7-1996

A Cascadable Pragmatic Block Decoding Algorithm Exploiting Channel Measurement Information

William H. Thesling
Cleveland State University

Fuqin Xiong
Cleveland State University, f.xiong@csuohio.edu

A CASCADABLE PRAGMATIC BLOCK DECODING ALGORITHM EXPLOITING CHANNEL MEASUREMENT INFORMATION

WILLIAM H. THESLING AND FUQIN XIONG
Department of Electrical Engineering, Fenn College of Engineering, Cleveland State University, Euclid Avenue at East 24th Street, Cleveland, Ohio 44115, U.S.A.

SUMMARY

The complexity of algorithms to perform soft decision decoding on block codes has impeded their inclusion in practical systems. A well-known class of algorithms for decoding block codes utilizing channel measurement information along with the algebraic properties of the code are the Chase algorithms.[1] In this paper a decoding method similar to Chase's third algorithm is presented. However, in this method, a single test pattern or alternate codeword makes up one stage of the decoder. The method uses information from the previous decoding(s) to assist in generating a test pattern. This single stage 'Second Chance Algorithm' can then be extended to a 'Third Chance Algorithm' (and beyond) to enhance performance. The method does not invoke the hard decision decoder as often as the Chase algorithms.

1. INTRODUCTION

It is well known that maximum likelihood decoding involves finding the codeword that is closest in Euclidean distance to the received vector. For example, consider an (n,k) binary block code. To perform maximum likelihood decoding on a received vector, the codeword that minimizes the (squared) Euclidean distance between the received vector and all possible codewords is searched. This is the codeword j that minimizes

$$\begin{aligned} dist_j^2 &= \sum_{i=1}^{n}(C_j(i) - r(i))^2 \\ &= \sum_{i=1}^{n} C_j^2(i) - 2\sum_{i=1}^{n} C_j(i)r(i) + \sum_{i=1}^{n} r(i)^2 \end{aligned}$$

wheere $C_j(i)$ is the ith element of the jth codeword and $r(i)$ is the ith element in the received vector $\mathbf{r}$ to be decoded. Note that the binary digits (0,1) have been defined as the real number (−1,+1) respectively. The first term and the last term are constants. The codeword that is closest to the received vector $\mathbf{r}$ is the codeword j that maximizes the correlation CC_j, where CC_j is defined by

$$CC_j = \sum_{i=1}^{n} C_j(i)r(i)$$

If signals of the form (−1,+1) are corrupted by additive white Gussian noise (AWGN), the decoding task is to correlate the received vector $r(i)$ with each of the 2^k possible transmitted sequences or codewords, and choose the one with the highest correlation. This is also known as correlation decoding. This 'direct' approach is appropriate for small codes or low rate codes where the number of codewords is small. However, it quickly becomes intractable for codes with large k. For example, direct correlation decoding of the (24,12) extended Golay code requires approximately 98 000 addition equivalent operations to decode the 12 information bits.

Several authors have devised decoding schemes for block codes which use channel measurement information yet do not require the computational complexity of correlation decoding.[1–5] Perhaps the most widely known methods of exploiting some or all of the soft information are the Chase algorithms.[1] David Chase developed three algorithms. All start with hard decision data and 'reliability' information. For signals of the form (−1,+1) and AWGN, this reliability information can be thought of as the absolute value of the signal measured at the bit time. Thus, for an (n,k) block code, we have two length n vectors. One is the hard decision bits. The other is the absolute value of the 'analog' signal which gave rise to those bits. It is usually quantized to three or four bits. This is also referred to as the 'soft information' or 'channel measurement information'. All of the Chase algorithms involve the concept of generating alternate hard decision vectors, and passing them to a regular hard decision decoder. Each alternate differs from the received vector by some error pattern or test pattern. These test patterns are generated, vector XORed to the received hard decision vector to generate several alternates. The alternates are then decoded using a hard decision decoder. This yields several codewords. Correlation is then used to decide among them. The key issues are: (a) how many alternates are required? (b) how are they created?

The notion of a test pattern is required to explain Chases' algorithm(s), and the second chance algorithm. A test pattern is simply a vector of length n, with 1s and 0s in it. This pattern is used to generate an 'alternate received binary vector'. This alternate vector is simply the result of XORing the received binary vector with the test pattern. As one might suspect, test patterns are patterns of mostly 0s.

Since Chase's first algorithm is rarely used, it is omitted. For Chase's second algorithm, a relatively large set of alternatives are considered. First the $\lfloor d/2 \rfloor$ (d is the minimum Hamming distance of the code) minimum values in the reliability vector are found, and the locations where they occurred are noted. Next all possible test patterns with ones confined to these locations are produced yielding $2^{\lfloor d/2 \rfloor}$ possible test patterns. This algorithm performs nearly as well as maximum likelihood decoding.

Chase's third algorithm is similar to algorithm 2, except that only $\lfloor d/2 \rfloor + 1$ test patterns are considered. The first test pattern has a single 1 in the position of the lowest confidence value. The second test pattern has three 1s, in the positions of the three lowest confidence values. The third test pattern has five 1s in the positions of the five lowest confidence values, and so on up to $\lfloor d/2 \rfloor + 1$ test patterns. Simulation results in Chase 1 (and repeated here) show that the performance of algorithm III is somewhat inferior to that of algorithms I and II on the (24,12) extended Golay code.

A scheme similar to Chase's third algorithm is considered. However, this method requires a hard decision decoder which always decodes to a nearby (in hamming distance) codeword. Test patterns are found using the results of both the hard decision decoder and reliability information.

2. SECOND CHANCE ALGORITHM (SCA)

In this algorithm, only one alternate or test pattern is created, giving the decoder a *second chance* to find the correct codeword. The (23,12,7) Golay code, and the (24,12,8) extended Golay code are considered and used as examples, but the ideas can be applied to any bit error correcting block code.

In the SCA, the test pattern has t 1s as the t locations of the t minimum confidence values, where t is the error correcting capability of the code, $t = \lfloor (d-1)/2 \rfloor$. However, if t is an even number, there are $t+1$ locations with ones in them in the test pattern. This, however, is not the complete story. The test pattern, and hence the alternate vector, are generated after initial hard decision decoding is done on the received hard decision vector. During the search for the t locations of the t minimum confidence values, those positions *alleged* to be in error by the hard decision decoder are masked out. This operation is optional, however there is a slight improvement in performance when this *error mask* is used.

This can be extended further by performing another search utilizing the outputs of the first two decoders. This idea leads to a third chance algorithm (TCA). The basic idea is to generate the successive test patterns under the assumption that the previous decoders failed to find the correct codeword.

The second chance algorithm can be summarized as follows:

1. Perform hard decision decoding (HDD) on a copy of the hard decision vector with a hard decision decoder which always decodes to a codeword.
2. Note the locations where the hard decision decoder complements bits in the decoding process.
3. If t is odd, find the t locations of the t lowest confidence values from the set of all confidence values where the corresponding bits in the hard decision decoder were not complemented. If t is even, find the $t+1$ locations of the $t+1$ lowest confidence values from the same set of all confidence values as above.
4. Complement the corresponding bits in a copy of the received hard decision vector.
5. Perform HDD on the copy of the hard decision vector (from 4) with a hard decision decoder which always decodes to a codeword.
6. Correlate: for each codeword (one from step 1, and one from step 5), add the confidence values where the bits in the hard decision vector agree with the bits in the codeword and subtract the confidence values where they do not agree.
7. Choose as the output codeword the codeword which gave the highest value in step 6.

3. SCA PERFORMANCE AND ANALYSIS

Consider the performance of the regular hard decision decoder which always decodes to a codeword. For the example code, the (23,12) Golay code, $t = 3$. Therefore the probability of a word error is the probability that each codeword (of length 23) has four or more bit errors. This is given by

$$P_W = \sum_{j=4}^{23} \binom{23}{j} p^j (1-p)^{23-j}$$

where p is the transition probability which for BPSK is given by

$$p = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

Here, E_b is the energy per channel bit, not information bit, and N_0 is the single sided power spectral density of the noise. To get the probability of bit error after decoding the following approximation can be used.[7]

$$P_b \cong \frac{1}{23}\sum_{j=4}^{23}(j+t)\binom{23}{j}p^j(1-p)^{23-j}$$

The $(j+t)$ comes about because when a 'false correction' is made, usually changes occur in t locations, adding t additional bit errors.

Assume that whenever the correct codeword is found, it will 'win' at the correlation stage. The approach is to find out how often the second codeword will yield the incorrect codeword for the cases of four, five and six errors. Thus, after correlation stage,

$$P_b \cong \frac{1}{23}\sum_{j=4}^{23}C_j(j+t)\binom{23}{j}p^j(1-p)^{23-j}$$

where C_j is the probability that a j error event (a received codeword with j errors in it) is not decoded correctly. For $j \geq 7$, $C_j = 1$.

4. SCA ERROR COEFFICIENTS

Consider the weight distribution polynomial for this code.[6]

$$A(z) = 1 + 253z^7 + 506z^8 + 1288z^{11} + 1288z^{12} + 506z^{15} + 253z^{16} + z^{23}$$

Because this is a linear code, the sum (mod 2, or a vector XOR) of any two codewords is another codeword. The weight distribution polynomial also gives the hamming distance distribution. We can therefore, without loss of generality, look at the case where the all zeros codeword was the transmitted codeword. There are 253 codewords which are a hamming distance of seven away, 506 codewords which are a hamming distance of eight away, and none at a hamming distance of nine or 10. It is important to keep in mind that the hard decision decoder always decodes to a codeword. This condition falls out naturally for the (23,12) Golay code because it is a perfect code. For other codes, this condition may need to be forced. The output of the hard decision decoder must be a codeword and this codeword is therefore different from the transmitted codeword by zero positions (a correct decoding), seven positions, eight positions, 11 positions, etc. all the way to 23 positions. The error correcting capability of this code is three, therefore the output of the hard decision decoder can differ from the received hard decision vector by at most three positions. This code has a minimum distance of seven, and an error correcting capability of three. If a codeword is received with four errors in it, the hard decision decoder *must* identify three bits as 'in error', and these three positions *must* be different from the four locations which *are* in error. Therefore, whenever there are four errors in a received codeword, the locations identified as being 'in error' by the hard decision decoder must correspond to locations where the bits are *correct*. Similarly, whenever a codeword is received with five errors in it, the locations identified as being 'in error' by the hard decision decoder correspond to locations where the bits are *correct*.

4.1. C_4

Given a code word of length 23 with four errors, the objective of the SCA is to identify correctly at least two positions which are in error, out of three guesses at it. If the guess 'hits' two bits correctly, two bits will be corrected. An additional error will be made for the third bit position which was not one of the four bits in error. This will reduce the total number of errors to three. Then the vector is passed to the hard decision decoder which will yield the correct code word. If the guess 'hits' on all three bit positions, then the total number of errors is reduced to one. The vector is again passed on to the hard decision decoder and the error is corrected. Therefore, the SCA needs to identify correctly at least two positions which are in error. This is the probability that two of the three lowest confidence values came from the set of four errors. This is out of a set of 23. Using the fact that the hard decision decoder identified three bit positions which are correct, this becomes the probability that the three lowest confidence values came from the set of four errors, out of a set of 20. Obtaining this coefficient from theory is rather painful, however the effect is quite easily simulated in a computer. This was done for varying values of SNR to obtain C_4 vs. SNR values.

4.2. C_5

Given a code word of length 23 with five errors, the objective of the SCA is to identify correctly three positions which are in error, out of three guesses at it. This is the probability that the three lowest confidence values came from the set of five errors. This is out of a set of 23. Using the fact that the hard decision decoder identified three bit positions (or two bit positions, but three is more likely) which are correct, this becomes the probability that the three lowest confidence values came from the set of three errors, out of a set of 20 (or 21). Again, this was simulated on a computer to get C_5 for various SNR values.

4.3. C_6

Getting the coefficient C_6 is essentially the same as for C_5. This is the probability that the three lowest confidence values came from the set of six errors. This is out of a set of 23. 'Masking out' those positions alleged to be in error by the hard decision decoder changes the situation slightly. The hard decision decoder will usually identify three locations where the bits are 'in error', one of which will correspond to one of the six positions in error in the hard decision vector. Masking out these errors will leave us with only five errors in our set of 20 which we will search through. This is the same situation as for C_5. Therefore, $C_6 = C_5$ when using the error mask.

Curves of theoretical performance for the SCA on the (23,12) Golay code with and without masking out the errors from the hard decision decoder are given in Figure 1. The theoretical curves (solid lines) were obtained via simulation to obtain the coefficients C_4, C_5 and C_6. The 'dots' near or on the curves are results from a simulation of the second chance algorithm on this code.

5. SCA ENHANCEMENT

A slight enhancement can be made to the second chance algorithm. The number of 1s in the test pattern can be adjusted as a function of the number of 1s in the error pattern (the error pattern's hamming weight $= ew$). If the received codeword has four errors in it, the hard decision decoder will identify three bit positions as being 'in error' $ew = 3$. Whenever the hard decision decoder generates an error pattern of weight $ew = 3$, which is the most likely case when there is a decoding error, proceed as described. If, however, the received hard decision vector has five errors in it, the hard decision decoder will generate an error pattern of weight $ew = 2$, or $ew = 3$. Therefore, for the case when the hard decision decoder generates an error pattern of weight $ew = 2$, the SCA can use four 1s in the test pattern, instead of three. Because the SCA is attempting to correct a codeword with five errors, had a test pattern been chosen with three 1s, all three would need to correspond to bits in error. However a test pattern with four 1s requires only three of the four to correspond to bits in error. The probability of this event is considerably higher. Similarly, if the hard decision decoder had generated an error pattern of weight $ew = 1$, a weight five test pattern would be used. If t is odd, the weight of the test pattern wtp is chosen to be

$$wtp = d_{\min} - 1 - ew$$

If, however, t is even, then

$$wtp = d_{\min} - ew$$

The improvement in performance due to this enhancement is quit small for the (23,12) Golay code ($\approx 3\%$ reduction in BER at 10^{-6}), and was not used in the simulation data for the (23,12) Golay code. However the improvement in performance due to this enhancement for the (24,12) extended Golay code is worth noting, ($\approx 30\%$ reduction in BER at 10^{-6}). This enhancement was used in the simulation data for the (24,12) extended Golay code (Figure 2). This improvement comes about because the (24,12) code is an even parity code. There are no codewords of odd value weight. This is illustrated by the weight distribution polynomial for this code.

$$A(z) = 1 + 759z^8 + 2576z^{12} + 759z^{16} + z^{23}$$

For the case where a received hard decision vector

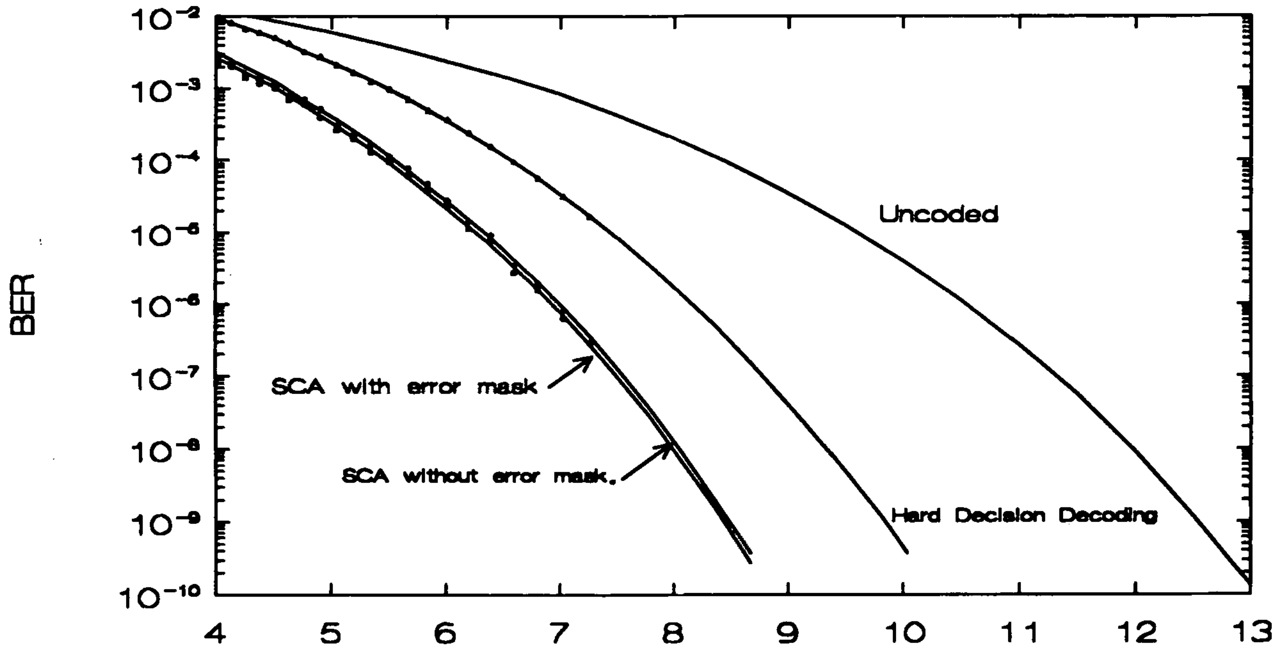

Figure 1. Golay code (23,12)

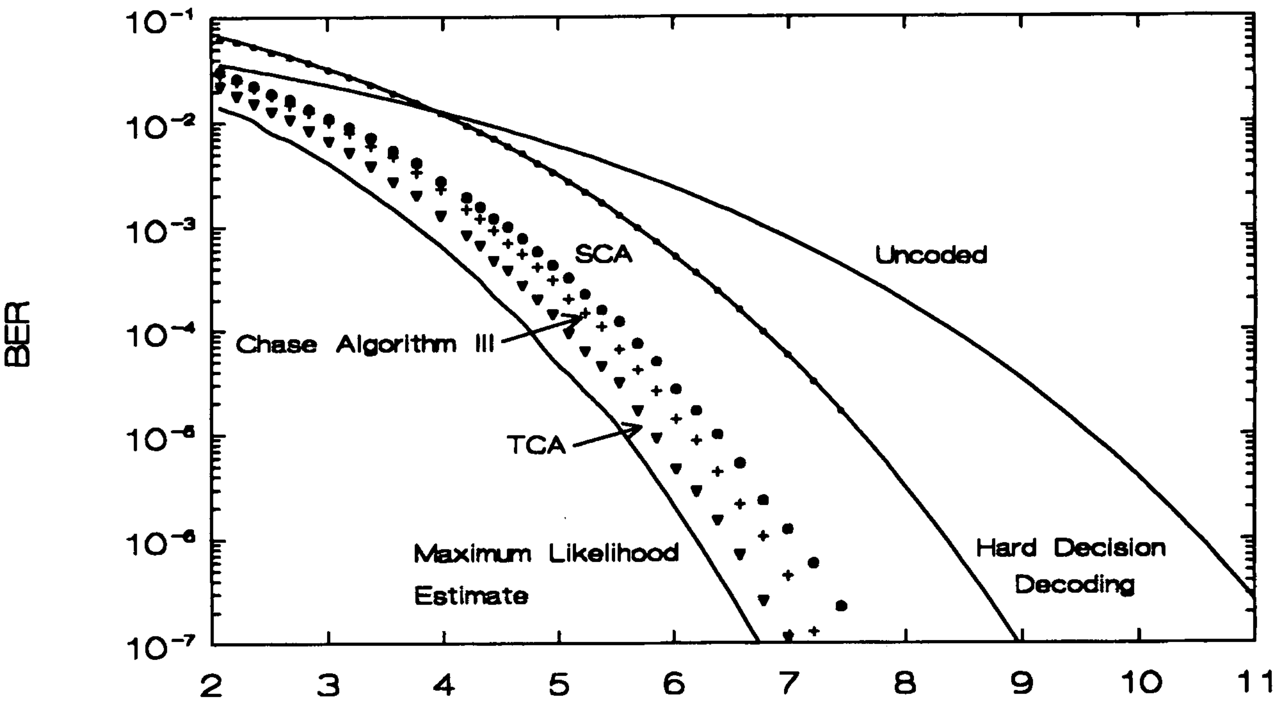

Figure 2. Extended Golay code (24,12)

has four errors in it, the hard decision decoder must generate an error pattern of weight four (forced decoding to a codeword) because the minimum distance of this code is eight. If however, the hard decision vector has five errors in it, the hard decision decoder must generate an error pattern of weight three. Four is the maximum weight error pattern the hard decision decoder can generate and this would cause a decoding which differs from the original transmitted codeword by an odd number of positions (the hamming weight is an odd number) which cannot happen, therefore a weight three error pattern must result. Therefore, when the hard decision decoder generates an error pattern of weight three, if the decoding is in error, there are five bits in error (or seven or nine etc.) in the original hard decision vector.

Figure 2 is a graph of the performance of the (24,12) extended Golay code with SCA decoding as well as Chase's third algorithm. The hard decision decoding curve corresponds to the case where the hard decision decoder is forced to decode to a nearest codeword. The SCA simulation data was found using the SCA with the error mask and enhancement as mentioned above. Finally, an extension to this algorithm employing an additional decoding was performed to arrive at a third chance algorithm (TCA). Simulation data is plotted for this algorithm as well.

6. THIRD CHANCE ALGORITHM (TCA)

This algorithm proceeds in a very similar manner to the second chance algorithm. The approach is to generate a third test patern under the assumption that the hard decision decoder failed to find the correct codeword, and the SCA also failed to find the correct codeword.

The third chance algorithm can be summarized as follows:

1. Perform the SCA through step 5.
2. Perform the vector XOR of the output codeword from the hard decision decoder with the output codeword from the SCA. Call the resultant vector the *Hamming difference vector.*
3. Find from the set of the non-zero elements in the Hamming difference vector, the j locations with the largest confidence values. j will be defined shortly.
4. Find from the set of the zero elements in the Hamming difference vector, the k locations with the minimum confidence values. k will be defined shortly.
5. Make a copy of the resultant codeword from step 5 in the SCA. Complement the bits in the j locations from step 3, and the k locations from step 4.
6. Perform HDD on the vector from step 5 with a hard decision decoder which always decodes to a codeword.
7. Correlate: for each codeword (one from step 1 and one from step 5 in the SCA, and step 6 in the TCA), add the confidence values where the bits in the hard decision vector agree with the bits in the codeword and subtract the confidence values where they do not agree.
8. Choose as the output codeword the codeword which gave the highest value in step 7.

The values for j and k are determined from the specifics of the code. This can be a little tricky. The hamming distance vector will have a hamming weight at least as large as the minimum distance of the code. This is a direct consequence of the way the second codeword was generated. Given that the first codeword is in error, and the second codeword is in error, both codewords must have at least $d_{\min}$ positions in error. The most likely location $\lceil \frac{d_{\min}}{2} \rceil$ errors are within the set of the non-zero locations of the hamming difference vector. Therefore, j is taken to be at least $\lceil \frac{d_{\min}}{2} \rceil$. The value of k is determined in a similar manner as the weight of the test pattern in the SCA. If t is even, $k = t + 1$. If t is odd, $k = t$. The TCA analog of the SCA enhancement is performed similarly. However in this case rather than considering the number of 'errors' found by the hard decision decoder, we consider the number of non-zero elements remaining in the hamming difference vector. That is the weight of the hamming difference vector minus j.

For the simulation data on the extended Golay Code in Figure 2, the TCA algorithm used the SCA with the enhancement, and the fixed values of $j = 5$, and $k = 4$ for the third stage. The value of $j = 5$ was found via a simulation search. This leads to $k = 4$. Both j and k were varied independently with the $j = 5$, $k = 4$ combination yielding optimum results.

7. CONCLUDING REMARKS

The decoding algorithm just described, while similar to Chase's third algorithm, differs in that this algorithm requires the use of a hard decision decoder which *always* decodes to a nearby codeword. The results of this decoding are used in constructing an alternate codeword to be decoded. Correlation is then used to decide between the two codeword candidates. This decoding method can be extended to the third chance algorithm where the results of the first and second decoding are used in generating another test pattern and a third codeword. It should be mentioned that both SCA and TCA ran faster than Chase Algorithm III in the computer simulation, with SCA the fastest. However this is a function of the specifics of the computer, the source code, etc.

ACKNOWLEDGEMENTS

W. H. Thesling was supported by a NASA space grant/OAI Fellowship.

REFERENCES

1. D. Chase, 'A Class of algorithms for decoding block codes with channel measurement information', *IEEE Transactions on Information Theory*, **IT-18**, 170–182 (1972).
2. E. J. Weldon Jr., 'Decoding binary block codes on Q-ary output channels', *IEEE Transactions on Information Theory*, **IT-17**, 713–718 (1971).
3. C. C. H. Yu and D. J. Costello Jr., 'Generalized minimum distance decoding algorithms for Q-ary output channels', *IEEE Transactions on Information Theory*, **IT-26**, 238–243 (1980).
4. J. Yuan and C. S. Chen, 'Correlation decoding of the (24,12) Golay code using neural networks', *IEE Proceedings—1* **138**, (6), 517–524 (1991).
5. Y. Be'ery and J. Snyders, 'Optimal soft decision block decoders based on fast Hadamard transforms', *IEEE Transactions on Information Theory*, **IT-32**, 355–364 (1986).
6. S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice Hall, 1983, p. 138.
7. B. Sklar, *Digital Communications Fundamentals and Applications*, Prentice Hall, 1988, p. 301.