual error pattern.

The above operation can be referred to as one 'trial' and is complete in k + n syndrome generator shifts.

In general, the soft weight of an error pattern is given by:

$$\sum_{i=1}^{n} [y_i(x)] \oplus [e_i(x)] \oplus [h_i(x)]$$

where $h_i(x)$ is the hard-decision estimate of $y_i(x)$, and this implies that n additions of J bit soft-decision numbers are required at each shift. In fact, significantly fewer additions are required by implementing the following method. Firstly, let us define the basic soft error (B.S.E.) of the received word y(x) as the soft distance between y(x) and h(x).

Thus $(B.S.E.) = \sum_{i=1}^{n} [\bar{c}_i]$, that is, confidence complemented. The soft weight of e(x) is then given by:

$$|e(x)|_s = (B.S.E.) + \sum_i [(2 \times c_i) + 1]$$

$$= (B.S.E.) + \sum_i [2 \times c_i] + (\text{syndrome weight})$$

where i only takes the values corresponding to 1's in s(x). As the B.S.E. is a constant, it need not enter into the minimum-distance computation, and therefore for algorithm 1 the number of additions performed at each shift is only equal to the number of non-zero bits in s(x), plus one. Also note that $2 \times c_i$ is simply obtained by a single shift of c_i.

The increased coding gain (over hard decision) achievable with algorithm 1 depends on whether or not the most probable error patterns of weight $\leq (d-1)$ are trappable. This is increasingly true the more bursty the channel, so that worst case performance occurs for the Gaussian channel.

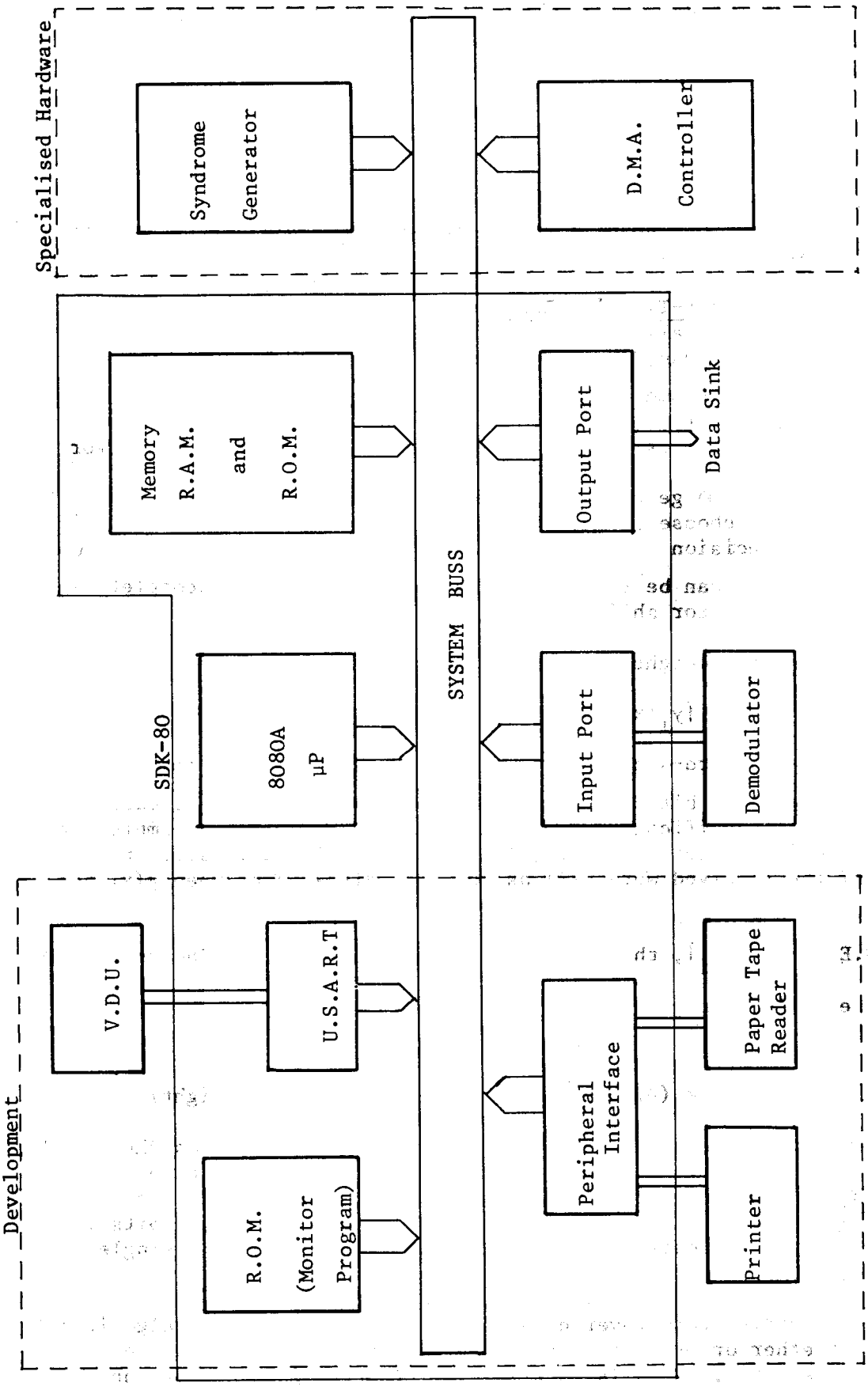


Fig. 1 Microprocessor Decoding System

4.2 Algorithm 2 In order to achieve reasonable soft-decision coding gains when a significant proportion of the most probable error patterns of weight $\leq(d-1)$ are not trappable, as for example with high rate codes on the Gaussian channel, it is necessary to use algorithm 2 which is iterative, and can either take a bounded-distance or minimum distance form. The steps in the bounded-distance form are as follows:

1. Execute one decoding trial, as with algorithm 1, and if a syndrome is found which gives a tentative error pattern whose soft weight $\leq t_s$, accept the syndrome as the error pattern.
2. If no such syndrome is found another decoding trial is initiated. Invert the hard decision estimate of the least confident digit in the received word and assume maximum confidence for this digit.
3. Decode this altered hard-decision estimate, testing for a syndrome that gives an error pattern whose soft weight is $\leq(t_s - \Sigma_i |[y_i] \oplus [\bar{h}_i]|_s)$ where the summation is over all inverted bits (in this case 1). If such a syndrome is found the syndrome plus the inverted bit equals the error pattern.
4. If no such syndrome is found, further decoding trials are executed, with decreasingly probable single or multiple bit inversion patterns being used at each trial.
5. If no such syndrome is found by the time all trials have been executed, the decoder chooses the minimum soft weight error pattern encountered in the trials.

The number of trials required to decode a given block thus depends on the actual error pattern which occurred and is upper bounded by

$\Sigma_i \binom{n}{i}$ for $0 \leq i \leq d - 1 - t_t$, where t_t is the guaranteed error-trapping correction power of the code. However, increasing numbers of trials only occur with decreasing probability, so that on average a maximum of only a few trials per block are in general needed. The number of trials required by algorithm 2 can be reduced (at the expense of coding gain on the Gaussian channel) by imposing constraints on the maximum number of trials. For example, only single bit inversions may be tried. In this way the number of trials can be reduced without much loss in coding gain if the channel exhibits both burst and random-error characteristics.

5. Microprocessor Implementation:

5.1 System Hardware. The decoding system is based on an Intel 8080A eight-bit microprocessor (μP), and uses the commercially available standard SDK-80 microprocessor development kit, with 1K of R.A.M. The system block diagram is shown in Fig.1, and outlines the parallel buss structure of the SDK-80 microcomputer. Each block is connected to the system buss which consists of 8 data lines, 16 address lines, and 6 control lines through which all the internal system signals pass. The system falls into three basic sections as outlined below.

Firstly, the development section allows the input and editing of programs, and the output of performance statistics. Thus algorithms 1 and 2, and any variation of these, can be implemented by designing the suitable software. The second section consists of the basic microcomputer elements : μP, memory, and input/output ports. The third section contains specialised hardware and consists of two units: a fast syndrome generator and the direct memory access (D.M.A.) controller.

The syndrome generator is shown in Fig. 3, and is of the type shown in Fig. 2, with the added refinement that syndrome weights are calculated.

Fig. 2 Syndrome Generator

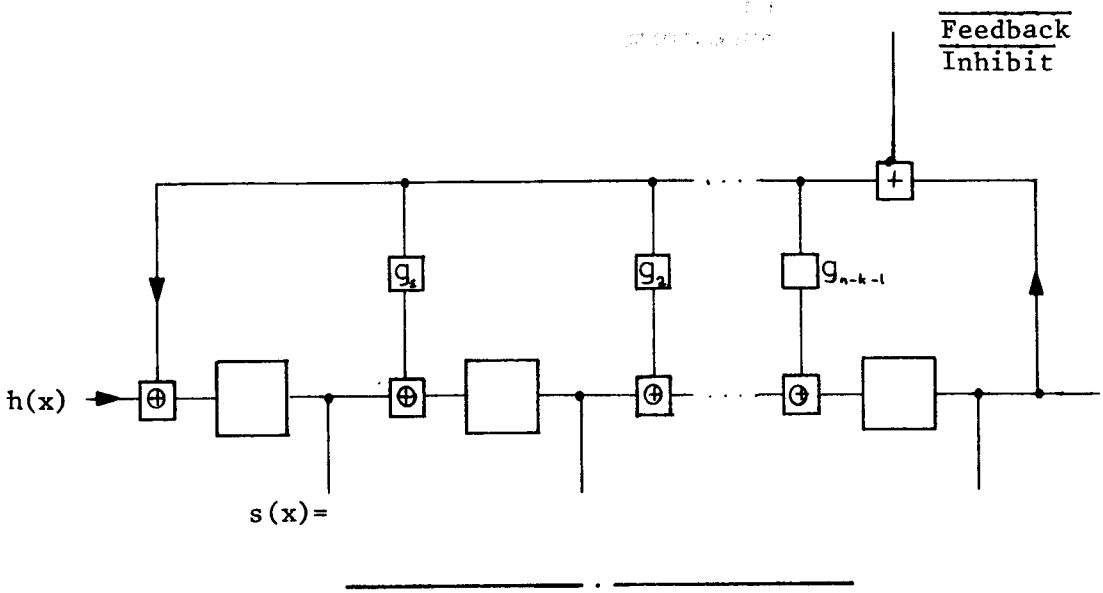
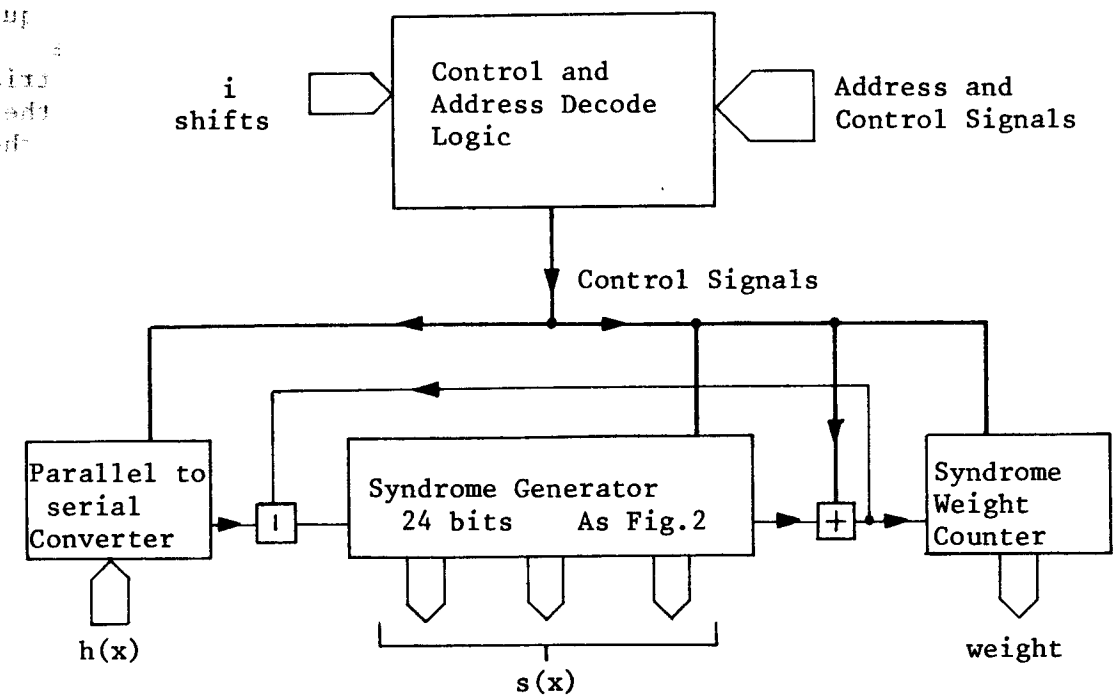


Fig. 3 System Syndrome Generator



In addition, the syndrome generator is generalised, that is, not dedicated to a particular code, and can be set up by the μP to operate in any code that has $n < 256$, and $(n-k) < 24$. The syndrome generator is provided with input and output ports which are organised as locations of memory, thus enabling the μP to read or write directly to it. The input port consists of two 8-bit registers. One is for the hard-decision estimate of the received block which is input in groups of 8 bits to this register, and then automatically shifted into the syndrome generator. The second register is used when an existing syndrome is to be shifted i places with feedback operating, and the shifts are automatically initialised on receipt of the value of i in this register. The output port consists of four 8 bit registers, one of which contains the syndrome weight while the other three contain the syndrome, which can be up to 24 bits long. The syndrome generator operates at a serial clock rate of 9.216MHz, thereby enabling data to be input at a rate of 1 byte per μs, which is as fast as the R.A.M. can be accessed, and is fast enough to make iterative algorithms a feasible proposition. Thus as far as the software is concerned syndromes are quickly calculated by simply writing to the appropriate memory location. The time taken to calculate the syndrome and syndrome weight from receipt of the last byte of input data is in the worst case equivalent to one machine instruction which means that as far as the μP is concerned the syndrome is available 'instantly'.

The D.M.A. controller enables the direct transfer of data from the input port to memory, and of corrected message digits from memory to the output port. The system will operate with or without D.M.A. control, but in the latter case the μP operates under interrupt control and channel throughput rates are reduced.

5.2 Storage Format.

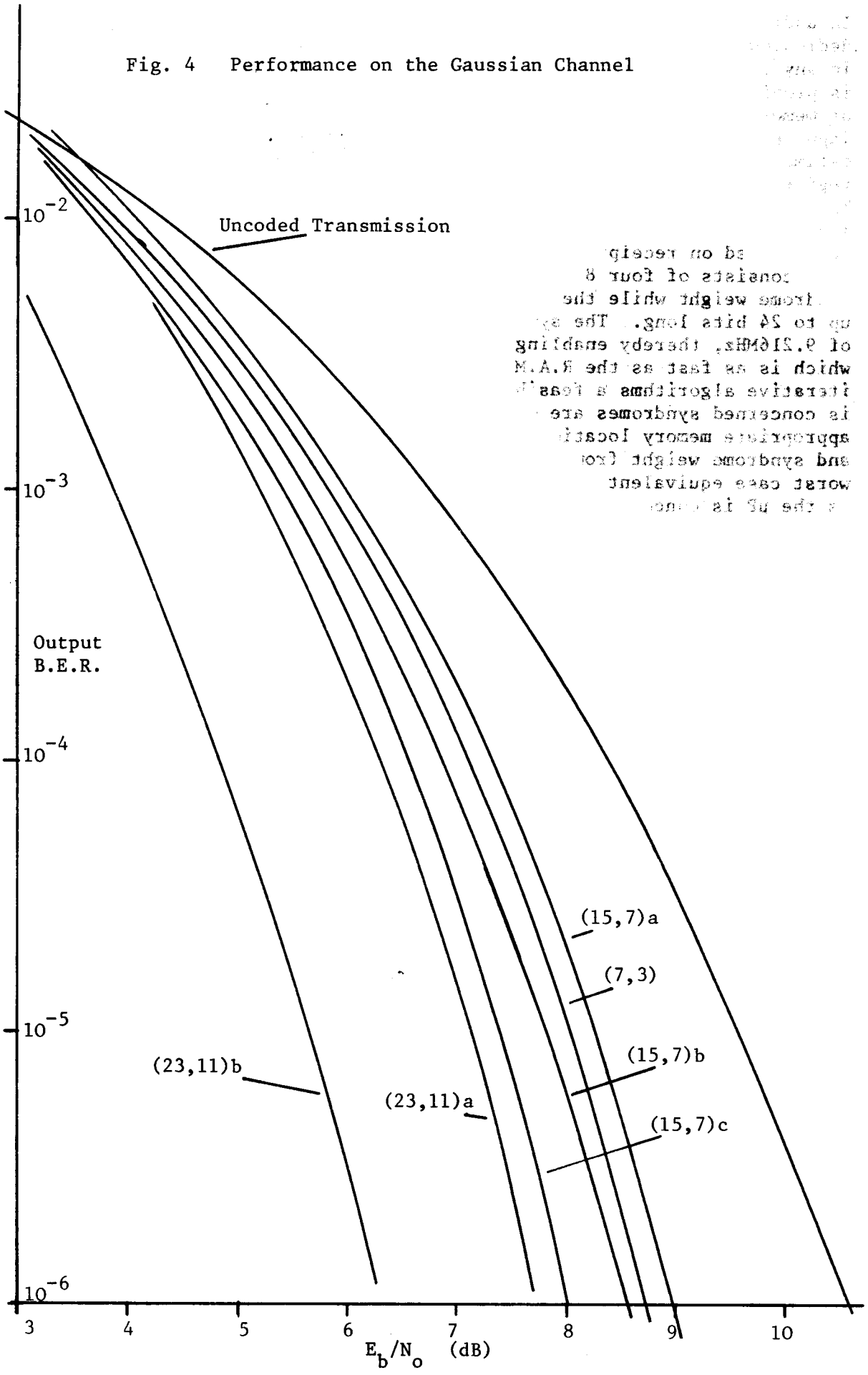
The demodulator outputs a parallel 4-bit quantised estimate of the corresponding channel digit, which takes values between 0000 and 1111. A simple hardware converter enables the estimate to be sent to the input port in one of three formats, depending on which one minimises the software execution time for the particular algorithm in use. These are: straight quantised binary; most significant digit as hard-decision, plus three bits confidence; and hard-decision plus confidence-complemented. These input soft-decision digits are then stored in memory and organised as two full (including hard decision) values per 8 bit byte, and 8 hard-decision bits per byte. Thus 8 channel-digit estimates require 5 bytes of storage. The storage duplication for hard-decision bits is designed to reduce software execution time, as the need to separate hard from soft information is eliminated.

The storage area allocated to the input data depends on the amount of R.A.M. available, which can be easily expanded to 64K. The storage area is used in a cyclic order to provide an automatic buffer capability. The hard and soft data are organised in separate streams in such a way that a two place right shift of the eight least significant bits of the soft-decision byte address gives the address of the corresponding hard-decision byte. Programs can be developed in R.A.M., without having any effect other than reducing the data buffer area, and later transferred to P.R.O.M., for program continuity during power-off.

5.3 System Operation and Data Movements.

The organisation of internal data movements depends on whether or not D.M.A. is being used. Without D.M.A., input data transfers operate under interrupt control, the demodulator causing an interrupt signal

Fig. 4 Performance on the Gaussian Channel



when one (machine) byte of soft-decision data is ready. In this case the μP, which may be executing the decoding algorithm on a particular block, must as a matter of priority service this interrupt before proceeding with program execution. Input data is then moved to the accumulator and then into memory, after which the μP can continue. Similarly, output transfers move data from memory to accumulator to output port, and are made as a matter of priority if the input buffer storage area is in danger of overflowing. This may entail suspending or terminating the decoding of a difficult-to-decode block, in order to prevent buffer overflow. In general, the system can operate without D.M.A. at the required 1K bits/sec with no danger of overflow if relatively short codes ($n < 24$) and non-iterative algorithms are used.

When operating with D.M.A. the demodulator signals the D.M.A. controller that a byte of data is ready, at which point the μP is held and the D.M.A. controller takes control of the system buss. The transfer of data is then made, and the μP is released. Input transfers made in this way take about 900 ns, but this does not mean that the μP is actually held for this length of time. By implementing cycle-stealing, data transfers are synchronised with a part of the machine cycle during which the μP does not require the use of the system buss. Data transfers made in this way will hold the μP for a maximum of 1 machine cycle (488ns) and may often not cause any loss of μP time. With D.M.A. any of the codes which the syndrome generator can handle are decodable at a channel bit rate of 1K bits/sec, although with long codes iteratively-decoded at high error rates, buffer overflow may occur if only 1K of R.A.M. is used.

6. Performance Results:

The performance of the algorithms and microprocessor system has been investigated for Gaussian noise conditions, as this represents the worst case. Binary antipodal signalling with equal-spacing 16-level quantisation and matched filter detection is assumed.

If the (single-sided) noise power density is given by N_o, the signal-to-noise ratio for the Gaussian channel is given by $\gamma = E/N_o$, and the bit probability of error is given by $p = Q(\sqrt{2\gamma_b R})$, where $\gamma_b = E_b/N_o = \gamma/R$ is the normalised signal-to-noise ratio per information bit, and R is the inverse of the bandwidth expansion (code rate). Note that all performance curves are plotted versus E_b/N_o to ensure a valid comparison between different coded systems, and between coded and uncoded transmission.

Three different codes are considered, these are the (7,3) $d_h = 4$ code, the (15,7) $d_h = 5$ code, and the (23,11) $d_h = 8$ code. Fig. 4 plots output bit error rate (B.E.R.) versus E_b/N_o. The curve for the (7,3) code shows a coding gain of 1.6 dB over uncoded transmission at an output BER of 10^{-5}, and this is only 0.3 dB less than that achievable with maximum likelihood decoding (MLD). Algorithm 1 is used in this case, showing that most patterns of $(d-1) = 3$ errors or less are trappable with this code. In general, the probability of output block error (P_{BLK}) for algorithm 1 can be overbounded by

$$P_{BLK} \leq P_{MLD} + \text{Prob [non-trappable error pattern]}.$$

Three curves are given for the (15,7) code, Curve (15,7)a shows the performance of algorithm 1. In this case coding gain is being sacrificed because all patterns of a given weight(w) are only trappable for $w \leq t_t = 2 < (d - 1) = 4$. Curve (15,7)b shows the performance of

Fig. 5 Soft Coding Gain

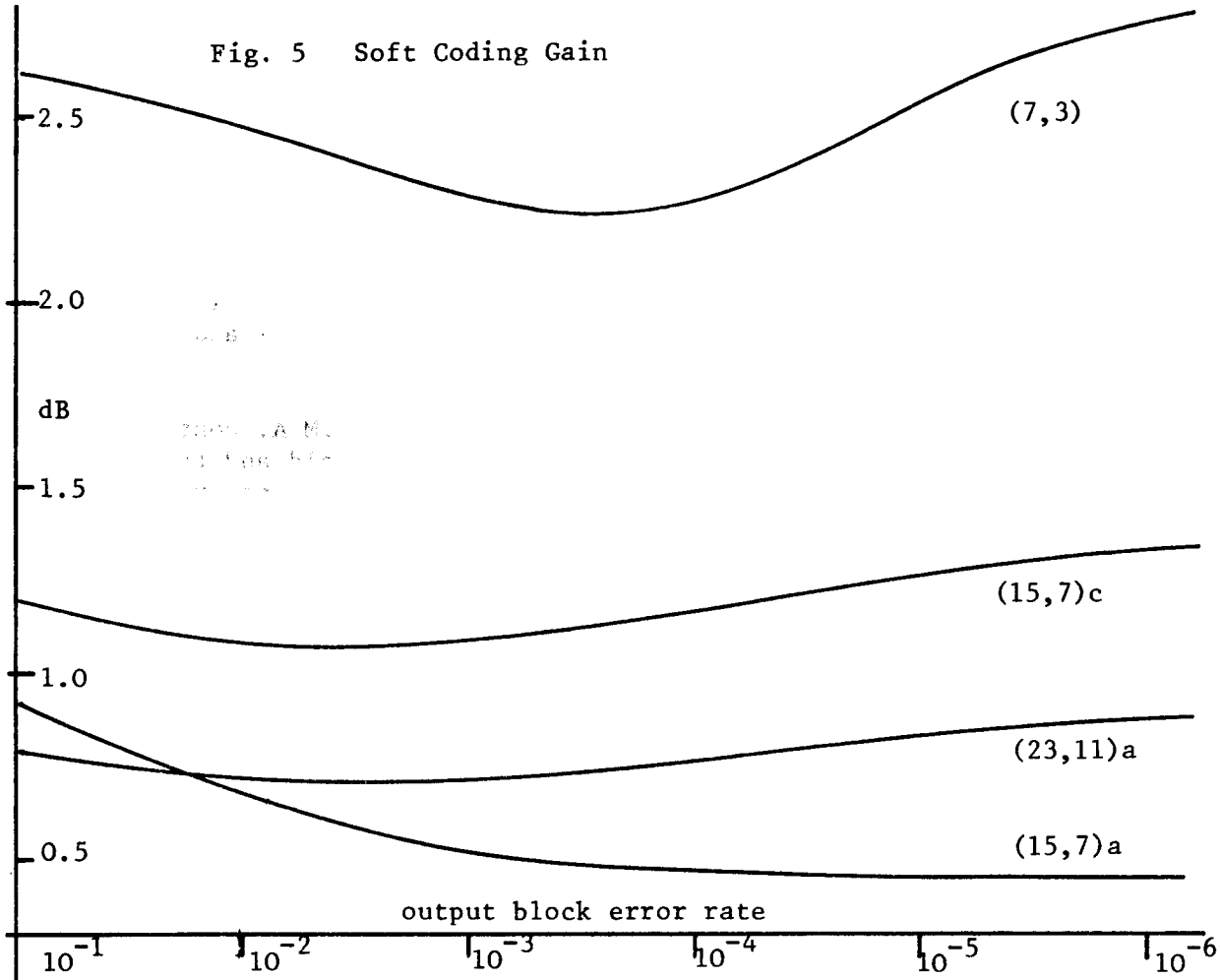
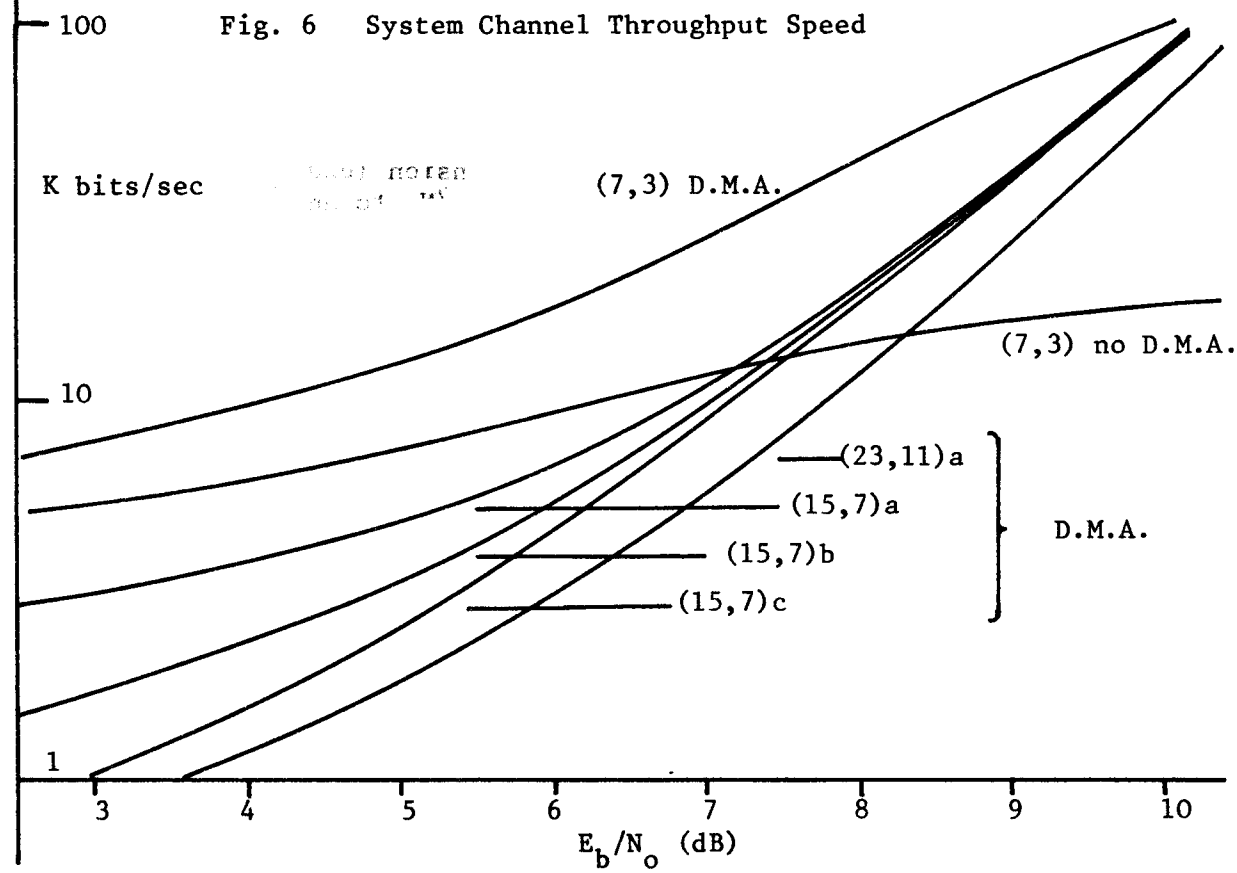


Fig. 6 System Channel Throughput Speed



algorithm 2 when the maximum number of trials is constrained by restricting inversions to those bits having the same least probable confidence level. Curve (15,7)c has a looser restriction on the maximum number of trials in that all single bits with a non-maximum confidence value are successively inverted in order of decreasing confidence. Curves (15,7)a/c thus establish a trade-off between coding gain and number of trials. Finally, curve (23,11)a shows the performance of this code under similar constraints to (15,7)c. In this case much of the code power is being wasted as shown by curve (23,11)b, which estimates the maximum-likelihood bit error performance of this code from the union bound $P_{bit} \simeq \frac{8}{23} \times P_{MLD} \leq \frac{8}{23} \sum_{i=8}^{n-1} W_i \; Q(\sqrt{2\, i\, E_b)N_o})$. Thus, in order to approach more closely to this bound an increase in the number of iterations is required.

Fig. 5 plots soft coding gain (that is, the increase in coding gain over hard-decision bounded-distance decoding) versus output block error rate. The largest improvements are noted for codes in which the number of trappable error patterns of weight $\leq (d-1)$ is significantly larger than the number of patterns of weight $\leq t_h$.

Fig. 6 plots the achievable channel bit throughput speeds versus E_b/N_o for the microprocessor system. The lowest speeds occur for the longer iteratively decoded codes at high channel error rates.

7. Discussion:

Two new soft-decision algorithms have been presented in this paper. Useful coding gains are achievable on the Gaussian channel, and these are expected to be even greater for a burst-and-random channel. The microprocessor controlled decoding system which has been built and tested with Gaussian noise is capable of implementing these algorithms at a rate well above the required 1K bits/sec, at low channel error rates. When using long codes at high channel error rates however, the possibility of buffer overflow exists. The degradation in output bit error rate is not as great as might be expected however, because the coding gain provided by most codes is small at high channel error rates. In addition, the use of latest-generation microprocessors can increase operating speeds by a factor of 2 to 3 times.

8. References:

1. Wozencraft, J.M., and Jacobs, I.M.: Principles of Communication Engineering., Wiley, New York, 1965
2. Chase, D.: 'A Class of Algorithms for Decoding Block Codes with Channel Measurement Information'. IEEE Trans., IT-18, 1972.
3. Goodman, R.M.F. and Farrell, P.G.: 'Data Transmission with Variable-Redundancy Error Control over a High-Frequency Channel'. Proc. IEE, Vol.122, 1975.
4. Forney, G.D.: 'Generalised Minimum Distance Decoding'. IEEE Trans., IT-12, 1966.