Adaptive cross-parity (AXP) code for a high-density magnetic tape subsystem

by Arvind M. Patel

This paper describes an error-correction system, called adaptive cross-parity (AXP) code, for the IBM 3480, a new high-density 18-track tape storage subsystem. Redundancy is applied to two interleaved sets of nine tracks in the same proportion as that in the previous IBM 3420 tape machines. The coding structure, however, is simpler, for it avoids the complex computations of Galois fields. The coding structure is based on a concept of interacting vertical and cross-parity checks, where the cross-parity checks span both sets of tracks and are used in either set in an adaptive manner. As a result, the overall error-correcting capability is substantially improved without increasing the redundancy. Decoding, in which simple parity equations are processed, is designed to progress iteratively. By means of adaptive use of redundancy, the new method corrects up to three known erroneous tracks in any one set of nine tracks and up to four known erroneous tracks in the two sets together. The code also identifies the first unknown erroneous track in each of the two sets, and subsequently identifies the second unknown erroneous track in one of the two sets while providing correction for all these tracks. The result is generalized for a system with any number of tracks divided into a multiple number of unequal sets.

Introduction

The IBM 3480 Magnetic Tape Subsystem uses half-inch-wide chromium dioxide tape in a compact cartridge, with a storage capacity of 200 million bytes per cartridge. The data are recorded at a linear density of 38 000 bytes per inch and delivered at a rate of more than three million bytes per second. The data density is six times that of its predecessor, the IBM 3420 tape machine; and the delivery rate has been more than doubled. The new product offers significantly greater data reliability, with major reductions in space, power, and maintenance requirements.

Tape storage products traditionally have made use of various error control coding schemes in order to preserve customer data integrity and provide high reliability. The cyclic redundancy check (CRC) was introduced with IBM 2401 Models 1, 2, and 3 at a data density of 800 bpi, which allowed correction of one erased track on the reread operation [1]. The IBM 2401 Models 4, 5, and 6, at a data density of 1600 bpi, used on-the-fly correction of an erased track. The IBM 3420 Models 4, 6, and 8, at a data density of 6250 bpi, introduced an error correction scheme [2] which provided on-the-fly correction of two erased tracks out of nine recorded tracks. Later, a novel tape product, the IBM

3850 Mass Storage System, with its new data format, used error correction coding to correct up to two 16-byte sections in each segment of 240 bytes of serial data [3].

In another scheme [4] for multitrack magnetic tape, correction of two erased tracks was obtained with almost the same redundancy as in the code for the IBM 3420, using two parity checks in place of Galois field algebra. We used a similar concept to obtain correction of three erased tracks in a multitrack system with one vertical-parity check and two cross-parity checks [5].

The IBM 3480 Magnetic Tape Subsystem introduces a tape cartridge with 18-track data format which uses a new coding scheme called adaptive cross-parity (AXP) code [6]. In this scheme, the 18 tracks are divided into two interleaved sets of nine tracks, with each set consisting of seven data tracks and two check tracks. The proportion of check tracks is thus the same as in the nine-track scheme of the IBM 3420. However, by adaptive use of the checks in the two interleaved sets, the new scheme corrects up to three erased tracks in any one set of nine tracks and up to four erased tracks in the two sets together. Going one step further, we could use a Reed-Solomon code over GF(256) with the same amount of redundancy and provide correction of ANY four erased tracks. In that case, the decoding function for all correctable combinations of errors and erasures is substantially more complex. In contrast, the interleaving of the sets with the aforementioned AXP capability provides adequate protection against the more likely combinations of errors and erasures, while keeping the decoding function very simple.

The AXP code is convolutional. The coding structure is based on the concept of interacting vertical- and cross-parity checks [5]. The vertical-parity checks are applied independently to each of the two sets of tracks, and the cross-parity checks extend over both the sets, providing adaptive usage of redundancy. The decoding procedure is iterative and uses parity equations which involve only one unknown variable at a time. The resulting implementation is simple and inexpensive.

During the iterative error correction process, the decoder identifies an approaching new erroneous track and corrects up to two erroneous tracks in any one of the sets and up to three erroneous tracks in the two sets together. The third erroneous track in one set and the fourth erroneous track of the two sets together are corrected on-the-fly or on reread when they are identified by external means as erasures.

This paper focuses on the case of two conventional nine-track sets in an 18-track system. However, the general result is applicable to systems with any number of tracks in which the two sets need not have the same number of tracks. Furthermore, the multitrack system could be divided not only into two but also into three or more sets, each of which could adaptively share the total capability of cross-parity checks.

Encoding equations

As shown in **Figure 1**, there are 18 parallel tracks recorded along the tape. The tracks are grouped into two sets; set A consists of any nine parallel tracks, and set B consists of the remaining nine parallel tracks. In Fig. 1 the two sets are shown side by side with a symmetrically ordered arrangement of the tracks. This is done for convenience in describing the code. In actual practice, however, the tracks from two sets are interleaved and may be arranged in any other order.

Let $A_m(t)$ and $B_m(t)$ denote the mth bit in the track t of set A and set B, respectively. The track number t takes on values from 0 to 8 in each set. The bit position m takes on values from 0 to M. Tracks labeled 0 and 8 in each set are check tracks.

Each check bit in track 0 of set A provides a cross-parity check along the diagonal with positive slope, involving bits from both sets as seen in Fig. 1. The mth cross-parity check of set A is given by the encoding equation

$$A_m(0) = \bigoplus_{t=1}^{7} A_{m-t}(t) \oplus \bigoplus_{t=0}^{7} B_{m+t-15}(t). \tag{1}$$

(In this and subsequent equations, a circle superimposed on the summation symbol or plus sign—i.e., $\bigoplus$ or $\oplus$—indicates modulo-2 sum.)

Each check bit in track 0 of set B provides a cross-parity check along the diagonal with negative slope, involving bits from both sets, as seen in Fig. 1. The mth cross-parity check of set B is given by the encoding equation

$$B_m(0) = \bigoplus_{t=1}^{7} B_{m-t}(t) \oplus \bigoplus_{t=0}^{7} A_{m+t-15}(t). \tag{2}$$

Equations (1) and (2) can be rewritten in a more convenient symmetrical form as

$$\bigoplus_{t=0}^{7} [A_{m-t}(t) \oplus B_{m+t-15}(t)] = 0, \tag{3}$$

$$\bigoplus_{t=0}^{7} [B_{m-t}(t) \oplus A_{m+t-15}(t)] = 0. \tag{4}$$

Note that at the beginning of the record, computations of the cross-parity check bits for positions 0 to 15 involve data bit values from void positions (with negative position numbers). For convenience, these data bit values are considered to have zero binary value. At the end of the record, in order to provide diagonal checks to all bits in each track, the zero check track in each set is extended 15 positions. The check bits on the extended positions also involve some data bit values from void positions which are assumed to have zero binary value.

Each check bit in the eighth track of set A is a vertical parity over the bits of same position number m in set A. The mth vertical-parity check of set A is given by the equation

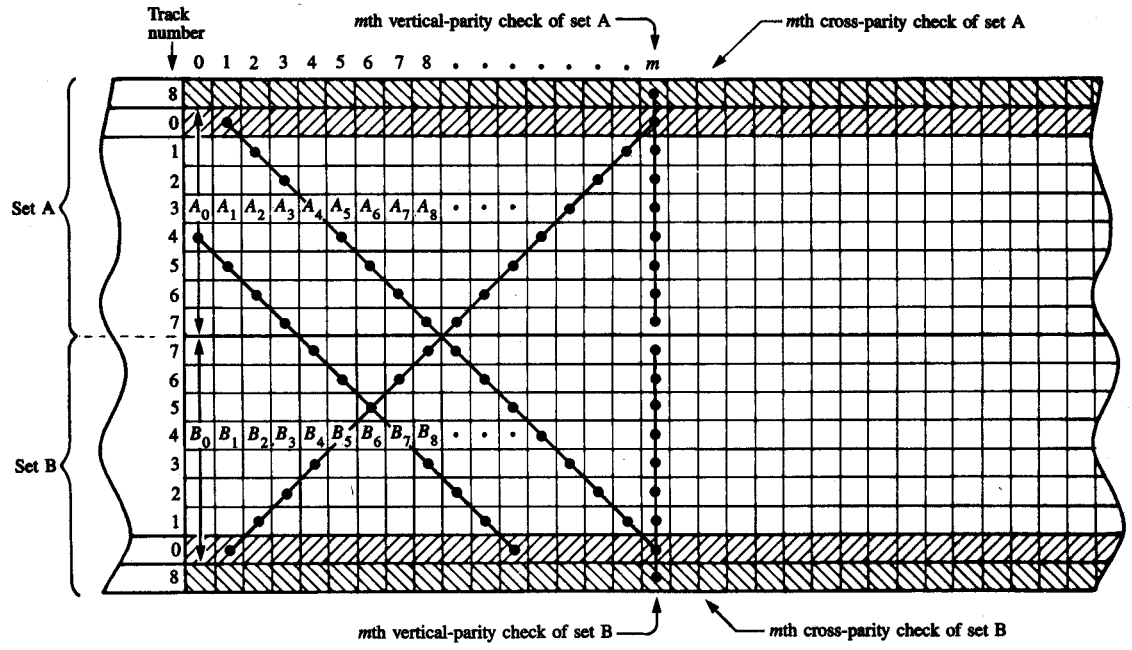

Figure 1

Data format: 18 tracks grouped into two sets.

$$\sum_{t=0}^{8} A_m(t) = 0. \tag{5}$$

Similarly, the mth vertical-parity check of set B is given by the equation

$$\sum_{t=0}^{8} B_m(t) = 0. \tag{6}$$

Other data formats

The two cross-parity checks can be obtained in various ways. We have presented one data format in Fig. 1 in which the cross-parity check bits appear in two check tracks. Thus, the vertical characters A_m and B_m consist of seven data bits and two check bits. This is different from the conventional nine-track recording of half-inch magnetic tape where the vertical characters are bytes consisting of eight data bits and a vertical-parity check bit. In **Figure 2** we present a data format that is compatible with such a convention. In this format A_0, A_8, A_{16}, A_{24}, $\cdots$ provide the cross-parity check with positive slope, and B_0, B_8, B_{16}, B_{24}, $\cdots$ provide the cross-parity check with negative slope. All other bytes are data bytes consisting of the conventional eight data bits with a vertical-parity check bit. The encoding and decoding equations are not affected by this change in the format. The encoding process is slightly changed in the input and output of the encoder. The decoding algorithm and hardware remain the same.

Another data format, which is used in the IBM 3480 storage subsystem, is shown in **Figure 3.** In this format, the 18-track record is partitioned into blocks, each of which contains 14 data bytes and four check bytes placed along the tracks. In this arrangement there are still four check tracks as in Fig. 1; however, the conventional eight-bit bytes are used in the encoding and decoding process and are recorded as "bytes along the tracks." In the IBM 3480 application, these bytes-along-the-tracks are further coded into nine-bit patterns using a run-length-limiting modulation code which limits the run length of consecutive 0s in the coded sequence to zero, one, two, and three only. Unique mapping of this (0, 3) 8/9 modulation code is realized [7] through simple logic equations and includes two additional features:

1. Each code word possesses opposite parity relation with the corresponding data byte.
2. The pattern $S = 100010001$ is not a code word and also does not occur anywhere in the coded sequence; S is used as a synchronization pattern at selected positions in the data stream to identify format boundaries.

The coded sequences are then recorded as waveforms on tracks of set A and set B which are interleaved in their physical positions on the tape.

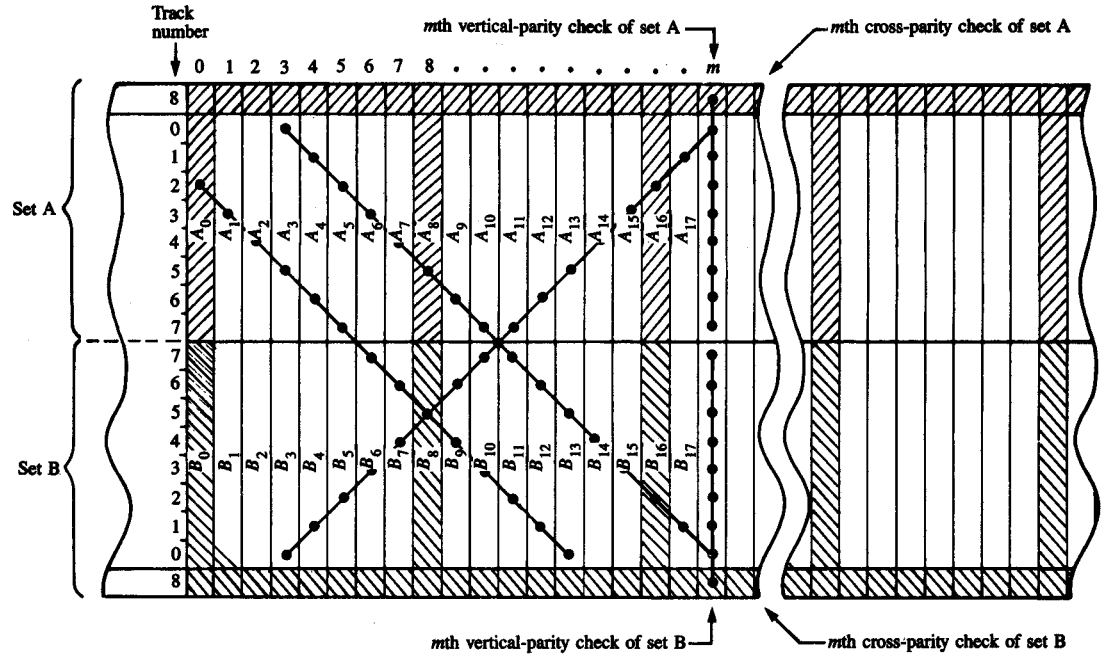

Figure 2

An alternate data format: "bytes across the tracks."

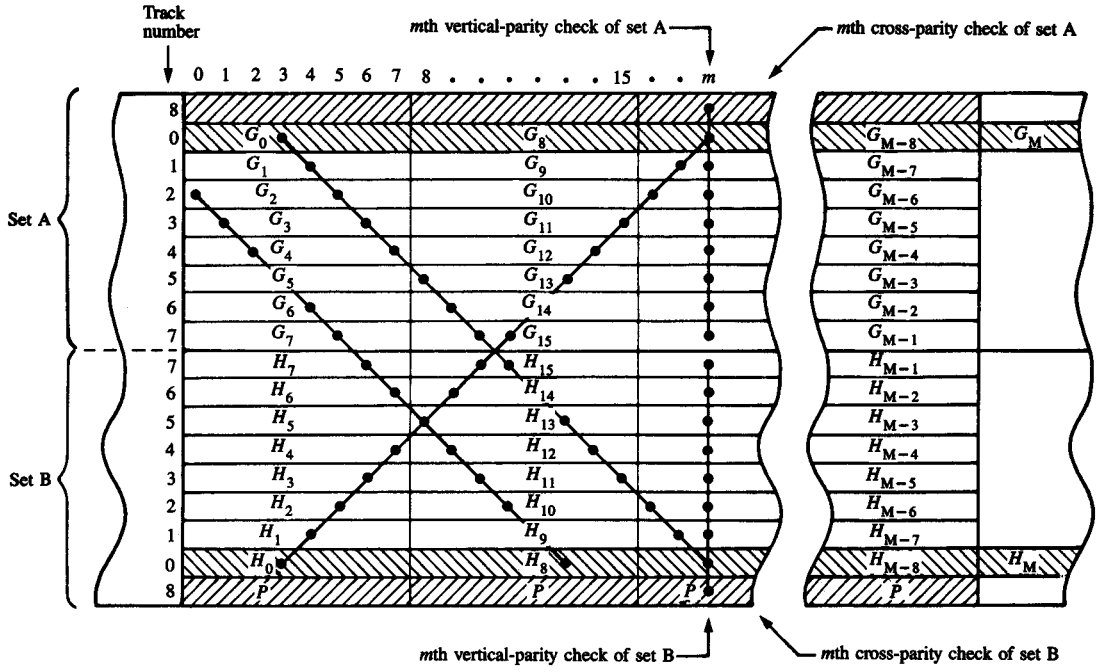

Figure 3

An alternate data format: "bytes along the tracks."

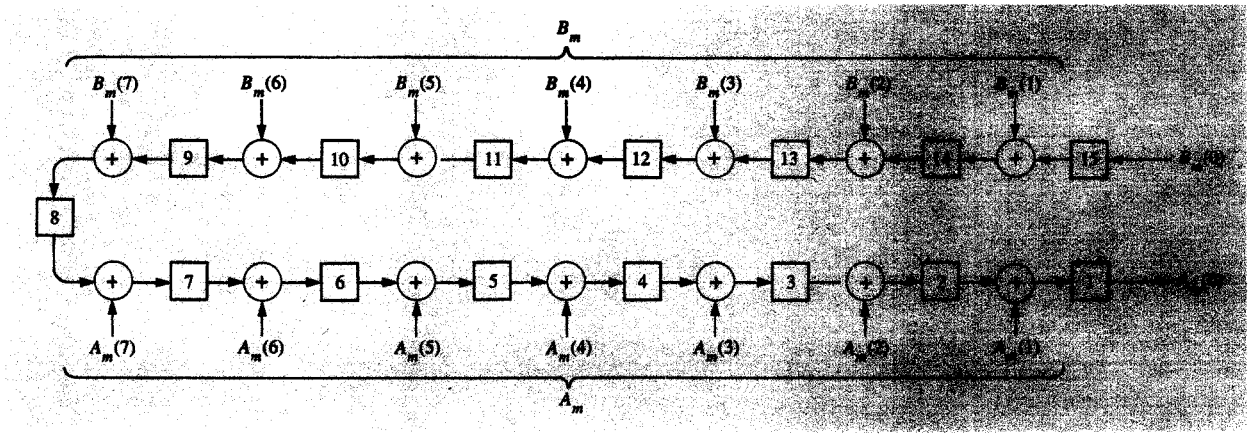

Figure 4

Generation of cross-parity check bits for set A.

Implementation of the encoding process

The encoding equations (1), (2), (5), and (6) are simple parity equations. These can be implemented by means of exclusive-OR circuits receiving inputs of appropriate data bit values stored in a random-access buffer memory prior to recording on the magnetic tape.

The computation of cross-parity checks for the mth bit position in the data format of Fig. 1 requires prior bit values ranging from bit positions $m - 1$ to $m - 15$. Alternatively, we could use a shift register, as shown in **Figure 4.** The shift register processes the incoming data characters A_m and B_m and produces the mth cross-parity check $A_m(0)$ of set A at the output. At the same time, the partial computations of the cross-parity checks $A_{m+1}(0)$ through $A_{m+15}(0)$ are being collected in its storage positions 1 through 15, respectively.

The input to the last stage in this shift register is $B_m(0)$, which is the mth cross-parity check for set B as computed at the output of a similar shift register for set B. All stages of the shift register are initially set to 0. The shift register produces the cross-parity check $A_m(0)$ at the output, as the data characters A_m and B_m and the check bit $B_m(0)$ are entered at the input. The process is continuous as m is incremented from 0 to the last bit position value M of the record. At the end, the contents of the shift register are shifted out and stored in the extended 15 positions of track zero in set A.

In the data format of Fig. 2, the computation of any cross-parity check byte A_{8n} requires bit values ranging from data byte positions $8n - 7$ to $8n + 7$ in set A, and byte positions $8n - 15$ to $8n - 1$, including the previously computed check byte B_{8n-8} in set B. In this case also, we can devise a shift register to process the incoming bytes to compute the check bits in an iterative manner. However, a buffer memory is required in any case, since the computation requires data bytes from positions preceding and following the check byte.

In the data format of Fig. 3, the incoming bytes are the characters G and H along the tracks. Here also, a buffer memory is required to compute and arrange the check characters in proper order.

Error syndromes

Let $\hat{A}_m(t)$ and $\hat{B}_m(t)$ denote the bit values corresponding to $A_m(t)$ and $B_m(t)$, respectively, as they are read from the tape. These readback bits may be corrupted by errors. The result of the parity checks of Eqs. (3–6) applied to the readback data is called the syndrome of error. A nonzero syndrome is a clear indication of the presence of an error.

The mth cross-parity check of set A yields the syndrome

$$Sd^a_m = \sum_{t=0}^{7} \hat{A}_{m-t}(t) \oplus \hat{B}_{m+t-15}(t). \tag{7}$$

The mth cross-parity check of set B yields the syndrome

$$Sd^b_m = \sum_{t=0}^{7} \hat{B}_{m-t}(t) \oplus \hat{A}_{m+t-15}(t). \tag{8}$$

The mth vertical check for set A yields the syndrome

$$Sv^a_m = \sum_{t=0}^{8} \hat{A}_m(t). \tag{9}$$

The mth vertical check for set B yields the syndrome

$$Sv^b_m = \sum_{t=0}^{8} \hat{B}_m(t). \tag{10}$$

The modulo-2 difference between the read $\hat{A}_m(t)$ and the written $A_m(t)$ is called the error pattern $e^a_m(t)$ in the mth position of track t in set A.

So for set A we have

$$e^a_m(t) = \hat{A}_m(t) \oplus A_m(t). \tag{11}$$

Similarly, for set B

$$e^b_m(t) = \hat{B}_m(t) \oplus B_m(t). \tag{12}$$

Now combine Eqs. (3) and (7), (4) and (8), (5) and (9), and (6) and (10); and substitute $e^a_m(t)$ and $e^b_m(t)$ of Eqs. (11) and (12) to obtain relations between errors and syndromes as follows:

$$Sd^a_m = \bigoplus_{t=0}^{7} e^a_{m-t}(t) \oplus e^b_{m+t-15}(t), \tag{13}$$

$$Sd^b_m = \bigoplus_{t=0}^{7} e^b_{m-t}(t) \oplus e^b_{m+t-15}(t), \tag{14}$$

$$Sv^a_m = \bigoplus_{t=0}^{8} e^a_m(t), \tag{15}$$

$$Sv^b_m = \bigoplus_{t=0}^{8} e^b_m(t). \tag{16}$$

Many different types of errors can be corrected by processing these syndromes. In tapes the predominant errors are track errors caused by large-size defects in the magnetic medium. The erroneous track may be identified by detecting loss of signal, excessive phase error, inadmissible recording pattern, or any other similar external pointer. In the absence of such external pointers, the erroneous track can be identified by processing the syndromes. We show that any one of the following combinations of track errors can be corrected by processing the syndromes:

1. Up to three known erroneous tracks in one set; *and* up to one known erroneous track in the other set.
2. Up to one unknown or two known erroneous tracks in each of the two sets.
3. Up to one known *and* one unknown erroneous track in one set; *and* up to one known erroneous track in the other set.

Correction of errors in known erroneous tracks

• *Three-track correction in set A*

Errors confined to three known tracks in set A are correctable if set B is error-free or has only one known erroneous track. The erroneous tracks are indicated by track-error pointers i, j, k in set A and y in set B. If y is undefined, then set B is assumed to be error-free.

For convenience in decoding, i is the lowest and k is the highest track index among the erroneous tracks from track number 0 to 7. Track j is the remaining erroneous track so that either ($i < j < k$) or ($j = 8$ and $i < k$).

Since set B has only one known erroneous track, the vertical-parity-check syndromes Sv^b_m yield the error patterns for this track. On eliminating the known zero-error patterns corresponding to the error-free tracks, Eq. (16) can be rewritten as

$$Sv^b_m = e^b_m(y). \tag{17}$$

Assume that all errors are corrected up to byte $(m-1)$ and the syndrome equations are adjusted for all corrected error patterns. Then, as shown in **Figure 5,** the error patterns for the mth position of tracks i, j, and k of set A can be determined from the syndromes Sd^a_{m+i}, Sd^b_{m+15-k}, and Sv^a_m. We can write the equations for these syndromes from Eqs. (13–15). On eliminating the known zero-error patterns corresponding to the error-free tracks and the corrected error patterns up to position $(m-1)$ in each track, these equations can be written as

$$Sd^a_{m+i} = e^a_m(i), \tag{18}$$

$$Sd^b_{m+15-k} = \begin{cases} e^a_m(k) \oplus e^b_{m+15-y-k} & \text{if } y < 8, \\ e^a_m(k) & \text{if } y = 8 \text{ or set B is error-free,} \end{cases} \tag{19}$$

$$Sv^a_m = e^a_m(i) \oplus e^a_m(j) \oplus e^a_m(k). \tag{20}$$

From Eq. (17) we have

$$Sv^b_{m+15-y-k} = e^b_{m+15-y-k}(y). \tag{21}$$

If $Sv^b_{m+15-y-k}$ in Eq. (21) is nonzero for some $0 \le y \le 7$ and y is unknown, then set B must be processed first to identify the unknown erroneous track.

Equations (18–21) yield the error patterns

$$e^a_m(i) = Sd^a_{m+i}, \tag{22}$$

$$e^a_m(k) = \begin{cases} Sd^b_{m+15-k} \oplus Sv^b_{m+15-y-k} & \text{if } y < 8, \\ Sd^b_{m+15-k} & \text{if } y = 8 \text{ or set B is error-free,} \end{cases} \tag{23}$$

$$e^a_m(j) = Sv^a_m \oplus e^a_m(i) \oplus e^a_m(k). \tag{24}$$

The mth bits in tracks i, j, and k are then corrected, using these error patterns as

$$A_m(i) = \hat{A}_m(i) \oplus e^a_m(i), \tag{25}$$

$$A_m(j) = \hat{A}_m(j) \oplus e^a_m(j), \tag{26}$$

$$A_m(k) = \hat{A}_m(k) \oplus e^a_m(k). \tag{27}$$

Before proceeding to the correction of the next position, we must modify the syndromes affected by these corrections. The modification is shown by an arrow pointing from the previous value of a syndrome (with its modification) to its new value:

$$Sd^a_{m+i} \leftarrow Sd^a_{m+i} \oplus e^a_m(i), \tag{28}$$

$$Sd^a_{m+j} \leftarrow Sd^a_{m+j} \oplus e^a_m(j) \quad \text{if } j < 8, \tag{29}$$

$$Sd^a_{m+k} \leftarrow Sd^a_{m+k} \oplus e^a_m(k), \tag{30}$$

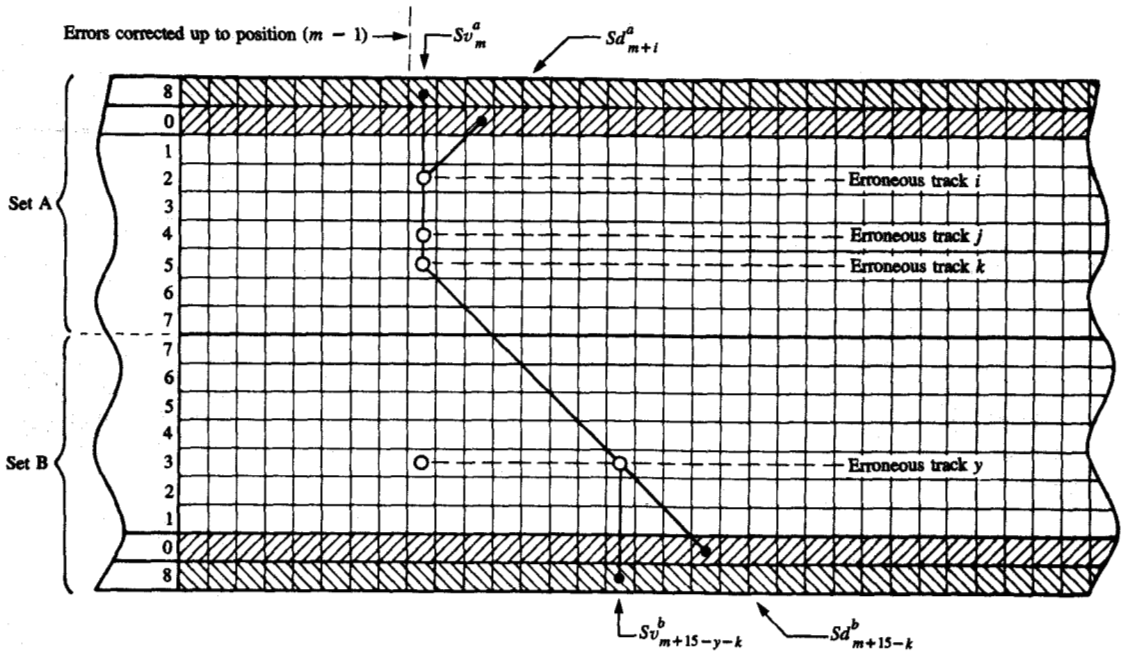

Figure 5

Example of three-track correction in set A.

$$Sd^b_{m+15-i} \leftarrow Sd^b_{m+15-i} \oplus e^a_m(i), \tag{31}$$

$$Sd^b_{m+15-j} \leftarrow Sd^b_{m+15-j} \oplus e^a_m(j) \qquad \text{if } j < 8, \tag{32}$$

$$Sd^b_{m+15-k} \leftarrow Sd^b_{m+15-k} \oplus e^a_m(k). \tag{33}$$

Now the decoding procedure can be applied to the next bit position by incrementing the value of m by 1. At the new bit position m, we have met the required condition that all errors up to bit position $(m-1)$ are corrected and the syndromes are adjusted for corrected error patterns. In particular, all error patterns affecting the syndrome value Sd^a_{m-1} are corrected. Consequently, we expect that

$$Sd^a_{m-1} = 0. \tag{34}$$

A nonzero value of Sd^a_{m-1} indicates the presence of uncorrected errors. This provides a partial check on the uncorrectable multitrack errors that are beyond the correction capability of the code.

• *Two-track correction in set A*

Errors in two known tracks in set A can be corrected if set B has at the most one unknown or two known erroneous tracks. The erroneous tracks in set A are indicated by track-error pointers i and j, where $i < j$.

The correction procedure for two known erroneous tracks is similar to that for three known erroneous tracks, except for the fact that the error pattern $e^a_m(k)$ corresponding to the track-error pointer k is presumed to be zero. The error patterns $e^a_m(i)$ and $e^a_m(j)$ for the tracks i and j can be calculated from the local syndromes Sd^a_{m+i} and Sv^a_m as may be seen from Eqs. (22) and (24). Thus, the two-track correction can be processed as a special case of the procedure for correction of three known erroneous tracks.

• *One-track correction in set A*

Errors confined to only one known track in set A can be corrected by using just the vertical-parity-check syndrome Sv^a_m of set A. Since the check Sv^a_m ranges over set A only, this correction capability is not affected by the error conditions in set B. The erroneous track in set A is indicated by a track-error pointer j.

The procedure for correction of one known erroneous track can also be integrated into the procedure for correction of three known tracks. In this case the error patterns $e^a_m(i)$ and $e^a_m(k)$ are presumed to be zero, and the error pattern $e^a_m(j)$ for the known erroneous track j is obtained from Eq. (24).

We pointed out before that the error conditions in set B do not affect the correction of one known erroneous track in set A. However, if set B is error-free or if it also has only one known erroneous track, then both the diagonal checks are available for detection of the beginning of errors in a new

track. This generation of track-error pointers is discussed in the following section.

Generation of track-error pointers

• *Pointer to first erroneous track in set A*

Errors confined to only one unknown track in set A can be detected and corrected if set B has, at most, one unknown or two known erroneous tracks. It is assumed that errors in all tracks in set B are corrected up to bit position $(m-1)$, and that the syndrome values are adjusted for all corrected error patterns. When all tracks in set A are error-free, the parity-check syndromes Sv^a_m and Sd^a_{m+i} are equal to zero for $0 < i < 7$. When any of these syndromes are found to be nonzero, it is an indication that an error is present in at least one of the tracks in the vicinity, that is, within the next seven bit positions. Assuming that only one erroneous track is affecting the syndromes, the index of the erroneous track can be determined by examining syndromes Sd^a_{m+7} and Sv^a_m as the bit position value m progresses. The following assertion characterizes the generation of the first track-error pointer in set A.

Assertion 1

Let m_1 and m_2 denote the lowest values of bit positions such that

$$Sd^a_{m+i} \neq 0 \qquad \text{for } m = m_1 \text{ and } i = 7, \tag{35}$$

$$Sv^a_m \neq 0 \qquad \text{for } m = m_2. \tag{36}$$

Then track q is in error at bit position m_2 and the track index q is given by

$$q = \begin{cases} 7 - (m_2 - m_1) & \text{if } m_2 \geq m_1, \\ 8 & \text{otherwise.} \end{cases} \tag{37}$$

The proof of Assertion 1 follows from the geometrical considerations depicted in **Figures 6** and **7.** Figure 6 considers the case in which m_2 is greater than or equal to m_1. Note that if the resulting value q in this case is smaller than zero, then the syndromes are affected by two or more unknown erroneous tracks and the errors are uncorrectable. Figure 7 considers the special case in which m_1 is not determined, even when the bit position value exceeds m_2, since an error in the vertical-parity-check track does not affect the cross-parity-syndrome.

The implementation of the above assertion fits in the general iterative decoding procedure as the bit position value m is incremented in an iterative manner. A counter is set to 7, when $Sd^a_{m+7} \neq 0$ is detected for the first time at $m = m_1$. The counter counts down by one each time m is incremented forward until bit position m_2 is reached, where $Sv^a_m \neq 0$. The resultant count value gives the index of the erroneous track. If the count goes below the value of zero, then the error is spread in more than one track and is uncorrectable. If $Sv^a_m \neq 0$ is detected first, and yet $Sd^a_{m+7} = 0$, then track 8 is in error.

Once we have the index value of the erroneous track, the errors can be corrected by applying the procedure for

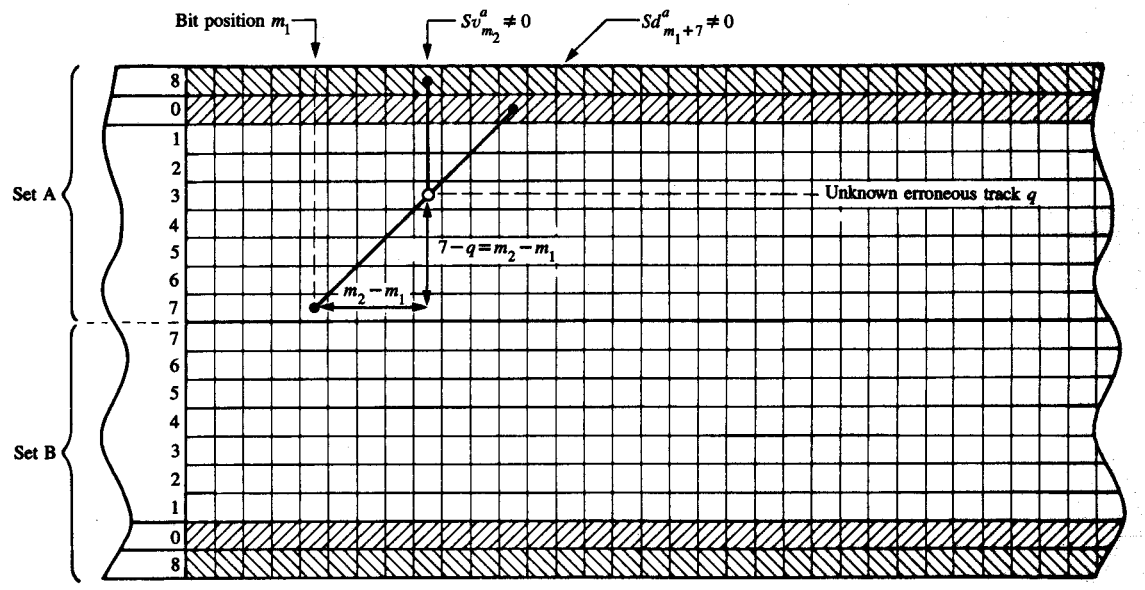

Figure 6

Generation of pointer to first erroneous track when $q \neq 8$.

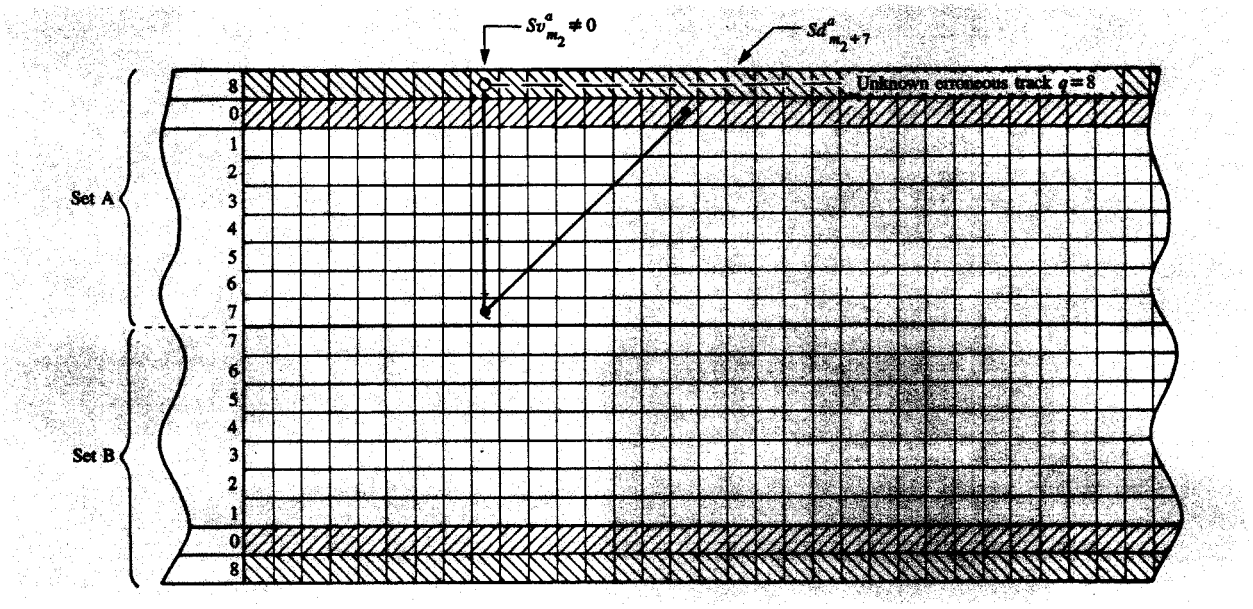

Figure 7

Generation of pointer to first erroneous track when $q = 8$.

correction of one track with the track-error pointer, as discussed previously.

• *Pointer to second erroneous track in set A*

Consider the case in which set A is being corrected for errors in a known erroneous track, and another unknown track in set A begins to be affected by errors. This second unknown erroneous track can be detected and both erroneous tracks of set A can be corrected provided that set B has at the most one known erroneous track.

For simplicity, we first explain the method for the case in which tracks 0 to 7 in set B are error-free. Later it will be easy to see how the equations can be modified to include the effect of a known erroneous track in set B.

Let p denote the known erroneous track in set A; and assume that, so far, all remaining tracks in set A have been error-free. Also assume that all errors have been corrected up to bit position $(m-1)$ and that the syndrome values have been adjusted for all corrected error patterns.

First we consider the case in which p is not the vertical-parity track, that is, $p \neq 8$. The error pattern of the mth position of track p affects the syndromes Sd^a_{m+p}, Sd^b_{m+15-p}, and Sv^a_m. In the absence of errors in any other tracks, we have

$$Sd^a_{m+p} = Sd^b_{m+15-p} = Sv^a_m = e^a_m(p). \tag{38}$$

However, as any one of the other tracks begins to be affected by an error, the syndrome relationship of Eq. (38) no longer holds. We can make the following assertion by observing the effect of the new erroneous track on this relationship.

Assertion 2

Let p denote the known erroneous track where $p \neq 8$. Let m_1 and m_2 denote the lowest bit positions such that

$$Sd^a_{m+i} \neq Sv^a_m \qquad \text{for } m = m_1 \text{ and } i = p, \tag{39}$$

$$Sd^b_{m+15-k} \neq Sv^a_m \qquad \text{for } m = m_2 \text{ and } k = p. \tag{40}$$

Then track q is in error starting at the bit position m, where m is the greater of m_1 and m_2; and index q is given by

$$q = \begin{cases} p - (m_2 - m_1) & \text{if } m_2 \neq m_1, \\ 8 & \text{if } m_2 = m_1. \end{cases} \tag{41}$$

The proof of Assertion 2 follows from the geometrical considerations depicted in **Figures 8, 9,** and **10.** Figure 8 considers the case in which m_2 is greater than m_1. Note in this case that if the resulting value q is smaller than 0, the syndromes are being affected by two or more unknown erroneous tracks and the errors are uncorrectable. Figure 9 considers the case in which m_2 is smaller than m_1. Here, if the resulting value q is greater than 7, the unknown erroneous track is in set B. This is detected and corrected later by the decoder in set B. Figure 10 considers the case in which m_2 is equal to m_1. Here, erroneous track q is the vertical-parity-check track of set A.

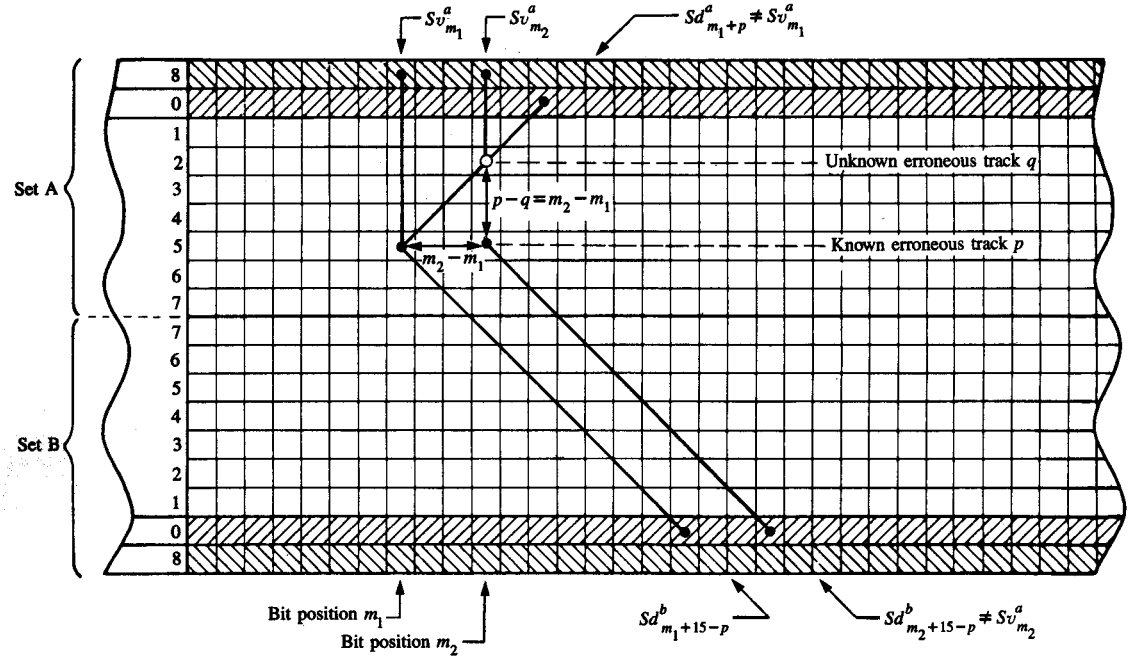

Figure 8

Generation of pointer to second erroneous track when $q < p$ and $p \neq 8$.

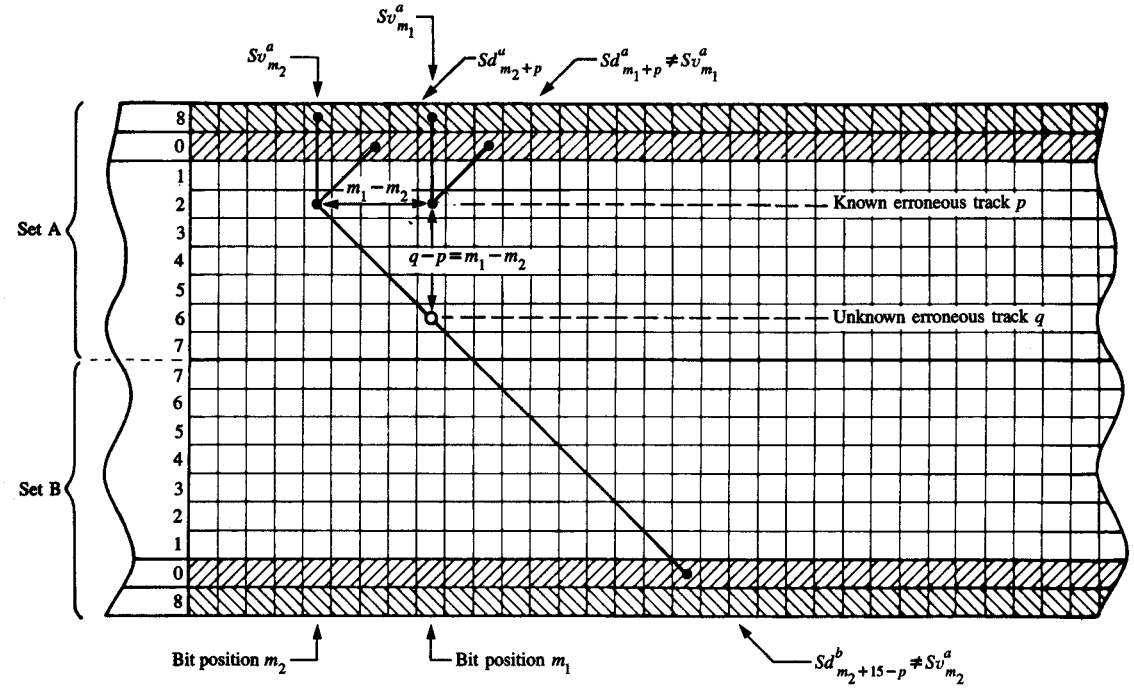

Figure 9

Generation of pointer to second erroneous track when $q > p$ and $p \neq 8$.

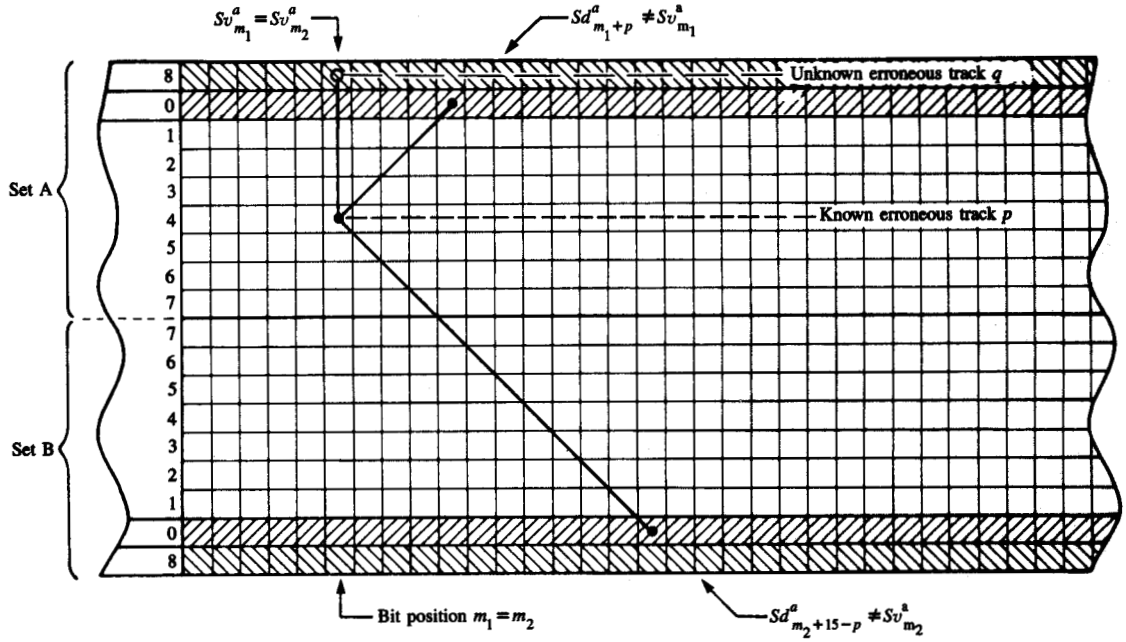

Figure 10

Generation of pointer to second erroneous track when $q = 8$.

The implementation of Assertion 2 fits into the general iterative decoding procedure. As the bit position value m is incremented, a synchronized counter counts up or down from value p depending on which one of the two inequalities, Eqs. (39) and (40), of Assertion 2 is satisfied first. The counting from set value p starts when one of the inequalities is satisfied and stops when the other inequality is also satisfied. The resultant count determines the index $q = p + (m_1 - m_2)$ or $q = p - (m_2 - m_1)$. The count value $q < 0$ is an indication of uncorrectable error, and the count value $q > 7$ is an indication that the unknown erroneous track is in set B. If both inequalities (39) and (40) are satisfied at the same bit position, track index q is taken to be 8 and the count value of the counter is ignored.

Next we consider the case in which the first unknown erroneous track is the vertical-parity-check track in set A, that is, $p = 8$. The error pattern of the mth position in track p in this case affects the syndrome Sv^a_m only. In the absence of errors in any other track in set A or set B, the cross-parity syndromes must all be zero. However, as any other track in set A begins to be affected by an error, the cross-parity syndromes are no longer zero as the bit position value m is incremented. We make the following assertion.

Assertion 3

Let p denote the known erroneous track where $p = 8$. Let m_1 be the lowest value of bit position such that

$$Sd^a_{m+i} \neq 0 \qquad \text{for } m = m_1 \text{ and } i = 7, \tag{42}$$

and let k be the smallest increment from bit position m_1 to m_2 such that

$$Sd^b_{m+15-k} \neq 0 \quad \text{for } m = m_2 \text{ and } k = 7 - (m_2 - m_1). \tag{43}$$

Then track q is in error beginning at bit position m_2, and index q is given by

$$q = k. \tag{44}$$

The proof of Assertion 3 follows from the geometrical considerations depicted in **Figure 11.** Note that if the resulting value of q is smaller than zero, the syndromes are affected by two or more unknown erroneous tracks and the errors are uncorrectable.

The implementation of Assertion 3 also follows the iterative decoding procedure. As the bit position value m is incremented, a counter counts k down from 7. The counter starts and stops counting when the inequalities (42) and (43), respectively, of Assertion 3 are satisfied. The final count determines the index of the unknown erroneous track. Note that one side of the inequality (43) is a function of the running-count value k in the counter at every bit-position value m.

Now we show the modification for the more general case in which set B has, at the most, one known erroneous track. Let y be the erroneous track in set B. The error patterns for this track are all known from the vertical-parity syndrome

Sv^b_m of set B. If $y \neq 8$, then these error patterns also affect the values of cross-parity-check syndromes Sd^b_m. In order to account for the effect of these error patterns, we use the adjusted value $Sd^b_m \oplus Sv^b_{m-y}$ in place of Sd^b_m for any required value of m. In particular, in Assertions 2 and 3, the syndrome Sd^b_{m+15-k} from set B is replaced by a composite syndrome SB, given by

$$SB = \begin{cases} Sd^b_{m+15-k} \oplus Sv^b_{m+15-y-k} & \text{if } y < 8, \\ Sd^b_{m+15-k} & \text{if } y = 8 \text{ or set } B \\ & \text{is error-free.} \end{cases} \quad (45)$$

If $Sv^b_{m+15-y-k}$ is nonzero for some $0 \leq y \leq 7$ and y is unknown, then set B must be processed first to identify the unknown erroneous track. The count value $q > 7$ in Assertion 2 indicates this condition.

Note that the syndromes used in the generation of track-error pointers, namely Sd^a_{m+i}, SB, and Sv^a_m, are the same in form as those used in Eqs. (22–24) for correction of errors. Thus, with appropriate values for the variables i and k, the same hardware can be used to generate syndromes for both processes. During the pointer-generation process, the variables i and k are assigned values as shown in **Table 1.**

Table 1 Assigned values for i and k in the pointer-generation process.

Pointer type	*Assertion*	*Assigned value of* i *and* k
First pointer	1	Running counter value, counting down from 7.
Second pointer with $p \neq 8$	2	First pointer value p.
Second pointer with $p = 8$	3	Running counter value, counting down from 7.

Table 2 Corresponding variables in set A and set B.

Set A	A_m	Sd^a_m	Sv^a_m	SA	i	j	k	e^a_m
Set B	B_m	Sd^b_m	Sv^b_m	SB	x	y	z	e^b_m

Error correction in set B

The coding rules possess a built-in mirror-image symmetry around set A and set B. In particular, the encoding and decoding equations for set B can be obtained from those of set A by substitution of the corresponding variables as shown in **Table 2.**

Thus the following types of errors are correctable in set B by applying the decoding process as described for set A:

1. Up to three known erroneous tracks in set B when set A is error-free or has at the most one known erroneous track.

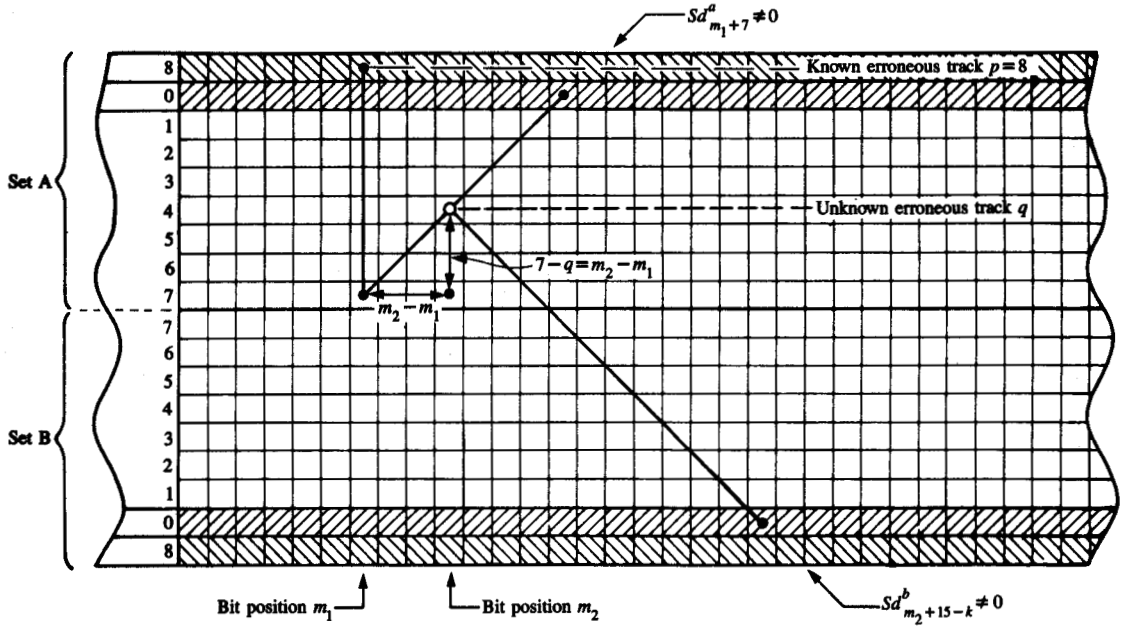

Figure 11

Generation of pointer to second erroneous track when $p=8$.

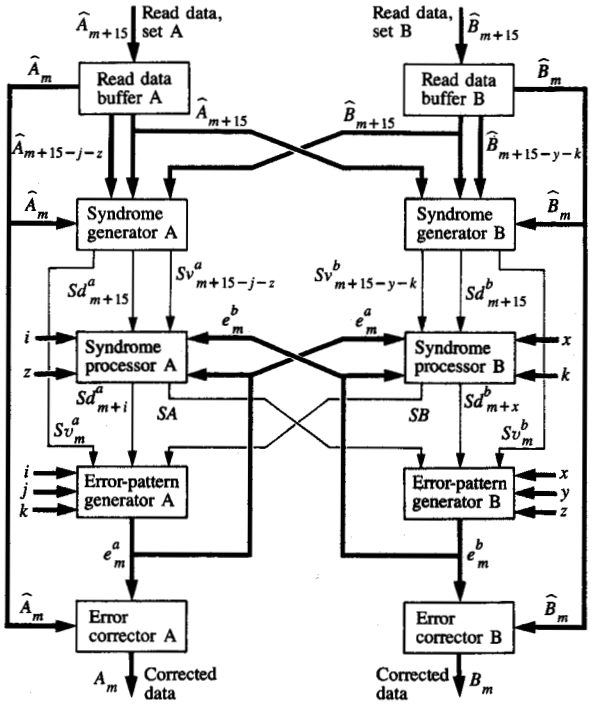

Figure 12

Adaptive cross-parity decoder.

2. Up to two known erroneous tracks in set B when set A has at the most one unknown or two known erroneous tracks.
3. Up to one known erroneous track in set B independent of set A.
4. Up to one unknown erroneous track in set B when set A has at the most one unknown or two known erroneous tracks.
5. Up to two erroneous tracks in set B (one of which is known) when set A has at the most one known erroneous track.

Implementation of the decoding process

The decoding process consists of two distinct functions: 1) the detection and identification of the erroneous tracks, and 2) correction of errors in known erroneous tracks. The internal pointer generator identifies the first erroneous track in both sets, and subsequently a second erroneous track in one of the two sets. Additional erroneous tracks may be detected and identified by external signals; these signals usually require some form of analog sensing of the playback conditions of the data recorded on the various tracks. This includes detection of loss of read signal, excessive phase error, inadmissible recording code patterns, or any other similar external indicators.

Once an internal or external track-error pointer is generated, it may be kept on for the entire remaining length of the record. Similarly, any track-error pointer may be turned off at an appropriate bit position in a record if the error patterns corresponding to the indicated track turn out to be zero consistently for a significant length of the record—thus confirming that the track is error-free. This allows replacement of the track-error pointer if and when some other track becomes erroneous, particularly in the case of long records.

It may be noted that the errors in a new erroneous track start affecting the decoding process as much as 15 bit positions ahead of the actual error. Thus the beginning of each new external track-error pointer must be extended 15 bit positions earlier by means of a pointer look-ahead function. For the same reason, in case of the internal generation of track-error pointers, the beginning of two erroneous tracks cannot always be detected successfully if they occur in proximity.

The error correction is done at the mth bit position, which progresses from the zero value to the last bit position value M in a recurring manner. To compute error patterns at the mth bit position, the syndrome values ranging from bit position m up to $(m + 15)$ are required. Furthermore, all errors up to bit position $(m - 1)$ must already be corrected and the corresponding syndrome values must be modified to account for the corrected error patterns.

Figure 12 shows the block diagram of the decoder, which consists of a read data buffer, syndrome generator, syndrome processor, error-pattern generator, error corrector, and pointer generator (not shown) for each of the two sets of nine parallel tracks. The functions and interrelationships of these blocks in set A are explained in the following.

The read data buffer for set A receives the read character $\hat{A}_{m+15}$ for storage in its empty storage location while the outgoing read character $\hat{A}_m$ in the farthest storage location is being corrected by the decoder. The syndrome generator for set A is a shift-register circuit similar to the encoding register shown in Fig. 4. The only difference is that the syndrome generator has an additional exclusive-OR circuit before the output for entering the read check bit from track zero. It receives two nine-bit read characters, $\hat{A}_{m+15}$ and $\hat{B}_{m+15}$, at the input and produces the syndrome Sd^a_{m+15} at the output. The syndrome generator also has access to any other read character in the buffer so that it can compute the modulo-2 sum of its components and deliver the corresponding vertical-parity syndrome as required by the syndrome processor.

The syndrome processor of set A is shown in **Figure 13.** This is a shift register that stores the syndrome values Sd^a_m through Sd^a_{m+15} in its storage positions 0 through 15, respectively. The AND/OR circuits connected to stages 0 through 7 select the particular syndrome Sd^a_{m+i} for the given pointer value i among the pointers i, j, and k for set A. The

syndrome Sd^a_{m+i} is forwarded to the error-pattern generator for set A. The AND/OR circuits connected to stages 8 through 15 select the syndrome Sd^a_{m+15-z} for the given pointer value z among the pointers x, y, and z for set B.

A composite syndrome SA is created by combining $Sv^a_{m+15-j-z}$ with Sd^a_{m+15-z}. The syndrome SA is forwarded to the error-pattern generator for set B. A corresponding composite syndrome SB is created by combining $Sv^b_{m+15-j-k}$ with Sd^b_{m+15-k} in the syndrome processor for set B. The syndrome SB is forwarded to the error-pattern generator for set A. The syndrome processor also modifies the stored syndrome values by applying the computed error-pattern values at the inputs of the appropriate stages as the value of m is incremented by shifting in a new value of Sd^a_{m+15}. After modification, the outgoing value of Sd^a_m is expected to be zero in the absence of any uncorrectable errors.

With appropriate assigned values of i and k in accordance with Table 1, the syndrome processors for set A and set B also provide syndromes for generation of the track-error pointers. The pointer generator (not shown in Fig. 12) for set A receives the syndromes Sd^a_{m+i}, SB, and Sv^a_m and generates the internal track-error pointers as described previously under "Generation of track-error pointers." The pointer generator continually updates the status of pointers by combining any external pointers, and creates the pointer vectors i, j, k for set A. It also provides pointer status signals such as "$j < 8$," "more than one track in error," and "three tracks in error."

The error-pattern generator for set A is shown in **Figure 14.** It receives the syndrome Sv^a_m from the syndrome generator for set A, and the syndromes Sd^a_{m+i} and SB from syndrome processors for set A and for set B, respectively. By appropriately combining these syndromes, it generates the error patterns $e^a_m(i)$, $e^a_m(j)$, and $e^a_m(k)$ for the mth bit position in tracks i, j, and k, respectively, in set A. The subsequent AND/OR circuit directs these error-pattern values to their appropriate track indexes according to the given pointer values i, j, and k. The input SB is inhibited and the error pattern $e^a_m(k)$ is forced to zero value by means of an AND gate when less than three tracks are in error. In addition, the input Sd^a_{m+i} is inhibited and error pattern $e^a_m(i)$ is set to a zero value when less than two tracks are in error. The output of the error-pattern generator is forwarded to the error-corrector network, where it is combined with A_m to produce the corrected character. It is also forwarded to the syndrome processor, where it modifies the stored syndrome values as mentioned before.

The decoding process progresses in synchronism with the incoming read data characters. The actual correction of each received data character is delayed by 15 bit positions.

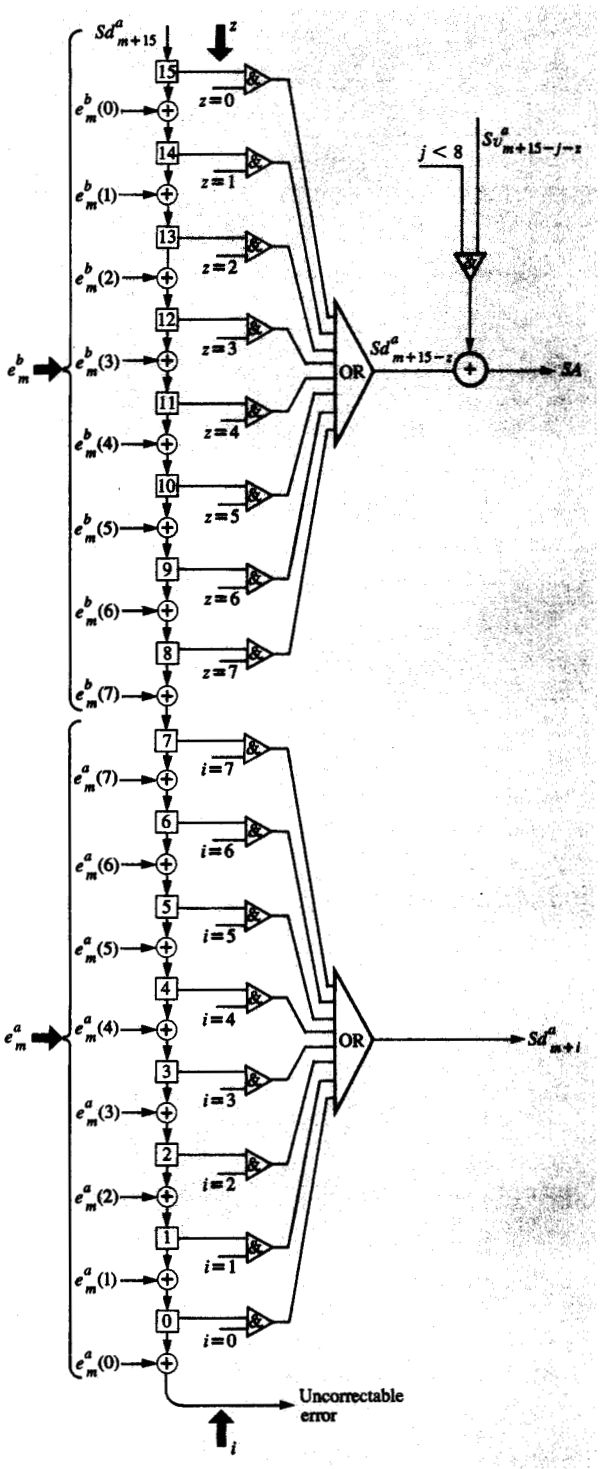

Figure 13

Syndrome processor for set A.

The general case

Thus far, we have focused on the special case of two sets of nine tracks in an 18-track system. However, it is easy to generalize the result to a system with any number of tracks in which the two sets may not have the same number of

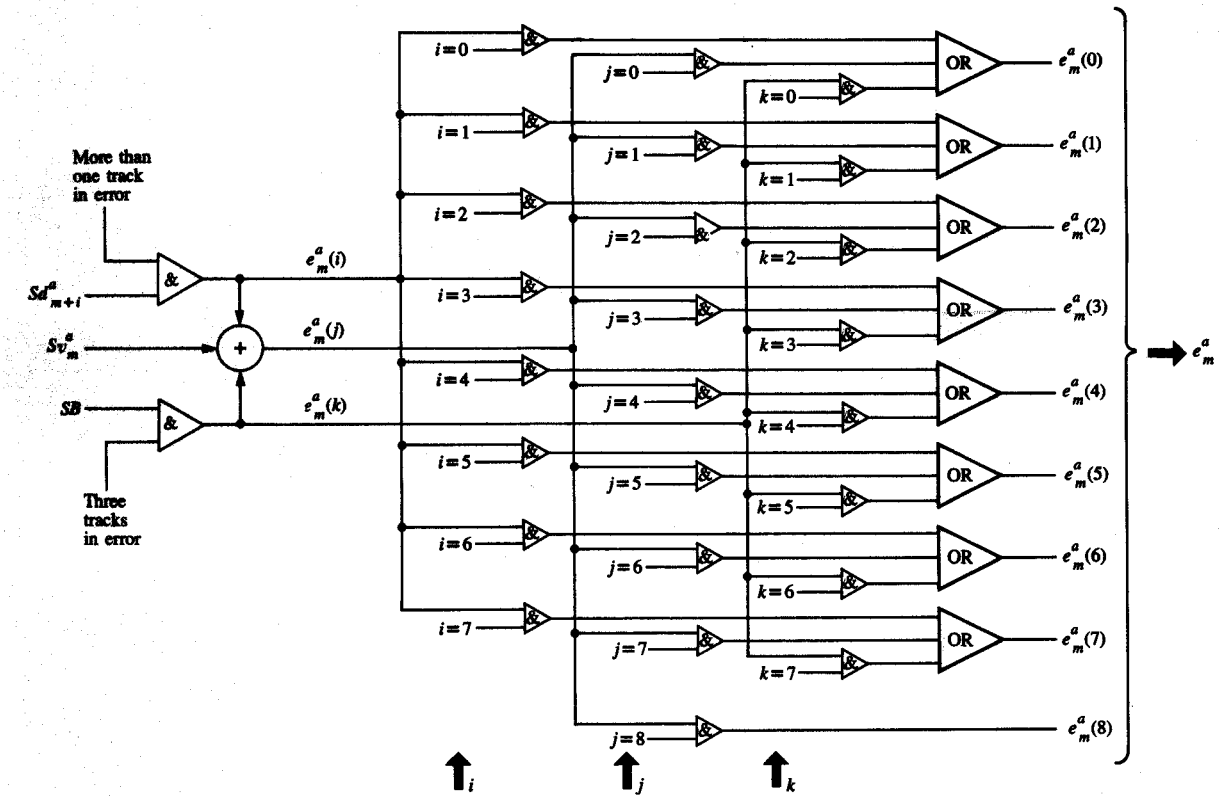

Figure 14

Error-pattern generator for set A.

tracks. Here, we give the encoding equations for such a system.

Suppose set A has $(T_1 + 2)$ tracks and set B has $(T_2 + 2)$ tracks. The encoding equations (3–6) can be written for this general case as

$$\sum_{t=0}^{T_1} A_{m-t}(t) \oplus \sum_{t=0}^{T_2} B_{m+t-T_1-T_2-1}(t) = 0, \tag{46}$$

$$\sum_{t=0}^{T_2} B_{m-t}(t) \oplus \sum_{t=0}^{T_1} A_{m+t-T_1-T_2-1}(t) = 0, \tag{47}$$

$$\sum_{t=0}^{T_1+1} A_m(t) = 0, \tag{48}$$

$$\sum_{t=0}^{T_2+1} B_m(t) = 0. \tag{49}$$

The decoding equations can be formulated for the general case in a similar manner. If T_2 is not equal to T_1, the decoding equations still remain substantially similar for set A and set B, which allows use of the same decoding hardware for the two sets by time-multiplexing the decoding process.

One effect of the increased number of tracks is the fact that the encoding and decoding processes involve corresponding numbers of bit positions along the tracks. This, in turn determines the size of the encoding and decoding buffers and the processing time delay. Another effect of the increased number of tracks is the corresponding increase in the number of additional check bits at the end of the record in order to complete the two cross-parity checks. In general, the two cross-parity check tracks are extended by $T_1 + T_2$ additional positions.

A further generalization is also possible in which a multitrack system is divided into three or more sets. Let T_1, T_2, T_3, $\cdots$ denote the number of data tracks in set A, set B, set C, $\cdots$, respectively. Each set has its own vertical-parity-check track. Two additional check tracks provide overall cross-parity checks along the diagonals with positive and negative slope, encompassing all data tracks. The total number of tracks in the system is, then,

$$2 + (T_1 + 1) + (T_2 + 1) + (T_3 + 1) + \cdots.$$

We illustrate the concept by giving the parity-check equations for a system of three sets. The geometrical representation of this system is shown in **Figure 15.**

The parity check along the diagonal with positive slope, called the d_m^+ check, is recorded in track 0 on the side of the first set. The corresponding encoding equation is given by

$$A_m(0) = \sum_{t=1}^{T_1} A_{m-t}(t) \oplus \sum_{t=1}^{T_2} B_{m-t-T_1}(t) \oplus \sum_{t=1}^{T_3} C_{m-t-T_1-T_2}(t). \quad (50)$$

The parity check along the diagonal with negative slope, called the d_m^- check, is recorded in track 0 on the side of the last set. The corresponding encoding equation is given by

$$C_m(0) = \sum_{t=1}^{T_3} C_{m+t-T_3-1}(t) \oplus \sum_{t=1}^{T_2} B_{m+t-T_3-T_2-1}(t) \oplus \sum_{t=1}^{T_1} A_{m+t-T_3-T_2-T_1-1}(t). \quad (51)$$

The vertical-parity check is recorded in a separate check track for each set. The corresponding encoding equations are

$$A_m(T_1 + 1) = \sum_{t=0}^{T_1} A_m(t), \quad (52)$$

$$B_m(T_2 + 1) = \sum_{t=1}^{T_2} B_m(t), \quad (53)$$

$$C_m(T_3 + 1) = \sum_{t=0}^{T_3} C_m(t). \quad (54)$$

These parity checks provide the following error-correction capability:

1. The vertical-parity check in each set provides correction of one known erroneous track in that set.
2. The vertical-parity check of a set in conjunction with one of the two cross-parity checks provides correction of two known erroneous tracks in that set.
3. The vertical-parity check of a set in conjunction with one of the two cross-parity checks provides detection and correction of one unknown erroneous track in that set.
4. The vertical-parity check of a set in conjunction with both the cross-parity checks provides correction of up to three known erroneous tracks in that set.
5. The vertical-parity check of a set in conjunction with both cross-parity checks provides correction of one known and another unknown erroneous track in that set.

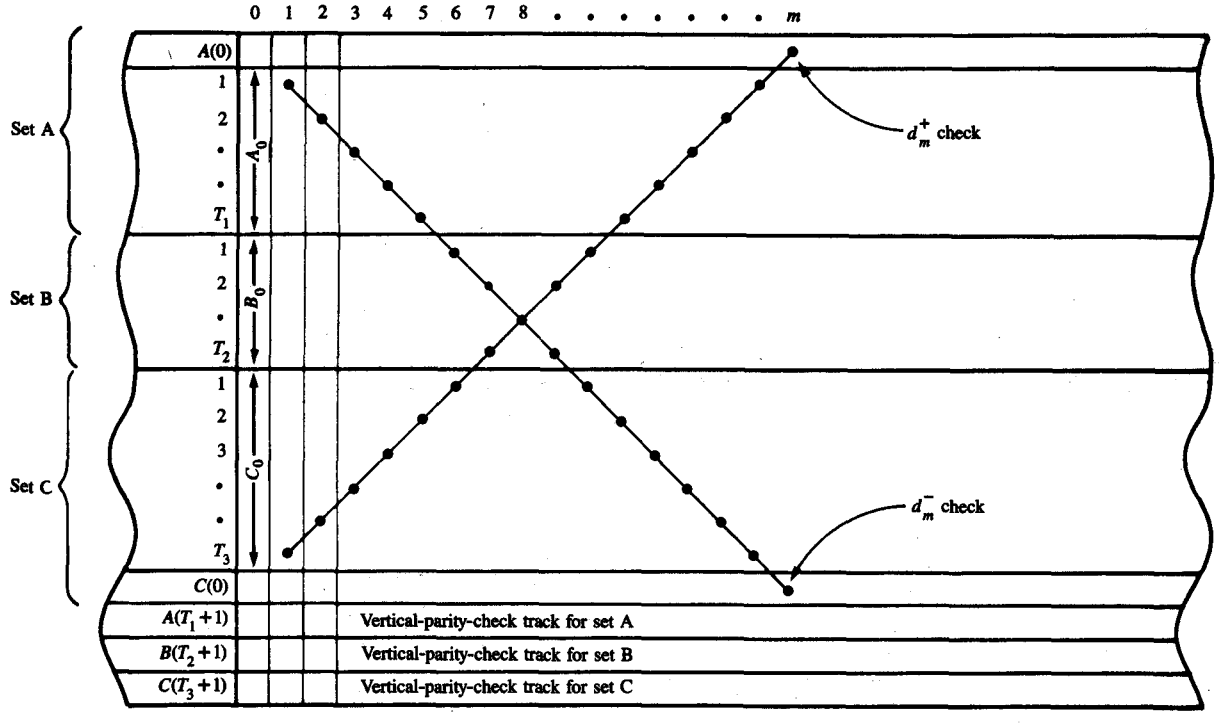

Figure 15

Cross-parity check in a system of three sets.

Each of the two cross-parity checks can aid in the error detection and/or correction in one set only. Thus the error-correction capability, as described in items 4 and 5 above, is available to any one set only. Similarly, the error-correction capability, as described in items 2 and 3 above, is available to any two sets only. The decoding equations can be formulated by considering various track-error combinations, as was done for the special case.

The proof for this generalization follows from Assertions 1, 2, and 3 and the three-track-correction procedure. In particular, if there are no errors in any set other than the one being corrected, Assertions 1, 2, and 3 and the three-track-correction procedure remain unaffected by the number of sets in the scheme. In case of a known erroneous track in any of the other sets, the vertical-parity check in that set provides correction for the erroneous track and the cross-parity syndromes are adjusted for these corrections. Thus Assertions 1, 2, and 3 and the three-track-correction procedure use composite syndromes [see Eqs. (23) and (45)] which include vertical-parity adjustments from all other sets affecting the cross-parity syndromes. Furthermore, both cross-parity syndromes require such adjustments depending on the relative placement of the set being corrected and the other sets.

In this general case, we have changed the method of ordering the tracks as compared to the one used in the special case of Fig. 1. Also, we have made use of the two cross-parity checks independently of each other by excluding one from the calculations of the other. These changes are made for mathematical convenience. Furthermore, Fig. 15 is only a geometrical representation of tracks corresponding to the algebraic coding equations. Any other suitable order can be used for the actual placement of the tracks on the tape.

Conclusions

In this report we have presented a scheme of coding data in an 18-track tape system that meets the main objective of providing substantially improved error-correction capability without a corresponding increase in redundancy or implementation cost as compared to that in Models 4, 6, and 8 of the IBM 3420 tape machines. The concept of adaptive usage of redundancy provides the necessary extension of the error-correction capability, and the concept of iterative decoding of interacting vertical- and cross-parity checks provides a simple and inexpensive implementation. The three different data formats described herein illustrate how the data and check bits can be arranged in various ways—both along and across the tracks. A general system is developed in which the redundancy is applied to a multiple number of unequal sets in an adaptive manner.

Acknowledgments

The author gratefully acknowledges the support and encouragement provided by various members of the advanced tape development groups at the IBM San Jose and Tucson development labs. Thanks and appreciation are also due the IBM San Jose management for providing ample support and freedom for creative endeavors which freely crossed IBM location boundaries.

References

1. D. T. Brown and F. F. Sellers, Jr., "Error Correction for IBM 800-Bit-per-Inch Magnetic Tape," *IBM J. Res. Develop.* **14,** No. 4, 384–389 (July 1970).
2. A. M. Patel and S. J. Hong, "Optimal Rectangular Code for High Density Magnetic Tapes," *IBM J. Res. Develop.* **18,** No. 6, 579–588 (November 1974).
3. Arvind M. Patel, "Error Recovery Scheme for the IBM 3850 Mass Storage System," *IBM J. Res. Develop.* **24,** No. 1, 32–42 (January 1980).
4. P. Prusinkiewicz and S. Budkowski, "A Double-Track Error-Correction Code for Magnetic Tape," *IEEE Trans. Computers* **C-19,** 642–645 (June 1976).
5. A. M. Patel, "Multitrack Error Correction with Cross-Parity Check Coding," *IBM Technical Report TR02.813*, San Jose, CA, March 20, 1978, available from the author on request. Also U.S. Patent No. 4,205,324, May 27, 1980, and IEEE Catalog No. 85CH 2201-2, papers presented at the International Symposium on Information Theory, Brighton, England, June 24–28, 1985.
6. A. M. Patel, "Plural Channel Error Correcting Methods and Means Using Adaptive Reallocation of Redundant Channels Among Groups of Channels," U.S. Patent No. 4,201,976, May 6, 1980.
7. A. M. Patel, "Improved Encoder and Decoder for a Byte-Oriented (0, 3) 8/9 Code," *IBM Tech. Disclosure Bull.* **28,** No. 5, 1938–1940 (October 1985).

Received February 22, 1985; revised May 20, 1985

Arvind M. Patel *IBM General Products Division, 5600 Cottle Road, San Jose, California 95193.* Dr. Patel is a senior technical staff member in the Magnetic Recording Institute, a GPD/Research organization at San Jose. He is currently involved with the development of computer storage products using magnetic recording technology. He received his B.E. from Sardar Vallabh-Bhai Vidyapeeth, India, in 1959, his M.S. from the University of Illinois, Urbana, in 1961, and his Ph.D. from the University of Colorado at Boulder in 1969, all in electrical engineering. Dr. Patel joined IBM at the Poughkeepsie, New York, laboratory in 1962. Since then he has worked on various aspects of magnetic recording technology and product development projects in the Poughkeepsie, Boulder, and San Jose laboratories. His main theoretical interest has been in exploring the area of information theory and coding for computer applications. His work on data encoding and error-correcting codes has won him four Outstanding Invention Awards from IBM in 1972, 1973, 1983, and 1985, and an Outstanding Technical Paper Award from the American Federation of Information Processing Societies in 1970. Dr. Patel has been elected a Fellow of the Institute of Electrical and Electronics Engineers with the citation: "For contributions to data encoding/decoding and error correction and their application to magnetic storage devices."